Handelshøyskolen BI

GRA 19703 Master Thesis

Thesis Master of Science 100% - W

Predefinert informasjon

Startdato:	16-01-2022 09:00	**Termin:**	202210
Sluttdato:	01-07-2022 12:00	**Vurderingsform:**	Norsk 6-trinns skala (A-F)
Eksamensform:	T		
Flowkode:	202210||10936||IN00||W||T		
Intern sensor:	(Anonymisert)		

Deltaker

Navn:	Alexandros Stee Peratinos og Johannes Piene

Informasjon fra deltaker

Tittel *:	Optimal Location of Electric Vehicle Charging Stations in Norway using the Flow refueling Location Model
Navn på veileder *:	Karim Tamssaouet

Inneholder besvarelsen konfidensielt materiale?:	Nei	**Kan besvarelsen offentliggjøres?:**	Ja

Gruppe

Gruppenavn:	(Anonymisert)
Gruppenummer:	152
Andre medlemmer i gruppen:	

Optimal Location of Electric Vehicle Charging Stations in Norway using the Flow refueling Location Model

By

Johannes Piene

Alexandros Peratinos

And Supervised By

Dr. Karim Tamssaouet

Abstract

This thesis investigates optimal locations for charging stations across the road network of Norway. To solve this problem, we deploy a flow-refueling location model, which considers the optimal locations based on traffic flows of inter-city and other long distance travel. To solve the model for a large network, such as the Norwegian, heuristic approaches are necessary. We have constructed a greedy-adding algorithm with and without substitutions and a genetic algorithm to solve the problem. These are compared and used together to produce quality solutions in an effective manner. We find that the current system of charging stations are sufficient to cover all the demand for longer electric vehicle ranges, but not sufficient for the shortest ranges below 200km

BI Norwegian Business School

Department of accounting, auditing and business analytics -

Master of Science in Business Analytics

2022

Acknowledgements

We would like to thank BI Norwegian Business School for five both challenging and rewarding years, and give a special thanks to the Department of Accounting, Auditing and Business Analytics for providing us the knowledge needed for this thesis. We would like to express our gratitude to our supervisor, Dr. Karim Tamssaouet, for his insightful comments and feedback.

Contents

1 Introduction

This section will start by briefly introducing the background and motivation behind the location of electric vehicle charging stations. Following this, the problem is defined, and the objective of the thesis is presented.

1.1 Background

For a long time, electric vehicles (EVs) have been regarded as one of the most propitious ways to reduce C02 emissions produced by road traffic. Many countries have begun implementing aggressive targets for the future adoption of EVs (Perkowski, 2017), and some have even set an end date for the sale of internal combustion engine vehicles (ICEV) (Petroff, 2017). One of these countries is Norway.

In the National Transport Plan 2018-2029, the Norwegian government proposed that all new passenger and light vans sold as of 2025 must be zero-emission vehicles (Ministry of Transport, 2017). This aspiring goal means that zero-emission vehicles must be able to replace all the uses that ICEVs currently hold, and it may not be reached unless the current infrastructure is improved. As transport stands for about 20% of the global C02 emissions, whereas roughly 75% of this stems from road transportation (Ritchie et al., 2020), it is apparent that making EVs a more viable option can drastically reduce the C02 emissions going forward.

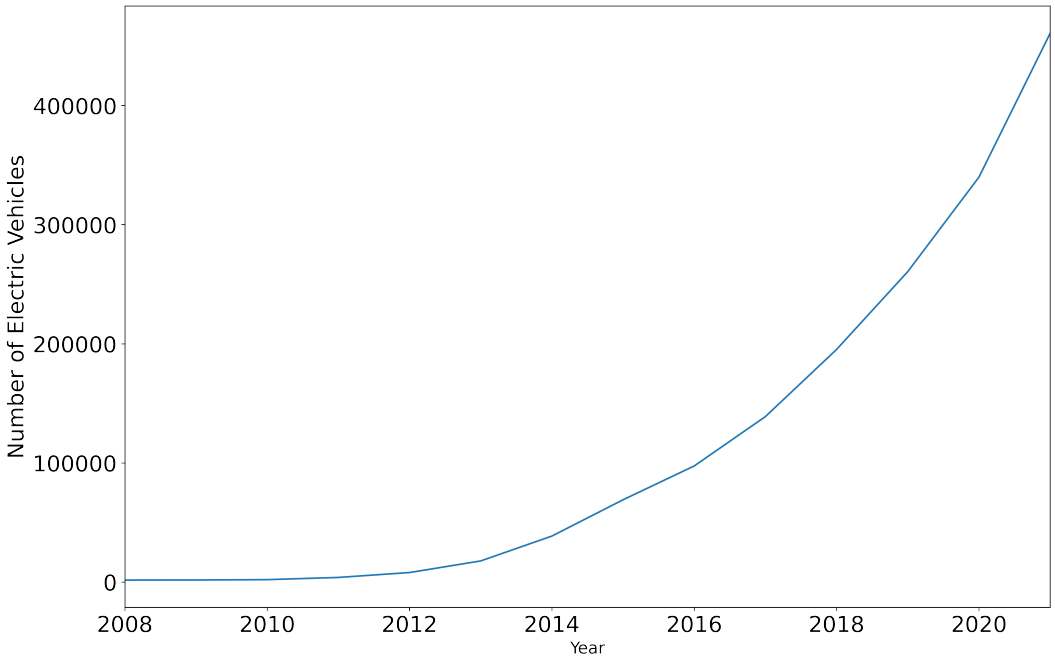


Figure 1: Electric passenger cars registered in Norway (Statistics Norway, 2022a)

In recent years, the number of registered EVs in Norway has increased considerably, starting at 1 693 in 2008, while there being over 460 000 in 2021. As

illustrated in Figure 1, it is apparent that there is a growing demand for EVs in Norway. However, certain barriers must be addressed until electric vehicles can fully replace ICEVs as primary vehicles. As of 2018, the leading EV user group has been households with multiple vehicles, while only a tiny portion of single-vehicle households prefer EVs over ICEVs. Therefore, to meet the proposed 2025 goal, single-vehicle households and multi-vehicle households must adopt EVs as their primary vehicle (Figenbaum, 2018). Although the demand for EVs is increasing, this has mainly resulted from economic incentives given to EV owners by the Norwegian Government. In 2018, the Norwegian Center for Transport Research released a report stating that mastering long-distance travel is one of the last obstacles that remain before electric vehicles can be mass adopted as the primary vehicle type in Norway (Figenbaum, 2018). In light of Figure 2 below, it is evident that mass adoption has yet to occur, which implies that challenges related to long-distance travel must be resolved before it is expected that EVs can fully replace the role of ICEVs. As of 2021, only 19% of all registered vehicles in Norway are EVs.

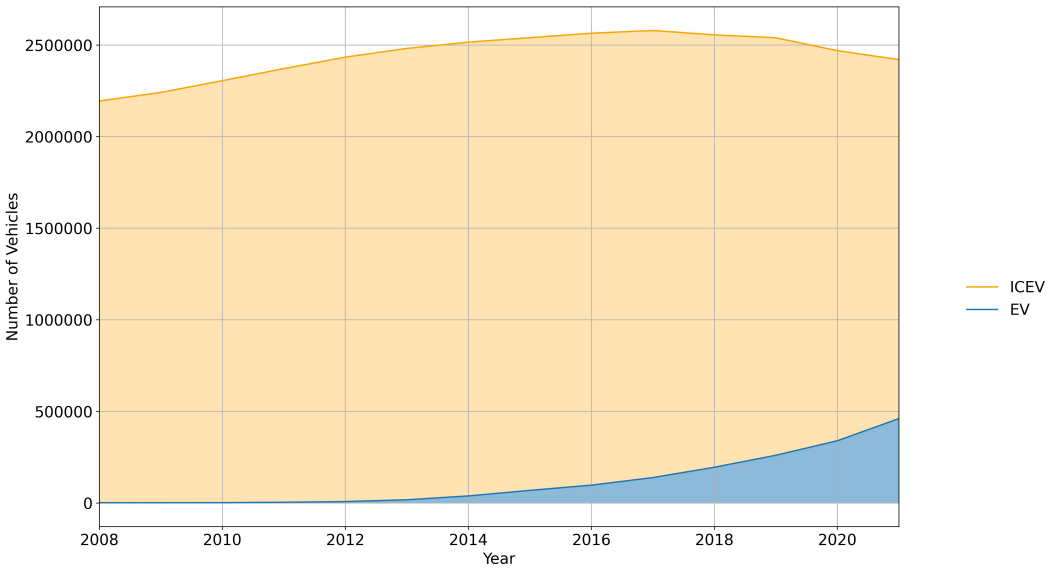


Figure 2: Total number of registered vehicles compared to EVs in Norway

One of the challenges related to long-distance travel is how to locate fast-charging stations (hereby known as CSs), so that older, less mobile EVs can be used effectively. Although the Norwegian government is world-leading when it comes to incentivising the use of EVs, both from an economic and behavioural point of view, newer models with a longer range are still considerably more expensive than the older ones. Many would therefore prefer to obtain a cheaper model with a lower range. These models may struggle to complete some long-distance routes in Norway, as the current infrastructure is sub par. For example, traveling from Lekanger to Nesna can be a significant challenge for an EV with a range less than 200km as the 250km long route lacks the required amount of CSs to ensure reachability(Christiansen, 2018).

Thus, sufficient infrastructure is required before EVs can be deployed on a large scale. Today, after centuries of ICEV use, refueling stations can be said to be omnipresent, as drivers of ICEVs hardly ever find themselves in a situation where a refueling station is out of reach. Notwithstanding, this is not the case when it comes to the location of CSs. EV Norway states that, as of February 2022, the current infrastructure is only capable of fast charging 4 600 vehicles simultaneously across the entire country of Norway (Elbil.no, 2022). This lack of recharging infrastructure may limit the utilization of EVs, restricting their adoption. As the EV user base is smaller compared to ICEVs, the demand is, by implication, also smaller. Therefore, the incentive to develop a functioning CS network is considerably lower. Many recognize this challenge as the "chicken or the egg" dilemma. (Chung & Kwon, 2015; MirHassani & Ebrazi, 2013). While the customer's adoption of EVs heavily relies on the already existing charging availability and infrastructure, market-driven investments in the infrastructure are not likely to happen until a certain level of adoption has occurred. Although we already see private actors like Tesla investing in infrastructure, the primary responsibility should still lie with the Norwegian government, as the current system is arguably inadequate for the mass adoption of EVs.

1.2 Problem statement and thesis contribution

The main objective of this thesis is to identify the optimal locations for CSs across the Norwegian road network using the Flow-Refueling Location Model (FLRM), as proposed by (Kuby & Lim, 2005). This model seeks to optimally place a given set of refueling stations in a geographical network to maximize the traffic flow between a set of prespecified origin and destination pairs (O-D pairs). In this context, a flow indicates the demand for CSs along the shortest path between each O-D pair. Therefore, CSs are placed in a way that seeks to maximize the total demand captured. The model assumes that there is no deviation from the shortest paths.

Improving the current infrastructure concerning long-distance travel is the main focus of this thesis, given that this seems to be the main challenge until full adoption of EVs can occur. Improving the infrastructure implies that the system must be able to compete with the current ICEV infrastructure. Regular charging stations can not compete with refueling stations when considering long-distance travel due to the inconveniently long recharging time. For this reason, we only consider fast charging stations capable of supplying an output of 40 kW or more.

It is also important to note that this thesis does not consider battery capacity in terms of kWh. Instead, battery capacities are translated into a range distance in kilometers. The EV ranges considered in this thesis are 170, 325,

and 585km, equivalent to battery capacities of 30, 45 and 95 kWh, respectively. The three EV ranges can be considered short, medium, and long-range.

Although the FRLM can be solved as a MILP problem, finding a globally optimal solution for large networks is generally not feasible. Thus, this thesis seeks to solve the problem by constructing three different heuristic algorithms: greedy-adding, greedy-adding with substitution, and the genetic algorithm. To assess the performance of the algorithms compared to the MILP solution, we have constructed a small test network with 25 nodes.

Finally, the proposed optimal solutions are compared with the current location of universal public CSs in Norway, which will indicate where the current system lacks a supply of CS and whether certain stations should be replaced to more effectively cover the demand. However, the universal stations are supplied by several competing operators, and it is not necessarily in their best interest to cooperate on where to build new CSs. Hence, we should expect the current system to deviate from the more effective solutions. Tesla has more than 80 CS in Norway, which until recently only provided charging for their cars, but as of mid-2022, they opened most of their network to the universal standard used by the other EV brands. The universal Tesla chargers will therefore be included in the set of existing CSs to which we compare our solutions.

2 Literature review

This section will introduce previous research on facility location models most relevant for alternative-fuel refueling stations. The study of facility location theory dates back to 1909, and since then, researchers from various fields have created new models while investigating extensions of already existing ones (Owen & Daskin, 1998). The general idea behind facility location models is to determine the optimal location for a set of facilities, given certain constraints. Most facility location problems aim to maximize profit, minimize distance or cover the most demand possible. Research related to the location of alternative-fuel refueling stations has seen massive growth in recent years and has been explored in all parts of the world, with the exception of Africa (Pagany et al., 2019).

2.1 Facility Location Problems

One of the most common facility location problems is the *p-median problem* (PMP). The PMP is a distance-based optimization problem that seeks to minimize the total *demand-weighted travel distance* between demand points and their corresponding facilities by locating p facilities (Owen & Daskin, 1998). Today, the applications of the PMP includes allocation of warehouses (Baumol

& Wolfe, 1958; Dejax, 1988), fire stations (Plane & Hendrick, 1977; Yao et al., 2019) and shopping centers (Nwogugu, 2006), to mention a few. Both linear programming and heuristics are common approaches to solve the PMP (Gwalani et al., 2021).

Another common facility location problem is the *Set Covering Problem* (SCP). The SCP is a classic *combinatorial optimization problem* similar to the p-median problem, but instead of minimizing the average distance traveled, the number of facilities to be located is minimized. Further, a *maximum service distance* is set so that all demand points are covered by at least one facility that is reachable within this distance(Owen & Daskin, 1998). The SCP has many real-life applications, ranging from vehicle routing (Cacchiani et al., 2014) to conservation biology (Moore et al., 2003).

2.2 Flow Refueling Models

Common facility location problems, like the PMP and SCP, assume that the demand originates in different point-based nodes. Using this assumption, the distance of interest is on the path from the demand node to the closest facility. In this lies a notion that travel is conducted with the primary purpose of reaching the facility.

The nature of customer demand for alternative fuel, however, differs from the demand for standard commodities. In most cases, refueling a vehicle is mainly exercised while traveling to a predefined destination for purposes other than refueling the vehicle. Thus, it may be inappropriate to express the demand at nodes. When goods and services are obtained during a trip, a more realistic way to model the demand may be as flows of customers passing through the nodes. Some previous methods evaluated locations based on passing traffic counts. However, this may cause double counting of traffic flows and fails to consider the problem of locating facilities too close to each other, resulting in self-cannibalization. (Hodgson, 1990) implemented this assumption in a flow model called the *flow capturing facility-allocation model* (FCLM). It uses a network of nodes that represents all potential facility locations. Each pair of nodes, termed origin-destination (O-D) pairs, has a path between them with an assumed flow representing the demand. A flow is captured, i.e., the demand on the path is served only if a facility is located somewhere along the shortest path between an O-D pair. One facility along a path is assumed to be able to capture the entire flow for the given path. The model further assumes that all flows are generated on the path between the O-D pairs and that all drivers make choices in a system-optimal manner and do not deviate from the shortest path. An origin node i and destination node j is given a single subscript as O-D pair q. The objective function aims to allocate locations for a given set of facilities that maximize the combined flow between all O-D pairs.

Facilities can be restricted to be located at intersections rather than midway along paths. This allows the facility to capture all flows crossing through the intersection and its connected arcs. The model can be solved by binary linear programming or by heuristic approaches. An intuitive greedy algorithm starts by locating facilities at the most heavily traveled node, calculates non-captured flow by removing captured flow from consideration, and selects the node most heavily traveled by the non-captured flow. This procedure is repeated until the selected number of facilities have been located.

(Kuby & Lim, 2005) proposed a more specialized flow model for the problem of refueling vehicles, namely the *flow refueling location model* (FRLM). In this model, the vehicle range becomes a key factor for considering whether a flow through a path is captured. While the basic assumptions are the same, some constraints are modified to account for a vehicle's range such that multiple facilities may be needed along each path to capture the flow. Suppose the range of the vehicle is shorter than the distance between a facility and the O-D node. In this case, the vehicle will not be able to complete its traveling distance as it will run out of fuel before completing the route. This implies that the actual facility location in an O-D path matters, unlike the FCLM, where facilities anywhere along the path will suffice to capture the flow. For this reason, restricting the facilities to be located at intersections will not guarantee all crossing paths to be captured and adding facilities midway between two junction nodes can increase the number of viable solutions.

(Kuby et al., 2009) uses the concepts of the FRLM and further includes capacitation at each facility in the *capacitated flow refueling location model* (CFRLM). Each facility's refueling capacity is limited to an exogenously set value. With this constraint, nodes with large flows passing through may not be sufficiently refueled by 1 station, and the optimal solution may be to locate multiple stations at the most trafficked nodes. They also performed a case study with the model, using data from a simplified road network of Arizona highways. The flows between nodes were allocated based on the population at each node, using a gravity model, and test solutions with a different number of facilities to be placed. By adding the capacity constraint, they aim to resolve one of the FRLM's shortcomings, providing a more realistic representation of refueling limitations. Nonetheless, the model is limited by assumptions that should be addressed before the CFRLM becomes a truly realistic model. The model also assumes a uniform flow, whereas vehicle traffic has a significant temporal variation. If a refueling facility can barely handle the average refueling demands over a given period, this temporal variation would put it over its capacity during peak periods. Furthermore, a considerable downside of the flow models presented is that they only take the EV users into account, and the costs of setting up these stations and distribution of power to the stations are not included.

2.3 Heuristic approaches for large location problems

One of the first heuristic algorithms used for solving location problems is *the vertex substitution algorithm*, developed by (Teitz & Bart, 1968). This algorithm swaps facilities from the non-selected set of facilities to a selected set in iterations until an optimal local solution is found. As this algorithm lacks a method to break out of the local optima, it is often run multiple times with different randomly generated sets of selected facilities to increase the chance of reaching a global optimum (Kuby et al., 2009). The vertex substitution algorithm is considered to be significantly more computationally demanding than more recent heuristic algorithms.

The *greedy-adding algorithm* (GAAL) is a recent, more frequently used algorithm for solving location problems. Instead of finding the optimal solution that maximizes the total flow volume that can be refueled in one iteration, the GAAL adds one facility at a time to maximize the additional flow covered in the selected set. For each iteration, the facility resulting in the highest increase of the objective value is added to the selected set. This process is repeated until a specified amount of facilities have been included (Daskin, 1997). The term greedy is used because the algorithm only considers the best solution for the current iteration when selecting a facility. A variant of this algorithm is the *greedy-adding algorithm with substitution* (GAAL with substitution). This algorithm performs the same steps as the GAAL, but it also attempts to substitute one selected facility at a time with other facilities. If the objective value increases, the substitution associated with the most significant improvement is kept in the solution for the next iteration. Thus, the GAAL with substitution usually performs better than the GAAL as it is better equipped to break out of local optimums. However, this is more computationally demanding, and the substitutions often lead to significantly longer runtimes. Both the GAAL and GAAL with substitution have previously been used to solve the maximal covering problem (Church & ReVelle, 1974) and the set covering problem (Chvatal, 1979).

The *genetic algorithm* (GA) is another heuristic inspired by the evolutionary theory in biological organisms. Like natural selection of the fittest gene pools over time in a population, the GA generates multiple generations of candidate solutions that change from merging and randomly mutating. For each iteration, only the best solutions are kept in the solution domain, and the algorithm continues to the next iteration (Mitchell, 1998). When working with the GA, the term *chromosome* may refer to a candidate solution consisting of smaller pieces called *genes*. For facility location problems, a gene refers to a specific facility location, and a chromosome is a set of facility locations that constitute a possible solution. When beginning the algorithm, an initial population of random chromosomes is generated. The chromosomes are further

separated into two sub-populations before a crossover operation of the genes between them is performed. Next, a new member selection takes place by evaluating each chromosome with a fitness function (equivalent to an objective function for location problems). The solutions with the best fitness values are kept in the population following an exogenously determined population limit. Before repeating this process with this new generation of chromosomes, mutations of genes may occur in a certain percentage of the population. This mechanism is included to help escape local optimums that may restrict new generations from improving their fitness. (Efthymiou et al., 2017) use a GA to allocate charging stations in the city of Thessaloniki. The algorithm accounted for the demand at a predefined set of candidate location nodes and the cost-weighted distances between locations.

(Kuby & Lim, 2010) developed heuristic algorithms per GAAL, GAAL with substitution, and GA and used them to solve the FRLM. They conducted a case study, optimally locating alternative refueling stations in Florida. They found that relatively standard implementations of the algorithms produced reasonably good results for a complex problem, except when the vehicle ranges were short compared to the length of the path edges. In general, the genetic algorithm produced slightly better results at the cost of significantly longer runtimes.

3 Data Collection and Preparation

This section describes the data collection process and how the data was managed and prepared to give the best representation of the current Norwegian road network and environment.

3.1 Data Collection

The data used in this thesis was collected from various publicly available sources. The road network of Norway was retrieved from the Norwegian Public Roads Administration (NPRA) through GeoNorge, the national website for map data and location information in Norway. At Geonorge.no, the dataset Road network for routing by the National roads database was collected (GeoNorge, 2022). This dataset represents the road network of Norway as nodes (locations) and edges (roads leading from and to each location). It also contains important characteristics like road lengths, road classes, drive times as well as the geographic coordinates of the roads. As the dataset did not contain any information about road gradients, this was retrieved using the Google Maps API in combination with OSMnx, a Python package that enables analyzing and modeling geospatial geometries (Boeing, 2017). The current location of all operational fast-charging stations in Norway was retrieved using

OpenChargeMap's API (OpenChargeMap, 2022). Finally, the data for number of dwellings and holiday homes in each O-D node was retrieved from Statistics Norway's API (Statistics Norway, 2022b).

3.1.1 Selection of O-D Pairs

The selection of O-D nodes is central to the model. A total of 41 nodes were manually selected across Norway, resulting in 1640 O-D pairs (41*40) with directed links across the network. Most of the nodes included are municipalities containing large cities, as this will presumably capture most of the intercity travel. The cities included are chosen to some extent based on the population size, where all the ten most populated cities are included. However, many of the 30 largest cities are placed close to each other in the southern parts. To get a more diversified set of O-D pairs with paths covering larger areas of the country, some less populated cities were included rather than larger alternatives lying close to cities that are already included.

In addition, many Norwegians have holiday homes in the mountains or near the coast, frequently used during weekends and vacations. For this reason, municipalities with the most holiday homes across the country, such as Hemsedal and Kragerø, are included as O-D nodes. The coordinates for each O-D node were found in google maps and stored for later implementation in the network.[1]. The location of the O-D can be viewed in Figure 3.

3.1.2 Gravity Model

A gravity model was used to estimate the flow volumes between each O-D pair. Each node is assigned a weight, which is usually determined by the population within the node(Black, 2003). However, the habits of long-distance travel in Norway are significantly impacted by holiday travelers. Using only the population count in nodes with many holiday homes would likely underestimate the number of people traveling to and from this node. For this reason, it may be reasonable to determine the weight based on the number of registered dwellings and holiday homes within the municipality. In addition, one can argue that a holiday home is likely to influence the flow to and from the node to a higher degree than a dwelling, as they are traveled to from longer distances for holidays and weekend trips. Hence, the number of dwellings and holiday homes are summed, with double weight given to the number of holiday homes. The weight for node i can be formulated as:

$$w_i = Dwellings_i + 2 * HolidayHomes_i \tag{1}$$

[1]All O-D pairs can be found on the Github link in Appendix B

For each O-D pair, the weights of the origin and destination node are multiplied:

$$w_q = w_o * w_d \tag{2}$$

This equation implies that the flow is higher between more populated nodes in terms of dwellings and holiday homes. At the same time, the flow volume for an O-D pair should be reduced in some proportion to the distance between the nodes. The distance is considered by multiplying the path weight with a generalized cost valuec_q. Finding a reasonable cost can be challenging and will depend on the given case. A generic way is to use the square of the distance between the nodes (Black, 2003). However, the standards of the interstate roads across Norway vary greatly, where some of the paths go through steep mountain passes. Using only the geographical distance may overestimate the flow between nodes seemingly close to each other, but the time it takes to drive between them is long. (Kuby & Lim, 2010) use a generalized cost given as:

$$c_q = 0.75 * kilometers + 0.5 * hours \tag{3}$$

Here, it is assumed that the distance is slightly more influential for the flow volumes than the drive time, but whether that is actually the case is not easily proven. One could argue that people are less likely to have reasons to travel places that are farther away, as alternative options somewhere closer are more likely to exist, and the flow may therefore be comparably less influenced by the drive time. Another concern, may be the fact that the units (distance and time) are completely different, and combining them in one variable like this could be viewed as illogical. However, the generalized cost will affect each flow equally relative to each other, and the flow used for the model will be standardized into a percentage of total flow. Thus, we believe this cost function to be sufficient for capturing the general tendencies for long distance travel, despite the fact that more accurate formulations probably exists and should be included with further research.

The flow volume for an O-D pair q can subsequently be calculated as:

$$f_q = w_q / c_q \tag{4}$$

The flow value is standardized to a fraction of the total sum of flows across the network and stored in a list for each O-D pair. It should be noted that the flow volume is an estimate for all vehicle types, and this will not influence the optimization results if the fraction of EVs for the population is the same within each node. Notwithstanding, this is not a reasonable assumption, as the geographic and socioeconomic differences throughout the country are likely to cause the percentages of EVs in different nodes to deviate. This point can be substantiated by looking at the distribution differences for Oslo and Hauge-

sund, which are the largest and 12'th largest cities of Norway respectively. In Oslo, 22% of the vehicle fleet are EVs, as opposed to only 11% in Haugesund (Røed, 2021). While data for identifying the actual distributions of EVs exist for some nodes, we have not been successful in finding extensive data for all nodes of the network and were thus unable to account for the differences in EV distributions.

3.2 Data Preparation

The Road *network for routing* dataset was first imported into QGIS, an open source geographic information system (GIS) used for geospatial data analyses, where a filtering operation was performed. The filtering process consisted of only keeping *functional road class* between 0 and 4, which are the five most important road classes in Norway[2] The dataset was exported into Python using the package *Geopandas* before it was transformed into a network graph by using the Python package *NetworkX*. This network consisted of 205,531 nodes and 217,531 edges. As mentioned in section 3.2, although heuristics makes it possible to solve the FLRM, the runtime of the algorithms is heavily dependent on the network size. With a network size of over 200 000 nodes and edges, none of the heuristics could produce any solutions within a reasonable time frame, even for a small number of CSs to be located[3]. The average path length, i.e., the number of nodes in a path, was 3 120, while the max path length was 9 697. Therefore, the objective function (see chapter 5.1) would spend an immense amount of time calculating whether a given combination of CSs could recharge each path in the network. Consequently, the network graph had to be simplified.

OSMnx(Boeing, 2017) offers a simplification solution where all interstitial nodes, i.e., all nodes that are not intersections nor dead-ends, are removed from the network graph. Unfortunately, this package could not be used, as it caused some of the O-D pairs to be removed. Furthermore, even after this simplification was performed, there were still 25 380 nodes and 74 857 edges remaining, resulting in a seemingly low reduction of the processing runtimes. To ensure that the degree of simplification could be adjusted and tested, the simplification process had to be custom-built. Thus, a simplification function was created, where the distance between each node in the network could be set to a specific value, hereby known as a cut-off value.

The cut-off value splits all paths in the network into intervals where only the first and last node within the interval is kept. For example, a cut-off value of 1km would mean that every path in the network is split into intervals with a length of 1km. All the nodes and edges within this interval are then removed,

[2]This includes state highways, smaller highways, county roads and trunk roads.
[3]The algorithms were run with p=5 and had not finished after 12 hours

before creating a new edge between the first and last node in each interval. This process enabled the creation of several simplified networks with varying path lengths and network complexity. The pseudo-code for this function is presented as algorithm 1.

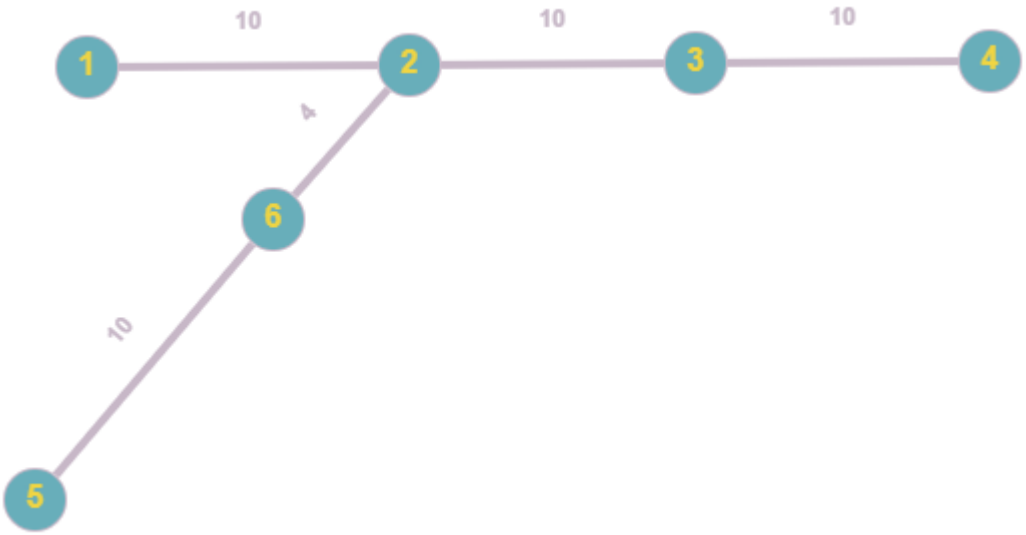
Algorithm 1 Shorten Edges by cut-off

1: **Input:** Unsimplified graph G, cut-off value
2: **Output:** Simplified Graph B by cut-off value
3: Initialize Empty Graph B
4: Initialize Empty list of used nodes $UsedNodes$
5: $AllCSnodes$:= List of nodes in G closest to each unique charging station location
6: Q := Shortest path between each O-D pair in G
7: **for** each $q \in$ Q **do**
8: $CumNodes$:= empty cumulative list of nodes
9: $CumLength$:= empty cumulative list of edge lengths
10: **for** each edge $\in$ q **do**
11: Add length(edge) to $CumLength$
12: Append edge to $CumNodes$
13: **if** $CumNodes[-1]$ is in $UsedNodes$ or $AllCSnodes$ **then**
14: $NewEdge := CumNodes[0], CumNodes[-1]$
15: Add $NewEdge$ to B
16: Empty $CumNodes$ and set $CumLengths$:=0
17: **else if** $CumNodes[-1]$ is in Q **then**
18: $NewEdge := CumNodes[0], CumNodes[-1]$
19: Add $NewEdge$ to B
20: Append $(CumNodes[0], CumNodes[-1])$ to $UsedNodes$
21: Empty $CumNodes$ and set $CumLengths$:=0
22: **else if** $CumLength >$ cut-off Value **then**
23: **if** Distance $CumNodes_{-1} - q_D$ is $<$ cut-off value /2 **then**
24: Pass
25: **else**
26: $NewEdge := CumNodes[0], CumNodes[-1]$
27: Add $NewEdge$ to B
28: Append $(CumNodes[0], CumNodes[-1])$ to $UsedNodes$
29: Empty $CumNodes$ and set $CumLengths$:=0
30:
31: **Return** B

Lines 3 and 4 initialize an empty graph B and an empty list of *UsedNodes*. Line 5 retrieves a list of the nodes in G closest to the actual location of all unique current CS locations. Line 6 retrieves the shortest path between all O-D pairs Q. In lines 7 to 29, each path q is looped through. In lines 8 and 9, the empty lists *CumNodes* and *CumLengths* are initialized. Lines 10-12 loop through each edge in path q, and every edge is appended to *CumNodes* while the length of the edge is added to *CumLengths*. Line 13-16 creates a simplified edge between the first and last node in *CumNodes* if the last node

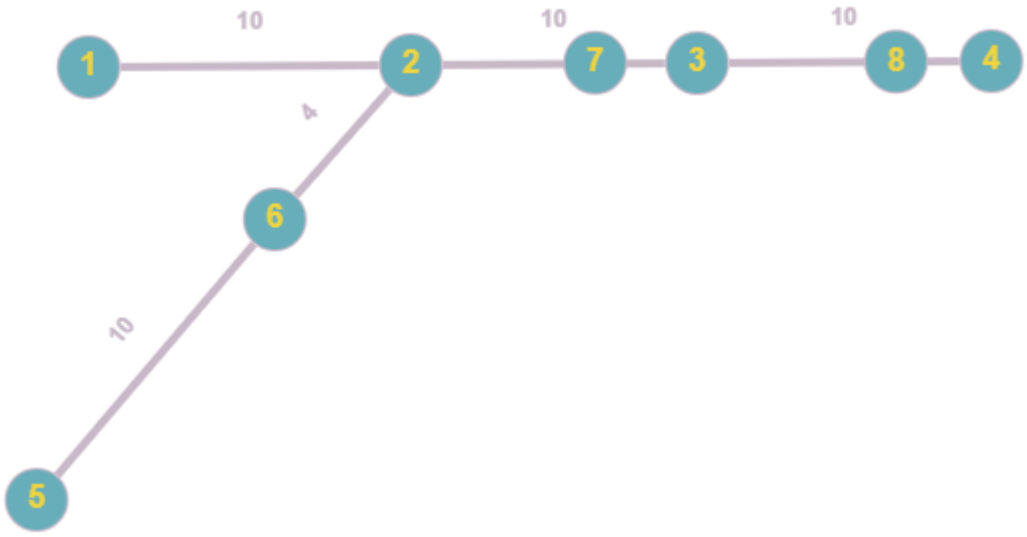
of the last edge added to *CumNodes* is in *UsedNodes* or *AllCSNodes*. If the last node of the last edge added to *CumNodes* is in *UsedNodes*, this means that the current path is entering a previously simplified path, and *UsedNodes* is therefore used to synchronize the paths. Line 16 empties *CumNodes* and *CumLengths*. Lines 17-21 repeat this process if the last node in *CumNodes* is an O-D node. Additionally, line 20 appends the first and last node in *CumNodes* to the list of *UsedNodes*. Lines 22-29 also repeat this process if *CumLengths* exceeds the cut-off value, but only if the distance between the last node n added to *CumNodes* and the destination node q_D is longer than the cut-off value/2. This is to avoid regular nodes being added too close to O-D nodes. If regular nodes were added regardless of the remaining distance to the destination node, the result would be clusters of nodes around the O-D nodes.

To summarize, lines 17-21 ensure that no O-D nodes are removed. Further, lines 22-29 create a simplified edge with an approximate length of the cut-off value specified. Finally, lines 13-16 ensure that current CS locations are kept in the simplified network, as well as ensuring that paths do not overlap. It is vital to emphasize the importance of *UsedNodes*. This list keeps track of all nodes that have been added to the simplified network. Including this list ensures that edges do not overlap when multiple paths run through the same area of the road network. However, it causes some edges to have a shorter length than the cut-off value. To get a better understanding of how *UsedNodes* works, consider the following example:



In this example, there are two O-D paths. The first path is [1,2,3,4] while the second path is [5,6,2,3,4]. Further, a cut-off value of 10 is assumed. It is also assumed that the first path was simplified prior to the second path. In the second path, the first edge [5,6] has a length of 10. However, the second

edge [6,2] only has a length of 4. This is because [2] was added to the list of used nodes during the simplification of the first path. Therefore, for the shared part of the paths [2,3,4], the length between each edge is constant at 10. If an EV range of 15 is assumed, it would be natural to place CSs at nodes [1,2,3,4]. However, if the list of used nodes was not considered, simplification of the second path would result in the following scenario:



Two unnecessary nodes, [7] and [8], are added, as well as edges [6,7] and [7,8]. Thus, the second path would now be [5,6,7,8,4]. Obviously, this counteracts the point of the simplification process. More importantly, in this case, the second path completely ignores [2] and [3]. Thus, these nodes would not be considered as potential CS locations for this path. If the same EV range and CS locations are assumed, EVs traveling on the second path are not able to complete the trip as the vehicle would run out of electricity before when traveling from [6,7]. Extending this to the 820 O-D paths in the road network of Norway, this would ultimately lead to major misallocations of CSs.

An important feature of *Shorten Edges by cut-off* is that it stores the removed nodes and edges and their respective attributes in the simplified network[4]. Doing this allows for a more precise calculation of the grade-adjusted length of each path in the simplified network, as the grade-adjusted road lengths in the simplified network can be calculated using the precision of the unsimplified network.

The gradient of the roads will significantly impact the energy consumption of EVs (Liu et al., 2017), which can be defined as *how many units traveled vertically for each unit traveled horizontally*. For example, a road gradient of 5% stipulates that for every meter traveled horizontally, there is a vertical gain of

[4]This is done by calling the function *GetEdgeAttributes* in line 17. For the complete codes, please use the authors' Github link in Appendix A, where all functions are included.

0.05 meters. Considering the mountainous geography of Norway, disregarding the road gradient could potentially result in significant electric consumption estimation errors. Using the OSMnx package in conjunction with the Google Maps API, node elevations for each node was retrieved. Next, each edge gradient was calculated using the node elevation and edge direction. (Liu et al., 2017) investigate how road gradient impacts the energy consumption of electric vehicles by combining the use of GPS and digital elevation mapping for 492 electric vehicles in Japan. Table 2 shows their proposed model for estimating how road gradient affects electricity consumption for vehicles:

Ind. Variable (Gradient Intervals)	Coefficient (kWh per Km)
β_0	0.372
$<$ –9%	-0.332
[–9%, –7%)	-0.217
[–7%, –5%)	-0.148
[–5%, –3%)	-0.121
[–3%, –1%)	-0.073
[1%, 3%)	0.085
[3%, 5%)	0.152
[5%, 7%)	0.203
[7%, 9%)	0.306
[9%, 11%)	0.358
$>$ 11%	0.552

Table 1: How Road Gradient Affects Electricity Consumption

The constant variable β_0 represents how many kWh an electric vehicle on average consumes when traveling one kilometer on a road with 0% gradient. In contrast, the rest of the variables explain how much the electricity consumption changes depending on the road's gradient. In this thesis, battery capacity (kWh) has been translated into vehicle range. Thus, instead of using the change in kWh per kilometer, we consider how the *effective length* changes based on the road gradient. The effective length can be defined as the adjusted length when accounting for road gradient, or more formally:

$$effective\ length = length_i * \frac{\Delta consumption}{\beta_0},\ \forall i \tag{5}$$

$$\Delta conumption = \beta_0 * grade\ interval\ consumption_j, \tag{6}$$

Using equation (5) and (6), we are able to translate (Liu et al., 2017) findings into lengths instead of EV kWh consumption. From the equation, it can be seen that the effective length decreases when the grade interval consumption is negative and vice versa. This reflects that negative road gradients reduce electricity consumption while positive road gradients increase the electricity consumption. The equation is applied twice to each edge in the network since both directions must be considered. First, equation (5) and (6) is used to calculate the effective length from $edge_i$ to $edge_j$, using the stored gradients for all previously removed edges. For $edge_j$ to $edge_i$, the sign of each gradient is reversed before using the equation. Both values are then stored, allowing for a simple retrieval of the effective length for both directions of each edge. Consequently, the graph becomes directed, which will have implications for the optimization models that we implemented in the next sections.

The final network, consisting of 1 583 nodes and 1 1802 edges, is presented in Figure 3. The figure also shows the location of all 41 O-D nodes, marked in red.

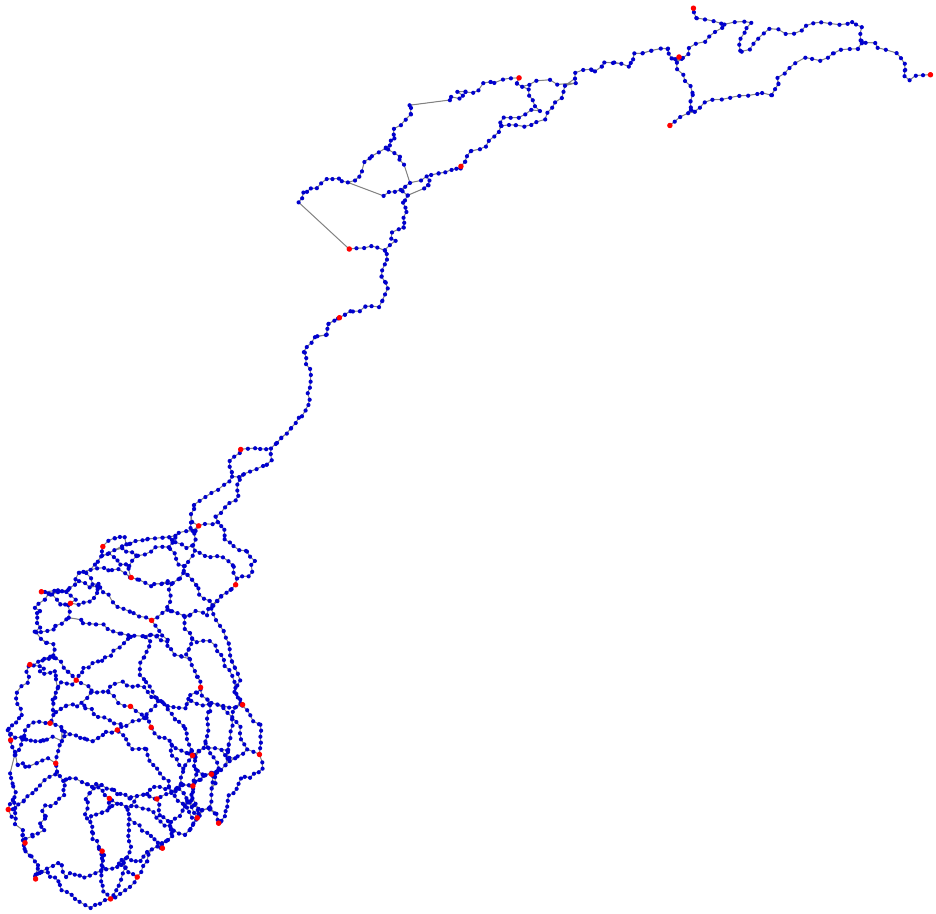


Figure 3: Road Network of Norway

As illustrated in Figure 3, there is, for the most part, a systematic distance between each node in the network. This is due to the simplification process performed, ensuring that most edges have a length of approximately 10km[5]. However, since O-D nodes and current CS locations are kept in the network, areas around these locations are generally highly populated by surrounding nodes. This is visible in the figure, especially around the Oslo area. Consequently, this inevitably affects the average edge length, which Table 3 reports to be 8.315km. Thus, there is a slightly imbalanced node distribution, where areas with many already existing CSs receive extra nodes. However, this does not significantly affect the results as it only leads to a potentially more accurate location of CS allocation in these areas.

Statistics	Simplified Network (10Km)	Unsimplified Network
Nodes	1 583	205 531
Edges	1 918	217 581
Density	0.001532	0.000010
Average Degree	2.4232	2.1173
Max Degree	7	5
Avg Path Length	91.2756	3 119.9463
Max Path Length	297	9 697
Avg Dist. Between Each Node	8 314.50	180.50

Table 2: Network Statistics Simplified vs Unsimplified Network

Table 3 also illustrates the trade-off between precision and runtime. While the nodes in the unsimplified network, on average, only have a distance of 180 meters between them, the average path length is over 3000 nodes. Using this network, one could locate the optimal set of CS with high accuracy. However, the time it would take to find the optimal solution, even for a relatively small set of CS, makes this network impractical, if not infeasible. In the simplified network, the average path length is approximately 91 nodes, drastically reducing the runtime compared to the unsimplified network. However, the average distance between each node is close to 8km, imposing a restriction in terms of accuracy. Nonetheless, as only inter-city/long-distance traveling is considered in this thesis, trading away some precision for a significantly reduced computational burden can be considered fair.

[5]The cut-off value used was 10km. The reason for this is explained in section 5.1

4 Optimization Models

This section presents the optimization process performed to locate CSs across the Norwegian road network. To solve the location problem in the Norwegian context, this thesis will apply the solution approaches proposed by (Kuby & Lim, 2010). First, the FLRM and its challenges is presented. Next, the logic behind the *objective function* is explained. Finally, the implementation of the heuristics algorithms is explained in depth, as well as some important considerations and assumptions.

4.1 The FRLM

To find the optimal locations for CS across the Norwegian road network, we will use the FRLM model, as proposed by (Kuby & Lim, 2010). Although heuristics is well-suited to solve the model, an explanation of the MILP formulation of the model may provide a better understanding of the underlying concepts and challenges.

4.1.1 MILP formulation

The FRLM can be formulated as the following *mixed integer linear program* (MILP):

Notations:
Q = Set of all O-D pairs
H = Set of all CS combinations
K = Set of all potential CS locations
q = index of O-D pairs
h = index of CS combinations
k = potential CS location
f_q = flow volume for shortest path between O-D pair q
$b_{qh} = 1$ if CS combination h can recharge the shortest path between O-D pair q, 0 otherwise
$a_{hk} = 1$ if CS k is in combination h, 0 otherwise
p = Number of CSs to be located
Variables:
$x_k = 1$ if CS k is used, 0 otherwise
$y_q = 1$ if the flow on the shortest path between O-D pair q is captured, 0 otherwise
$v_h = 1$ if the CSs in combination h are used, 0 otherwise

$$\max \quad Z = \sum_{q \in Q} f_q y_q \tag{1}$$

$$\text{subject to:} \quad \sum_{h \in H} b_{qh} v_h \geq y_q \quad \forall q \in Q \tag{2}$$

$$x_k \geq a_{hk} v_h \quad \forall k \in K \tag{3}$$

$$\sum_{k \in K} x_k = p \tag{4}$$

$$x_k \in \{0, 1\} \quad \forall k, h, q \tag{5}$$

$$0 \leq v_h \leq 1 \quad \forall h \tag{6}$$

$$0 \leq y_q \leq 1 \quad \forall q \tag{7}$$

The objective function (1) maximizes the sum of flows f_q that is captured in the network, where the flow through each O-D pair q is considered captured if at least one eligible CS combination h is initialized to recharge the path (2). Constraint (3) will set v_h to 1 if all the CS in combination h are open, and (4) requires p number of CSs to be initialized. This means that the model is solved for an exogenously determined number of CSs. From constraint (5) it follows that all the variables are binary.

A set of O-D pairs Q must be selected across the network for the model. The list of potential CS locations K will have all the O-D nodes, but can also include additional nodes. The model also needs an estimate of the flow volume associated with each O-D pair.

To deploy the model, a set H with all possible combinations of CS locations that can recharge the paths of the network must be identified. Figure 3 shows an O-D path going from the origin node A, through B and C, to the destination node D. This path will help illustrate the logic behind which combinations are counted as eligible and included in H, where an EV range of 16 is assumed.

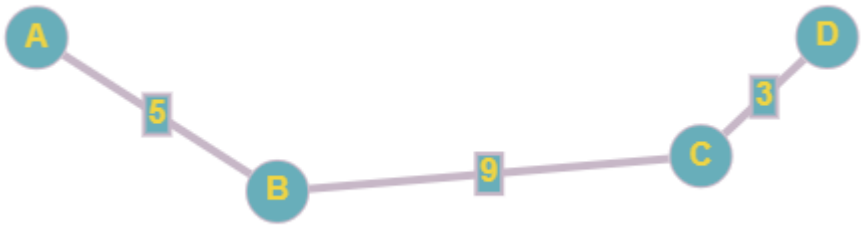


Figure 4: Exemplified O-D path

Before knowing the combinations of locations that can recharge the path, an assumption must be made about the battery's state of charge [6] before beginning the trip from the origin node. If a CS is placed in A, the EV can charge before leaving, and the range should reflect a fully charged battery. If there

[6] State of charge (SoC) is the battery lever for an EV, ranging from 0-100%.

is no station in A, some EVs may not have a fully charged battery when beginning the path. Thus, an average EV is assumed start with half of the total range if no CS is initialized in the origin node.

With this assumption, there must be a CS in B if A does not have one, as C is out of range with half a battery charge (8 in this example). If there is a CS in A, the vehicle can reach C without a CS in B. A CS in B would allow the vehicle to travel to the destination node and back to C, but it would then be unable to return to B before running out of electricity. This exemplifies why the round-trip on the path should be considered. With a CS in B, either C or D must have a CS to complete the round-trip. A possible combination would also be to initialize in both C and D together with B. Nonetheless, this would be a superset of another valid combination. Because only one CS is required to recharge the path in conjunction with B, it is not necessary to include the bigger combination in the MILP model.
In this example, the possible combinations (excluding combinations that are supersets of other combinations) of stations that can recharge the path would be [A, C],[B, C] or [B, D]. These combinations are included in H, and the process is repeated for all O-D paths in the network.

4.1.2 Challenges

A flow-based demand may be a realistic approach for optimizing CS allocation, but several challenges must be considered. Solving the MILP formulation requires an exogenously pre-generated set of all possible combinations of CSs that can recharge each path (H). While this works for small-sized networks, generating all the combinations will be highly computationally expensive, if not infeasible, for most real-world problems considering long-distance travel. It may be particularly challenging if many potential facility nodes are included midway along paths, between intersections in the network. If the nodes included in Figure 3 represent intersections, additional nodes could be added midway between them to increase the precision of the optimized solutions. However, as this would further increase the problem's complexity, heuristic algorithms are often required to solve the problem.

In addition, the trade-off between the solution quality and runtime must be considered. More extensive networks generally contain more nodes along each path, increasing the number of possible combinations of CSs that can recharge a path. While this gives the model flexibility to produce better solutions, the runtime is expected to increase drastically as the network expands. Conversely, less complex networks may not include enough potential station locations, thus concealing the optimal locations.

Moreover, road gradients should be accounted for when using the model on networks with considerable elevations. The nature of battery consummation

for different grades displayed in table 2 implies that the consumption of a path will differ depending on the direction being traveled. Consequently, Traveling the path sequence O-D-O will not be the same as D-O-D, which may result in a separate set of CSs being eligible to recharge a depending on the direction. Consequently, the model cannot handle each O-D pair as an undirected path with a single subscript q, and the optimizing algorithm must calculate the validity of a combination of CSs twice (one time for each direction), drastically increasing the processing time to solve the model. To fully implement the feature of undirected graphs, the flow volume should also be differentiated depending on the direction. Two different flows for each path would allow one direction to be recharged, and the flow counted as captured, even though the opposite direction cannot be recharged. Nonetheless, this is not easily implemented in the sterile gravity model deployed for this thesis and is left for potential further research. For this implementation, the flow on a path will only be considered captured if a set of CSs can successfully recharge the flows in both directions.

Moreover, lack of data is generally a significant obstacle when conducting refueling/charging allocation experiments (Pagany et al., 2019). As no data shows the habits for people long-distance traveling, defining the number and location of O-D nodes can be a considerate challenge. Specific qualitative considerations must be made to determine the reasonable origin and destination nodes in the network, which is difficult to confirm and validate. The biggest challenge for the FLRM is the estimation of traffic flows between each O-D pair, as the data required to do so generally does not exist. Due to the lack of data, preceding research on this topic has often applied a gravity model to obtain an estimated flow (Kuby & Lim, 2005; Capar & Kuby, 2012). The only publicly available data in Norway is traffic flow counts registered by traffic points. While such data may be used to calculate traffic flows for intra-city travel, it can not effectively be used for inter-city travel, as it gives no information about each vehicle's origin or destination.

4.2 Implementation of Algorithms

In the algorithms, K refers to the full set of potential CS locations, which are the 1583 nodes of the network referred to as G. A certain set of CS locations is noted K with an upper subscript, for instance K^t, and k is referring to a singular CS node. Q is the full set of O-D pairs in the network, and q refers to one such pair. To be precise, each q represents a list of potential CS nodes on the shortest path between the origin and destination node.

All the information needed for the following algorithms is stored in the graph that represents the Norwegian road network. This includes which nodes that have been selected as O-D nodes, and the flow volumes between them.

4.2.1 Calculating objective value

The optimizing algorithms compare different solutions with an objective function, which in this case calculates the flow volume that is captured by initializing a given set of CS. To get the objective value, an algorithm for evaluating whether initializing a set of stations would be able to recharge an O-D path is required. Algorithms 2 and 3 are designed to operate together to find the objective value in the heuristic algorithms, so that only the O-D pairs with a CS anywhere on the path are evaluated. This approach is considerably less computationally expensive than making a complete set of all potential CS combinations that can recharge each path, the way it would be in a MILP model.

Algorithm 2 Evaluate set of CSs for recharging path

1: **Caption:** *EvaluateCSCombination*(K^t, q, r, G)
2: **Input:** A set of CS (K^t), an O-D path (q), EV range (r), a graph G
3: **Output:** $a = 1$ if K^t can recharge path q, 0 otherwise
4: q_f := Entire path for a round-trip between origin and destination in q
5: k_{origin} := The origin node in q
6: *RemainingRange* := $r/2$
7: **if** k_{origin} is $\in K^t$ **then**
8: *RemainingRange* = r
9: **for** each k on the path q_f **do**
10: k_{+1} := The next node on the path q_f
11: $d_{k,k_{+1}}$:= Effective length between k and k_{+1} in G
12: Subtract d_n from *RemainingRange*
13: **if** *RemainingRange* < 0 **then**
14: $a = 0$
15: **break** loop
16: **if** k_{+1} is $\in K^t$ **then**
17: *RemainingRange* = r
18: **continue** loop
19: **if** $k_{+1} = k_{origin}$ **then**
20: $a = 1$
21: **break** loop
22: **return** a

Algorithm 2 takes a list of CS, K^t, a list of nodes in the network on the shortest path between an O-D pair, q, and a constant for the EV range r. In addition, it needs a graph G to access the distance between any two nodes in the network. The output is a binary variable, a, which equals 1 if the set of CS is able to recharge the path and 0 otherwise.

As discussed in section 3, we have to consider round trip travel between the origin and destination node. Line 4 creates a round trip path, q_f. If for example a path goes from node A, through B to the destination node C, q_f will be a vector [A - B - C - B - A], resembling a round trip between A and

C. Line 5 saves the origin node for the path a separate variable. Lines 6 to 8 initialize a variable for the remaining range of the vehicle, *RemainingRange*, which will be used as the indicator of whether the EV is capable of driving a given distance. If the origin node has a charging station, the vehicle can recharge before beginning its travel and the initial remaining range value is set to equal the full EV range. Otherwise, the initial range is assumed to be half of the EV range.

From line 9 to 21 is a loop, where each iteration resembles an EV traveling from one node k to next k_{+1} on the shortest path from origin to destination and back to origin. Each node represents a potential CS location. Traveling from one node to the next, the remaining range variable is reduced by the distance between the nodes, represented by lines 10 to 12. The distance is the effective length between the nodes where the gradient is accounted for.

If the remaining range becomes negative after the inter-node travel, it means the vehicle has run out of power before reaching the next CS, and the given set of K^t CS will not be able to recharge the path. Lines 13 to 15 check if the remaining range has become negative and will set a = 0 and breaks out of the loop to stop the algorithm if that is the case. If not, lines 16 to 18 are executed when the node traveled to is in the given set of CS, K^t. The vehicle can now fully recharge its batteries and the remaining range is set to equal the full EV range. Lines 19 to 21 are executed if the node the vehicle traveled is the origin node. This means the vehicle has successfully been able to travel the full round trip, and a is set equal 1 before stopping the algorithm. The loop continues until the first or third if-statement above is executed, stopping the algorithm and returning the indicator variable a.

Algorithm 3 Objective function

1: **Caption:** *ObjectiveFunction(*G, K^t, r*)*
2: **Input:** A graph (G), a set of CS (K^t), EV range (r)
3: **Output:** Captured flow by initializing K^t
4: Q := Set of paths for each O-D pair in G
5: F := List of flow volumes through each $q \in Q$
6: *CapturedFlow* := 0
7: **for** each $q \in$ Q **do**
8: **if** q effective length $\cdot$ 2 <= $r/2$ **then**
9: Add $f_q \in F$ to *capturedFlow*
10: **continue** to next loop iteration
11: **if** any $k \in K^t$ is in q **then**
12: qr := The reverse order of nodes in path q
13: a_q = *EvaluateCSCombination(*K^t, q, r, G*)*
14: a_{qr} = *EvaluateCSCombination(*K^t, qr, r, G*)*
15: **if** $a_q + a_{qr} = 2$ **then**
16: Add $f_q \in F$ to *capturedFlow*
17: **return** *capturedFlow*

Algorithm 3 is the objective function, which evaluates the flow volume captured by a given set of CS using algorithm 2 on each O-D path in the network. The input is the list of CS to be evaluated, K^t, a graph G and an EV range r, and the output is a value of the flow captured in the network. Line 4 stores the list of nodes on the shortest path between each O-D pair in a data frame, Q, and F is a list of the respective flow volumes. Line 6 initializes a variable for storing the flows that is capture for each path.

Lines 7 to 14 loop through each path q to identify which can be recharged by the set of CSs given as input. For the longer EV ranges, some O-D pairs will be short enough for a flow to be captured without any CS on the path. Lines 8 to 10 immediately add the flow volume through q if the distance for a round-trip on the path is shorter than half the EV range[7]. If the path can be recharged without CS on the path, the loop can immediately go to the next loop iteration. If that is not the case, line 11 ensures that algorithm 2 is only executed for paths with at least one CS in K^t along the path. If there is a CS in the path, lines 12 to 14 are executed. Because the paths are directed from the inclusion of road gradients, traveling a path form origin (O) to the destination (D) and back to O, will not result in the exact same battery consumption as traveling from D to O and back to D. Thus, line 12 makes a path of the nodes in q in the reverse order, qr. Lines 13 and 14 executes algorithm 2 for both q and qr, and returns two indicator variables a_q and $a_q r$. If the set of CS, Kt, is able to recharge the path for both directions, $a_q + a_{qr}$ will equal two and line 16 adds the flow volume to the total captured flow.

4.2.2 Greedy Algorithms

To solve the FRLM on the Norwegian road network, we aimed to use three heuristic approaches: Greedy-adding, Greedy-adding with substitution and a genetic algorithm. For the greedy algorithms, the number of CS to be deployed, p, and the number of substitution iterations[8], s, must be decided.

[7]If no CS is placed in origin, we assume a starting EV range of half the full range.

[8]Setting the number of substitutions=0 corresponds to greedy-adding, while values above this corresponds to greedy-adding with substitution

Algorithm 4 Greedy-adding/ Greedy-adding with substitution

1: **Input:** A graph G, EV range (r), number of CS (p), sub-iterations (s)
2: **Output:** A list of CS K^o
3: K := Set of potential CS locations in G
4: K^o := Empty list
5: **for** CS= $1, 2, \ldots, p$ **do**
6: $ObjectiveValue_{max} := 0$
7: $k_{max} := 0$
8: **for** each $k \in K \notin K^o$ **do**
9: K^{temp} := copy of K^o
10: Append k to K^{temp}
11: $ObjectiveValue_k = ObjectiveFunction(G, K^{temp}, r)$
12: **if** $ObjectiveValue_k > ObjectiveValue_{max}$ **then**
13: $ObjectiveValue_{max} := ObjectiveValue_k$
14: $k_{max} := k$
15: Append k_{max} to K^o
16: **for** Sub-iteration= $1, 2, \ldots, s$ **do**
17: $SubObjectiveValue_{max} := 0$
18: $Subk_{max} := 0$
19: $k_{sub} := 0$
20: **for** each $k \in K^o$ **do**
21: **if** $k \neq k_{max}$ **then**
22: K^{temp} := K^o with k removed
23: **for** each $k_i \in K \notin K^o$ **do**
24: K^{sub} := K^{temp} with k_i appended
25: $ObjectiveValue_{ki} = ObjectiveFunction(G,\ K^{sub},\ r)$
26: **if** $ObjectiveValue_{ki} > SubObjectiveValue_{max}$ **then**
27: $SubObjectiveValue_{max} := ObjectiveValue_{ki}$
28: $Subk_{max} := k_i$
29: $k_{sub} := k$
30: **if** $SubObjectiveValue_{max} > ObjectiveValue_{max}$ **then**
31: Remove k_{sub} from K^o
32: Append $Subk_{max}$ to K^o
33: $ObjectiveValue_{max} := SubObjectiveValue_{max}$
34: $k_{max} := Subk_{max}$
35: **if** $SubObjectiveValue_{max} < ObjectiveValue_{max}$ **then**
36: **break** substitution loop
37: **return** K^o

Algorithm 4 displays the logic of how the greedy-adding algorithms are implemented for this problem. Line 3 creates a set of all the potential CS nodes in the network. Line 4 initializes an empty list where the CS capturing the most flow will be added with each iteration of lines 5 to 36. Line 6 and 7 define two empty variables that will contain information on the current max objective value and the associated CS. Lines 8 to 15 iterate through each CS node in the network not already included in k^o, and calculate the objective value associated with adding each CS to K^o in isolation. The station that

generates the highest flow volume will be stored in k_{max} and appended to K^o in line 15. This part constitute the greedy-adding algorithm, and is the only part that will be executed if the the number of substitution iterations s is set to 0.

Lines 16 to 36 sequentially substitute out stations in k^o to see whether a better solution is available after adding each new station. The number of substitution iterations depends on the user input s. Lines 17 to 19 initialize variables that will be updated with the information from the best solutions provided by the substitutions. $Subk_{max}$ is the CS that is substituted into k^o to provide the $SubObjectiveValue_{max}$, and k_{sub} is the CS being replaced. Lines 20 to 29 sequentially substitute each CS in k^o, except for the one last added (k_{max}), with every CS in K that is not already in k^o. The substitution that results in the highest objective value is then compared to the current max objective value, and if it is higher, lines 30 to 34 will perform the substitution and update the max objective value. If no substitution produced a higher objective value, line 35 and 36 stop the substitution loop, as the next iteration would produce the same result. When all p CS has been iterated through, line 37 returns the list of flow optimizing CS, K^o, as output.

4.2.3 Genetic Algorithm

The GA is inspired by the evolutionary theory of natural selection from mixing gene pools by making offspring and random mutations in the genes of new generations. In this implementation, a *gene* is referring to a certain CS location k. A *chromosome* is made up of multiple genes, meaning it will represent a set of CSs. The size of each chromosome (how many genes it consists of), will be determined by the number of CSs, p, we want to optimize. The fitness value of a chromosome refers to the flow volume it captures in the network, and the population is the entire collection of chromosomes. Figure 5 illustrates a population with two chromosomes (sets of CSs) with five genes in each (p=5). The total gene pool consists of CSs 1-9.

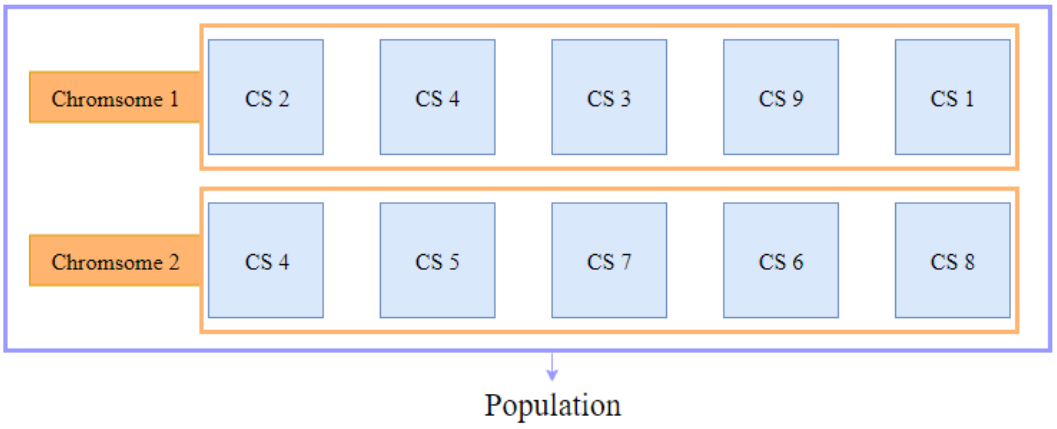


Figure 5: Relationship between genes, chromosomes and population

The algorithm works by first making a random set of chromosomes to constitute an initial population (IP). Figure 5 shows an IP with a size of two chromosomes. This will subsequently be split into a *mother* and *father* sub-population. A new sub-population of *offspring* is then made by pairing the chromosomes in *mother* and *father*. The genes in the first chromosome in *mother* will be merged with the genes in *faters*'s first chromosome, and so on. This is called a crossover of genes is visualized in Figure 6.

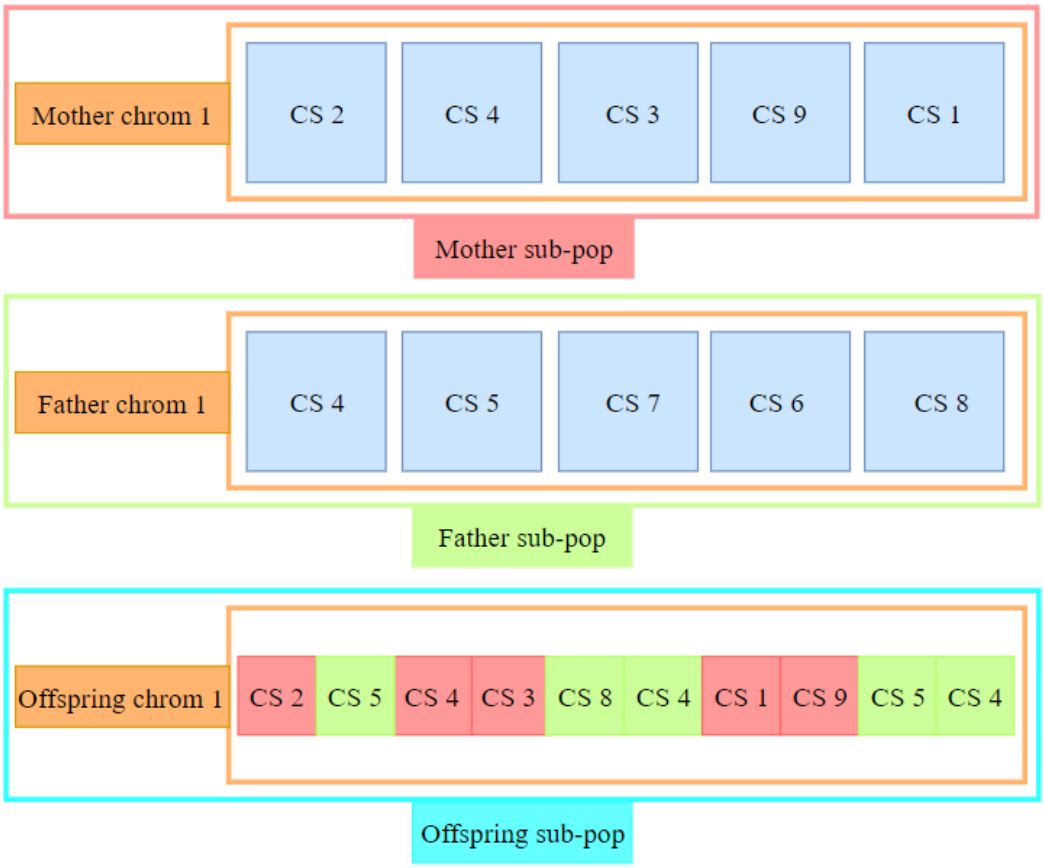


Figure 6: Gene crossover between *mother* and *father*

The *offspring* sub-population now has $p * 2 = 10$ genes in each chromosome, p from *mother* and p from *father*. The red and green colors have been applied to the genes in the offspring chromosome to visualize which come from *mother* and which is taken from *father*. To reduce number of genes in each chromosome, p genes can either be picked at randomly, or by identifying and keeping the genes with the highest fitness value. We have implemented an algorithm for greedy deletion of genes, which iteratively removes genes that results in the least reduction in objective value. In the exemplified offspring chromosome in Figure 6, the objective value is calculated for all combinations of nine genes out of the total ten. These are compared, and the genes associated with the highest value will be kept. This process is repeated until five genes remain. Such a greedy deletion of the crossover genes will help the GA converge quicker, but reduce the diversity of genes in the populations as the generations evolve.

The next phase conducts a the new member selection, where the fitness value of each corresponding *mother*, *father* and *offspring* chromosome is compared. In the example from Figure 6, the objective value for the three chromosomes displayed will be compared and the two fittest chromosomes are included in the population to represent the next generation. The same is done for each corresponding chromosome index across the three sub-populations. This method, as opposed to ranking all the chromosomes and choosing only the fittest, will result in a higher diversity of genes, and reduce the risk of getting stuck in local optimums.

Lastly, random mutations are inflicted upon some of the chromosomes, where genes are substituted with a randomly selected new gene. The algorithm iterates through multiple generations to let the populations evolve and produce better solutions.

To initiate the algorithm, the number of generations, mutation frequency and a population limit must be specified. These values will depend on the problem at hand and certain qualitative assessments must be made to balance the processing time and the quality of the solution. The number of generations, gen, determines how many generations of offspring will be produced, in other words how many iterations it will run before terminating and returning the chromosome capturing the most flow. This depends on how quickly the best chromosomes for each generations converge towards a certain value and do not continue to improve. The population limit, PL, determines both the initial and maximum number of chromosomes in each generation. A large PL will increase the probability of a better solution, as a more diverse gene pool of unique CS locations decreases the likelihood of missing a station in the optimal solution. However, large populations drastically increase the processing time. The mutation frequency, mf, determines the rate at which mutations occur in new generations. When a gene within a chromosome mutates, it will be substituted by a new CS k from the full set of potential locations. This is included to help escaping local optimums, but setting the frequency too high will make the algorithm slower to converge and possibly produce worse solutions as it has a higher chance of changing out CS in too many of the effective chromosomes. The pseudo code for this algorithm can be found below.

Algorithm 5 Genetic algorithm

1: **Input:** Graph G, EV range (r), number of CS (p), number of generations (gen), population limit (PL), mutation frequency (MF)
2: **Output:** A list of CS K^o
3: K := List of all potential CS locations in G
4: P := Population of chromosomes with p genes each from K
5: **for** generation= $1, 2..., gen$ **do**
6: $M := PL/2$ random chromosomes from P
7: $F := P - M$
8: C := Union chromosomes in M and F
9: **for** each $K^C \in C$ **do**
10: **for** gene= $1, 2..., p$ **do**
11: $ObjectiveValue_{max} := 0$
12: $k_{max} := 0$
13: **for** each $k \in K^c$ **do**
14: $K^{temp} := K^c$ with k removed
15: $ObjectiveValue_k$ = *Objective function(G,* K^{temp}*, r)*
16: **if** $ObjectiveValue_k > ObjectiveValue_{max}$ **then**
17: $ObjectiveValue_{max} := ObjectiveValue_k$
18: $k_{max} := k$
19: Remove k_{max} from K^c
20: $P := M + F + C$
21: **for** i= $1, 2..., (PL/2)$ **do**
22: $Value_{M,i}$ =*Objective function(G,* $K^{M,i}$*, r)*
23: $Value_{F,i}$ =*Objective function(G,* $K^{F,i}$*, r)*
24: $Value_{C,i}$ =*Objective function(G,* $K^{C,i}$*, r)*
25: $Value_{min} = \min(Value_{M,i}, Value_{F,i}, Value_{C,i})$
26: **if** $Value_{min} == Value_{M,i}$ **then**
27: Remove $K^{M,i}$ from P
28: **if** $Value_{min} == Value_{F,i}$ **then**
29: Remove $K^{F,i}$ from P
30: **if** $Value_{min} == Value_{C,i}$ **then**
31: Remove $K^{C,i}$ from P
32: **if** P is larger than PL **then**
33: Keep the PL fittest chromosomes in P
34: **for** each $K^p \in P$ **do**
35: Replace $k \in K^p$ with a random $k \in K$ with probability MF
36: $ObjectiveValue_{max} := 0$
37: K^o := Empty list
38: **for** each $K^p \in P$ **do**
39: $ObjectiveValue_{k^p}$ = *Objective function(G,* K^p*, r)*
40: **if** $ObjectiveValue_{k^p} > ObjectiveValue_{max}$ **then**
41: $ObjectiveValue_{max} := ObjectiveValue_{k^p}$
42: $K^o := K^p$
43: **return** K^o

Line 3 creates a list of all potential potential CS locations included in the network G. Line 4 generates a set of chromosomes to represent the initial population. The size of the initial population is often set to equal the population

limit, but can also be of a different size. To ensure that all the unique genes are included for the first generation, the chromosomes are made sequentially using p genes that has not been included in any of the other chromosomes until at least one instance of each is included.

Lines 5 to 35 iterate through the generations of the genetic algorithm. Line 6 and 7 splits the population into two sub-populations of mother and father chromosomes. Line 8 initializes a new sub-population of offspring by performing a crossover of genes in each chromosome from the mother and father sub-population, using the logic displayed in Figure 6. Lines 9 to 19 is a greedy deletion algorithm for choosing which genes to keep in each chromosome of the offspring sub-population. This is done by sequentially removing one gene k from each chromosome K^c and identifying which set of remaining genes will result in the highest objective value.

Line 20 sets the total population P to equalize the sum of chromosomes across the three sub-populations. As a result, the population will now have more chromosomes in the population than the population limit allows. Lines 21 to 31 iterate through each chromosome in each sub-population to remove the least fit chromosome so that the population limit is satisfied. In detail, lines 22, 23 and 24 calculate the objective value for each chromosome in position i in the mother (K^M, i), father (K^F, i) and offspring (K^C, i) sub-populations. Then, Lines 25 to 31 identify the chromosome associated with the lowest value and removes it from the population.

For the first generation, the population size could be larger than the population limit, depending on the initial population. If that is the case, line 32 and 33 picks the chromosomes with the highest objective vale, so that the population size can be reduced in accordance with the set limit.

Lines 34 and 35 iterate through each chromosome K^p in the population and mutates one gene with a probability equal to the mutation frequency given as input. Line 32 ensures that no mutations occur in the last generation of the algorithm, so that the best solution of the last generation is kept.

Lines 36 to 42 are executed after all the generations have been iterated, and identifies the chromosome with the highest objective value in the last generation. This set of CS, K^o, is returned as the output when the algorithm terminates.

5 Performance and Results

This section starts by presenting the current CS system in Norway, as well as how it is modeled in the network, taking the steps performed in section 4.2 into consideration. Next, the performance of the heuristics algorithms is evaluated for both a 25-node test network and the Norwegian road network. Finally, the results produced by the heuristics algorithms are presented.

5.1 Current System

In section 3.2, the concept of using a cut-off value to simplify the network graph was discussed. Determining a suitable cut-off value is important, as too high values limit the number of potential CS locations, which further decreases the representativeness of the actual system. Conversely, a low cut-off value gives rise to a more complex network, drastically increasing the computational processing requirements. Previous research on applying heuristics to solve the FLRM shows that changing p from 5 to 10 increases the genetic algorithm's runtime by up to 6 times, while setting p=25 may cause the algorithm to run for 60 times as long (Kuby & Lim, 2010). This is an important consideration as the GA should be run multiple times due to its random nature. Table 1 shows the runtimes reported in seconds for all cut-off values tested. All heuristics algorithms were executed with p=5 and an EV range of 325km. The GA was run with a population limit of 500, mutation frequency of 0.1, and 15 generations. The greedy-adding with substitutions was run with one sub iteration. It should be noted that we let the algorithms run for a maximum of 6 hours when testing the cut-off values. Hence, runs that did not finish within 6 hours are marked as 21 600*.

Statistics			Runtime (in Seconds)		
Cutoff Value(km)	Nodes	Edges	Greedy-Adding	Greedy-Adding w/1 sub	Genetic Algorithm
0	205 531	217 581	7 937	21 600*	21 600*
10	10 535	11 812	407	14 933	4 809
20	6 084	6 878	196	9 544	3 088
50	2 828	3 286	96	8 671	2 853
100	1 583	1 1918	60	372	500

Table 3: Runtimes for Various Cutoff Values

Considering both time-efficiency and the representational potential of the network, a cut-off value of 10km meters deemed sufficient. At this cut-off value, the heuristics algorithms were able to produce solutions for a low p, while keeping the runtimes relatively low. Examining a cut-off value of 5km, the runtime of the GAAL with substitution and the GA is significantly higher. Thus, trading away 5km of precision on average, for a drastic reduction in runtime can be considered a reasonable trade.

It should be mentioned that for a cut-off value of 10km, *Shorten Edges by cut-off* was unable to capture all current CSs along each O-D path. Therefore, after the network graph had been simplified, a new list of *RemaningCSNodes* was created, which contained the coordinates of each CS that had not been placed in the network. For each element in this list, the closest node in the simplified network was found. If the distance between the node in the network and the CS in the list was less than 10km, the CS would be added to the network. Naturally, this caused a slightly higher average deviation between the

estimated CS locations and their actual location, but as discussed in section 6.1, this does not significantly affect the outcome of this thesis.

According to OpenChargeMap's API, a non-profit open source collection of worldwide EV charging stations, there exist 582 publicly available, operational fast charging stations in unique locations in Norway, as of spring 2022. Out of these, only 330 can be placed in a unique location in the simplified network. The reason for this lies in the logic of the model. Since the FLRM only considers the flow between O-D pairs, roads that are not used when travelling the paths between each O-D pair are not considered. This is because this thesis assumes that there is no deviation from the shortest path. This logic can also be applied to CSs across the country. If a CS is located on a road that is not used by any of the O-D paths, including this CS would serve no practical purpose as no vehicles will travel on this path. Therefore, only relevant CS locations are included in the current network system. The comparison between the actual system and the network system is represented in Figure 4 below.

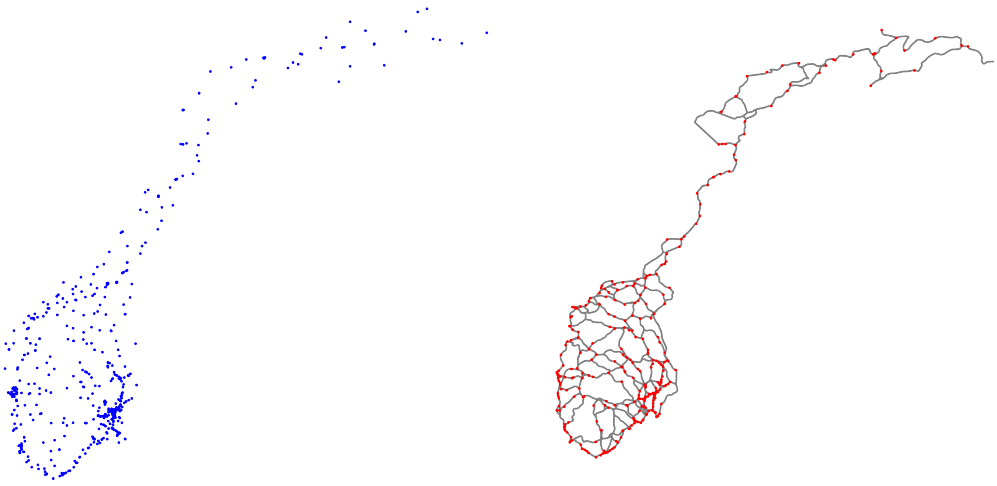


Figure 7: CS Locations in current system (left) vs current network system (right)

To further clarify why the current network system does not contain all 582 available CSs, Figure 5 below aims to visualize why this is the case. In this figure, the red nodes represent the CS locations in the current network system, while the blue nodes reflect the CSs that are not included in the current network system. Examining the figure, a significant amount of the blue nodes are not located inside the road network. These CSs can therefore not be used in the current network system, as their location lie outside all of the shortest O-D paths in the network. However, in some parts of the network, especially in the Oslo-area, some of the blue nodes seem to be placed on the paths. In many of these instances, the nodes are in fact located slightly outside of the path and are thus not included in the network. Even if some of the blue nodes do lie on the path, this is not considered an issue as these instances only occur

in areas where there in general exist clusters of CSs, and it would therefore not affect how the current situation is modelled significantly.

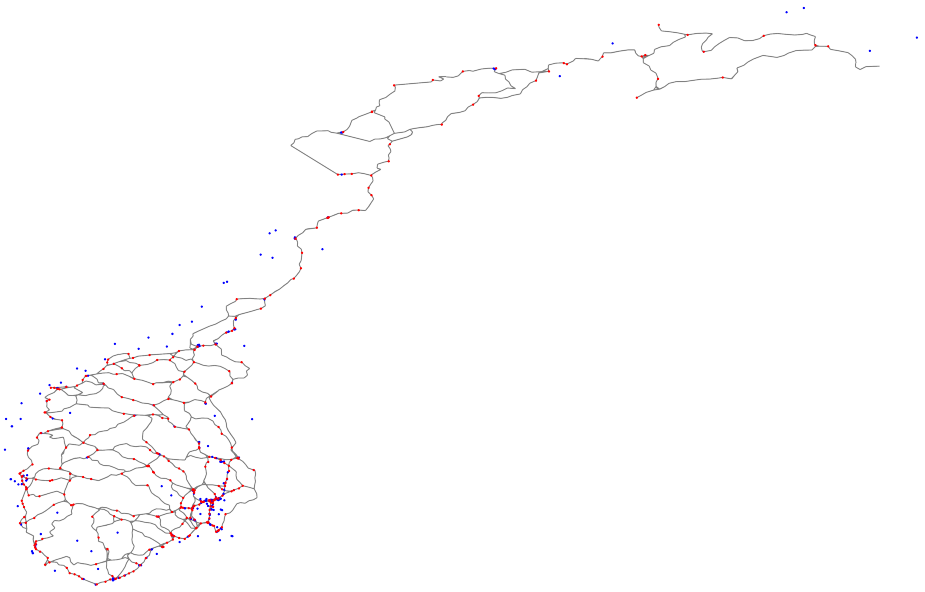


Figure 8: Included and non-included CSs in the current network system

Table 4 reports the minimum, maximum, mean and median deviation in kilometers between CS locations in the current system and to the current network system. Although all 582 charging stations are not modeled in the network, the 330 estimated ones are generally modeled with fairly high accuracy. However, as previously mentioned, some CSs had to be added to the network after the simplification had been performed, as the *Shorten Edges by cut-off* function was unable to capture all CS locations in each path. This implies that some CS locations may deviate with up to 10km, which is the cut-off value used when simplifying the network. This is evident in the table, where the maximum deviation reported is 9.935km. However, seen in light of the median and mean deviation, as well as Figure 5, the current network system can be said to represent the current CS infrastructure sufficiently.

Min Dev.	Max Dev.	Mean Dev.	Median Dev.
0.0002	9.932	1.639	1.521

Table 4: Deviation between actual and modeled CS locations (metres)

Lastly, Table 5 presents the flow captured in the current situation. As it can be seen from the table, the current allocation of fast charging stations in Norway is not sufficient for low-range EVs as only 87.9% of the flow is captured. At

first glance this might appear as an acceptable flow coverage. However, with the current CS infrastructure, low-range EVs can only complete 375 out of the total 820 paths in the network. Furthermore, for the non-captured paths, the average path distance is approximately 1 260km. This is significantly higher than the average network path distance of 779km. This implies that the current infrastructure does not support long-distance travel for low range EVs. The current infrastructure seems to be sufficient for both medium and long range EVs. EVs with a range of 325 km or more can therefore travel between any O-D pair without running out of electricity.

EV Range	170km	325km	585km
Flow Captured	87.9%	100%	100%

Table 5: Flow Captured with current CS locations

5.2 Performance of the Heuristics Algorihms

5.2.1 Test Network

A significant obstacle when solving the FLRM as a MILP is the pre-generation of valid combinations of CSs than can recharge each path in the network. Considering the road network of Norway, it is practically impossible to pre-generate every possible combination, even after the network simplification process has been performed. Consequently, assessing how well the heuristic algorithms perform is impossible, as one cannot compare the results to the optimal solution. Therefore, to analyze the algorithms' performance, a test network consisting of 25 nodes, serving as origins, destinations, and potential CS locations, was created (Berman & Simchi-Levi, 1988). Following (Hodgson, 1990) process, each node and edge was assigned a weight and length, as shown in Figure 5 below.

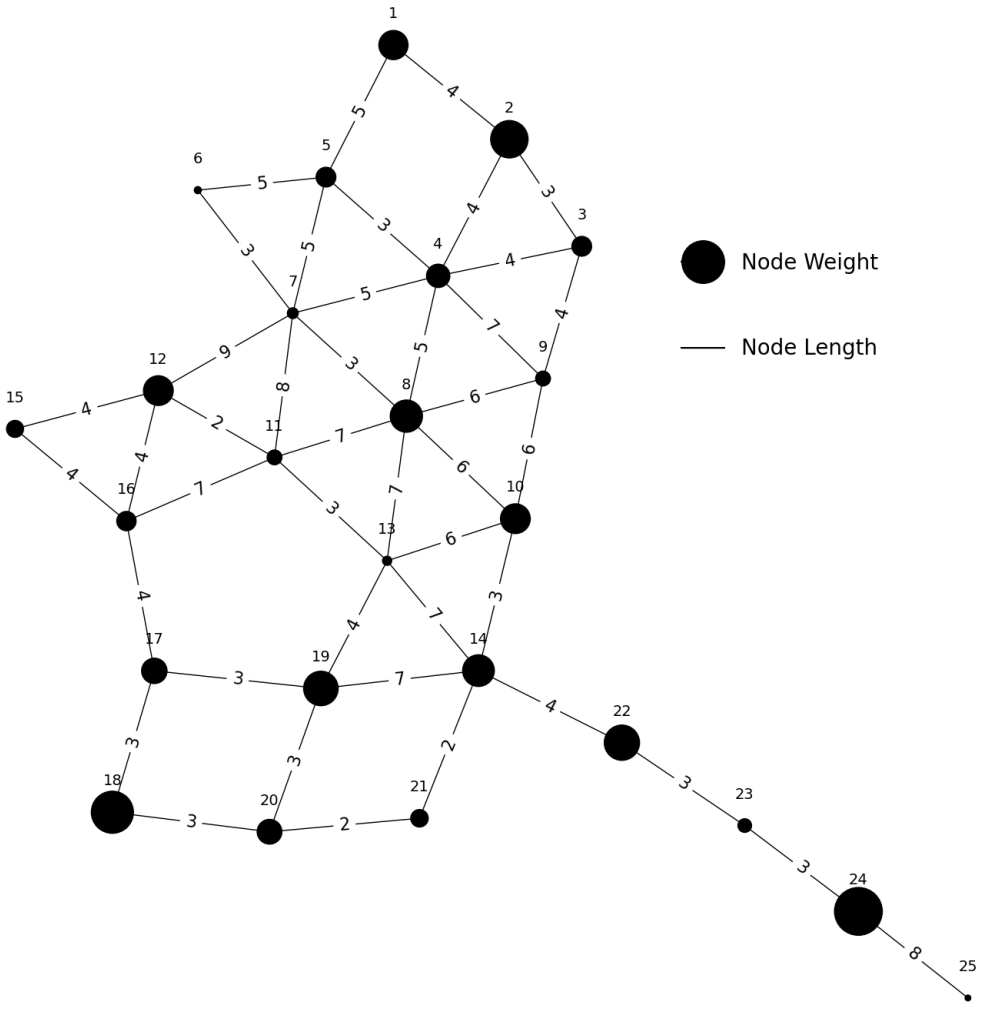


Figure 9: 25-node Test Network

Next, 300 O-D pairs (n(n-1)/2) were defined and each pair was assigned a flow using the following gravity model (Hodgson, 1990):

$$OD_{i,j} = \frac{W_i W_j}{d_{i,j} * *1.5} \tag{7}$$

This thesis considers three different vehicle ranges for solving the FLRM. Therefore, an important part of the testing process was to investigate how the heuristics algorithms performed for different vehicle ranges. The chosen vehicle ranges were 4,10 and 16, as they can be considered short, medium and long range, respectively. For each vehicle range the problem was solved using a p of 5, 10, 15, 20 and 25. The greedy-adding with substitutions was used with up to three substitutions and for the genetic algorithm we used a population limit of 50 and a mutation rate of 0.1. The MILP was solved in Python using *Pulp*, an open source LP modeler written in Python. In order to be able to solve the MILP, all valid combinations of CS that could recharge each path

for a given range had to be pre-generated (Kuby & Lim, 2005), before setting up the parameters. Needless to say, this had to be done for each set of vehicle ranges.

			Optimality Gap (% of gloabl optimum)				
Range	p	Optimal Solution	GA	Greedy-Adding	Greedy Sub 1	Greedy Sub 2	greedy Sub 3
4	5	26.34	0.00	0.70	0.00	0.00	0.00
4	10	56.26	0.00	0.42	0.42	0.42	0.21
4	15	66.56	0.00	0.38	0.38	0.38	0.00
4	20	70.10	0.00	0.40	0.33	0.22	0.00
4	25	70.30	0.00	0.00	0.00	0.00	0.00
10	5	66.81	0.00	0.03	0.00	0.00	0.00
10	10	92.74	0.00	0.02	0.00	0.00	0.00
10	15	99.71	0.00	0.00	0.00	0.00	0.00
10	20	100.00	0.00	0.00	0.00	0.00	0.00
10	25	100.00	0.00	0.00	0.00	0.00	0.00
16	5	77.35	0.00	0.00	0.00	0.00	0.00
16	10	99.03	0.00	0.00	0.00	0.00	0.00
16	15	100.00	0.00	0.00	0.00	0.00	0.00
16	20	100.00	0.00	0.00	0.00	0.00	0.00
16	25	100.00	0.00	0.00	0.00	0.00	0.00

Table 6: Results of test network with vehicle range of 4,10,16

To measure the performance of the heuristics algorithms a measure known as the *optimality gap* is used. For maximization problems this can be defined as:

$$\frac{global\ optimal\ value - heuristics\ \ value}{global\ optimal\ value} \tag{8}$$

In this case, the global optimal value is the optimal solution produced by using the MILP formulation defined in section 3.1. Evident from Table 6, the greedy algorithms did not perform well for a low vehicle range. With a vehicle range of 4 and a p=10, the highest possible flow captured is 56.26%. With 0 and 1 substitution, the greedy algorithm is only able to capture 42% of the optimal flow volume. The result does not improve until 3 substitutions, nonetheless, an optimality gap of 21% is still significantly far away from the optimal solution. For a vehicle range of 10 and 16 the greedy algorithms are generally able to find the global optimum, with some minor exceptions. The genetic algorithm performs flawlessly, finding the global optimum 100% of the time. In fact, the GA is able to find the optimal optimum in only three generations in most instances. Utilizing multiprocessing[9], the GA is also able to produce a result faster than the greedy-adding with 2 and 3 substitutions. To understand why the greedy algorithms perform so poorly for a low vehicle range, the logic of the algorithms must be broken down. Since the greedy-adding algorithm (GAAL) only considers the best solution in the current iteration, hence the name greedy, the chance of becoming trapped in a local optima is fairly high,

[9]This is further explained in Appendix A.

especially for low vehicle ranges. For low vehicle ranges, the first CS selected can only recharge the shorter paths in the network, but this CS might not be part of the optimal combination of CSs. If this is the case, it is ,already after the first iteration, determined that the GAAL will not find the global optimum. This also explains why the GAAL with substitution performs better for short vehicle ranges, as it can substitute the first CS with a more viable location. However, as apparent from Table 6, even three substitutions may be insufficient in order to break out of the local optima in some cases. The results indicate that the genetic algorithm performs better than all variants of the greedy algorithm, especially for lower vehicle ranges. This coincides with the findings that (Kuby & Lim, 2010) reported.

5.2.2 Road Network of Norway

The previous section presented the performance of all three heuristic algorithms in a small 25-node test network. Due to the size of the network, execution time was not an issue for any of the algorithms. The genetic algorithm could be executed with a fairly high population limit for all values of p. There was also no need to consider different mutation frequencies, as a mutation rate of 0.1 was sufficient to obtain the optimal solution. However, for the more complex road network of Norway, these parameters need to be carefully calibrated in order to find the best balance between time efficiency and optimality. Furthermore, the performance of the greedy algorithms must be assessed to determine whether they can handle a network size of 1 583 nodes.

Greedy Algorithms

Although time-efficiency wasn't an issue when the 25-node test network was used, it was uncertain how the greedy algorithms would perform when applied to the road network of Norway. Both algorithms were expected to be able to produce solutions for for a low p, especially since the code had been optimized several times[10]. The GAAL was expected to handle this network size as its runtime is relatively linearly increasing with p. However, it was uncertain whether GAAL with substitutions would be able to produce any output in a timely-manner when increasing p. The reason behind this is that once substitutions are considered, the relationship between p and runtime becomes nonlinear. This is illustrated in Figure 6 below, where the GAAL and GAAL with up to three substitutions were executed in the 25-node test network.

[10]This is explained in Appendix A

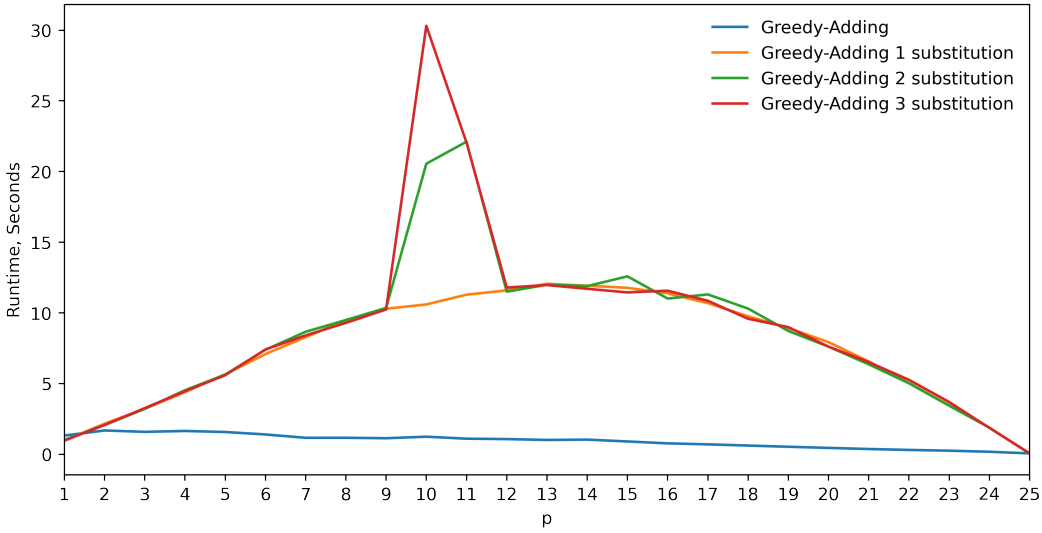


Figure 10: Runtime for the greedy algorithms in the 25-node test Network

It is important to note that each variation of the GAAL was run once with a vehicle range of 25 and p=25. Thus, the figure represents the runtime of adding an additional CS to the selected set. As expected, the runtime of the GAAL is relatively stable for each new CS added. The general concave form of the substitution algorithms' runtime represents the additional time it takes to attempt one substitution. Since the algorithm breaks out of the substitution loop if no substitutions increase the objective value, all three substitution variants generally have an equal runtime. However, when p is between 9 and 12, it is illustrated how the execution time is affected once more than one substitution is performed. In these instances, the substitution loop is not broken, and multiple substitutions are performed. Naturally, this increases the runtime significantly. However, even if only the case with one substitution is considered, the runtime per CS added is still almost eight times as high on average compared to the GAAL.

Unfortunately, due to the non-linearity described above, it was not possible to apply the GAAL with substitutions to the road network of Norway. When it was attempted to run the algorithm with p=50, the algorithm had only selected 27 CSs after 23 hours. Given the nonlinear relationship between runtime and p, it was therefore likely that the algorithm would require several days, if not longer, before it could provide a solution. As a consequence of this, this variation of the GAAL could not be used. The GAAL without substitution, however, was kept.

Genetic Algorithm

The GA is more flexible for the user to calibrate and shape to best fit the problem it is applied for. We tested different values for the population limit (PL), initial population (IP) and mutation frequency (MF) to identify the most efficient way to produce high quality solutions with the algorithm.

Population limit and initial population

The PL given as input determines the maximum set of chromosomes in each generation. A low PL allows the algorithm to run significantly quicker and it tends to converge faster. However, it may compromise the gene diversity in the population, increasing the risk of omitting the optimal set of CS. Hence, the PL must be calibrated to balance the processing efficiency and the quality of the solutions. The IP is the population of chromosomes that is initiated to be used for the first generation. When there are many unique genes to consider, the IP must be large enough to include at least one instance of each gene. To ensure this, sets of chromosomes with p genes each are generated iteratively until every unique gene has been included.

With the 1583 different potential CS locations in the Norwegian network, the IP must have at least $1583/p$ chromosomes to have all the unique genes included (provided that there are no duplicates of the genes). If for example the algorithm solves for 10 CSs (p=10), at least 159 chromosomes are needed to include all the unique genes. It is normal for GAs to have the size of the IP follow the given PL. However, this will result in considerable processing times. In the example of $p = 10$, a population limit above 160 can be considered a high population. At the same time, reducing both the PL and IP will cause genes to be excluded from the algorithms solution space. For this reason, we have performed tests with a PL lower than the size of the IP. The IP is constructed to include all the unique genes, and the first generation will follow the process of crossover and new member selection with this population. Then, only the fittest chromosomes are selected to be part of the next generation. The number of chromosomes selected are determined by the PL, so the subsequent generations can run more effectively with a smaller population while having included all the genes for the initial generation to use.

Table 7 displays the results from using different PLs with $p = 10$ and 200km range. A mutation frequency of 10% is used for the rests, despite the fact that it has not been tested and calibrated yet. With 10 genes in each chromosome, the IP must have at least size of 159. Thus, the PL of 160 allows all the unique genes to be included in each generation, while the lower PLs reduce the population after the first generation. The tests were conducted with a mutation frequency of 10% and the results reflect the averages from three executions.

PL	Flow Captured	Runtime
40	56.11%	456
80	59.68%	615
120	59.98%	892
160	60.84%	1170

Table 7: PL testing

Reducing the PL to 80 and 120 (50% and 75% of the IP) produced slightly worse solutions while reducing run times by 42% and 24% per generation respectively. This is arguably a trade-off worth taking, especially for the further testing and tuning of parameters. A PL of 50% of the size of the IP appears to be the optimal method for balancing the processing time and solution quality. To investigate whether this method would work sufficiently for optimizations many CS, we tested the GA to optimize 70 CS for all three ranges considered in this thesis. To include all genes in the IP, $1583/70 = 23$ chromosomes must be generated. Table 8 compares the results from a PL of 23 and 12 (50% of 23 rounded up), while using the same IP.

PL	EV Range	Percent Flow Captured	Runtime (seconds)
23	170	95.15%	16091
12	170	95.44%	9493
23	325	100%	23816
12	325	100%	14587
23	585	100%	15383
12	585	100%	13192

Table 8: PL testing with $p = 70$

This test suggests that the PL can be reduced to 50% of the IP to significantly reduce the run times, without compromising the quality of the solutions. The captured flow was even slightly higher with the reduced PL. It should be noted, however, that these results come from only one execution and not averages of multiple runs of the algorithm [11]. Hence, The fact that the lower PL produced a better solution for 170km is likely random for this instance, but indicate nonetheless that it should be equally proficient. From these tests, we find it reasonable to apply the efficient implementation of the PL for the further testing and the final results in this thesis. To conclude, the IP will consist of

[11] Given the long run times, multiple executions was not considered a practical option.

1583/p chromosomes and the PL for the subsequent populations is set to 50% of the IP size rounded up.

Calibrating Mutation Frequency

The MF determines how many chromosomes that are mutated in each generation. Such mutations are performed to help the algorithm escape local optimums. Notwithstanding, if the MF is too high, the algorithm may struggle to converge towards the optimal solution and potentially also produce worse solutions, as genes in the optimal solutions risk being substituted with less efficient CSs. To find an appropriate MF for the application on the Norwegian network, we compare the average results from 4 executions of the algorithm with different MFs. To save computational time, the tests were executed to optimize 10 CSs and with an EV range of 200km [12]. Table 9 displays the results form using frequencies of 0, 5, 10 and 20%. In addition, a frequency of 50% was tested to see whether setting it too high would cause evident problems to the convergence of the solutions. "First generation optimum" shows how quickly the algorithm converged to the best solution it was able to find.

Mutation Frequency	Flow Captured	First Generation Optimum
0%	59.78%	11.67
5%	59.70%	13.0
10%	59.09%	15.0
20%	59.45%	14.67
50%	58.83%	15.34

Table 9: Mutation frequency effect on GA results

Interestingly, the results indicate no apparent benefit from using a MF, while there is a tendency that higher frequencies cause a slower convergence.

However, the way in which the MF is implemented, it is applied to each chromosome in the population and not the genes directly. As algorithm 5 illustrates, each chromosome has a certain chance of mutating, and if it does, *one* gene will be substituted. This makes the chance for a gene to mutate highly dependent on number of genes in each chromosome, or in other words, the number of CSs the algorithm solves for. For instance, if a 10% MF is used to solve for 10 CSs, each gene will have a 1% chance of mutating (10% * 10%). However, if this MF is used when we solve for 20 CSs, the marginal probability for a gene to mutate will only be 0.5%. As a result, the fact that these tests were conducted with 10 CSs, means that the results only apply to

[12]It follows from the previous section that we set the PL=80 when p=10: 50%*(1583/10) rounded up

this exact problem, and a different MF may be optimal when solving for a different number of CSs. A way to omit this problem would be to give each gene an equal chance of mutating. We modified the algorithm to test this approach with the same input as above, and a MF for each gene of 0.5% and 1%. This corresponds to a MF of 5% and 10% with the original method. However, as Table 10 shows, the average results from three executions for each MF were worse compared to the original implementation displayed above. In addition, the runtimes increased by 15-20%.

Mutation Frequency	Flow Captured	First Generation Optimum
0%	59.78%	11.67
0.5%	58.40%	12.34
0.1%	58.69%	14.0

Table 10: Mutation frequency for each gene directly

This is not something we would expect, as the marginal mutation probability for each gene is identical, but we will not investigate this any further and keep the MF in its original implementation. Nevertheless, as Table 10 displays, a MF between 0% and 20% does not appear to influence the results, but the higher frequencies need more generations to converge. For this reason, we decided to set a low MF of 5% in the GA for the optimization of CSs in the Norwegian road network, as it may assist the algorithm escape local optimums in some cases and does cause any apparent problems for the solutions.

5.3 Results

This section presents the optimal solutions produced by the GAAL and the GA. Further, it is investigated whether a combination of the two algorithms can provide a superior solution.

5.3.1 Greedy-adding Algorithm

To produce comparable results between the GAAL and GA, it was decided to run both algorithms for three different numbers of CSs for each EV range. For each EV range, it was decided to choose 25, 40 and 70 CSs to be located. The results are presented in Table 11 below in the form of a result-matrix. Computing all nine values took 11.2 hours in total.

	170km	325km	585km
25CSs	71.46%	85.97%	99.36%
40CSs	77.11%	95.08%	100%
70CSs	90.11%	99.70%	100%

Table 11: GAAL result-matrix

The GAAL produces three seemingly good solutions for each EV range, indicating that the process of greedy-adding CSs may be a valuable approach to capture a high flow, considering the EV range. For an EV range of 585km, the GAAL identifies solutions capturing 100% flow by placing both 40 and 70 CSs. Naturally, as the EV range decreases, the same number of CSs is no longer sufficient to capture 100% of the flow. For an EV range of 325km, the captured flow is 99.70% when 70 CSs are placed, and this number drops to 90.11% once an EV range of 170km is considered.

In addition, we wanted to identify which number of CSs that is able to cover all the flow in the network for each range. The GAAL was slightly modified to add CSs until the solution could no longer be improved. Instead of iterating for a specified p number of times, as in the original implementation, the algorithm would now add p CSs until the objective function no longer improved. With this implementation, the GAAL was executed three times, one for each EV range. The results are presented in Table 12.

EV Range (km)	#CS	Flow Captured	Runtime (sec)
170	88	94.40%	6 711
325	62	99.70%	5 138
585	35	99.99%	3 815

Table 12: Results from GAAL for maximizing captured flow

As evident from the table, the GAAL cannot find a solution that captures 100% flow for any of the given EV ranges. For an EV range of 170km, the GAAL suggests that the number of CSs required to maximize the flow is 88. However, this allocation of CSs only results in a flow coverage of approximately 94.4%. Moreover, for an EV range of 325km, the highest captured flow the GAAL reaches is approximately 99.7% with 62 CSs. Finally, for an EV range of 585km, the algorithm identifies a set of 35 CSs that captures 99.99% of the flow volume.

Interestingly, for an EV range of 325 and 585km, the modified GAAL does not improve the captured flow presented in Table 11 for p=70. The flow

capture for the 585km range is, in fact, lower. Nonetheless, the number of CSs the GAAL picks in this implementation is lower than the 70 chosen in the previous execution. In some paths, a set of CSs will be required for the flow to be successfully recharged. The GAAL, however, only considers the extra flow generated from adding one new station in each iteration. Thus, when there are no non-captured paths left that can be recharged by only one additional CS, the algorithm must add a random instance to the solution and try again for a new iteration. As a result, the GAAL is unsuitable for identifying the optimal number of CSs and their locations to capture 100% of the flow. Nevertheless, considering the simple logic behind the algorithm, the results produced can be said to be surprisingly good.

5.3.2 Genetic Algorithm

To compare the results between the GA and the GAAL, they were executed with the same number of CSs for each EV range. For the GA specifically, the population limit and mutation frequency were set according to the calibration results in section 6.2.2. It should be noted that the GA was only executed twice for each range for two main reasons. The first reason is the considerable time consumption, especially for high values of p. In addition, the GA proved seemingly robust after the algorithm-specific parameters were calibrated, producing consistent results over multiple runs. The results produced by the GA are presented in Table 13.

	170km	325km	585km
25CSs	73.69%	88.32%	99.85%
40CSs	79.60%	97.26%	100%
70CSs	94.16%	100%	100%

Table 13: GA Result-matrix

Examining the table, it is evident that the GA is a time-consuming yet powerful algorithm. The total runtime was more than 16 hours, which is considerably longer than the GAAL. Nevertheless, it produced significantly better results, justifying the required computational resources. As the results from Table 12 display, the GAAL was incapable of finding a solution that could capture 100% flow for any EV range by strictly greedy-adding CSs. For this reason, it would be necessary to use the GA to investigate the minimum number of stations needed to capture 100% of the flow. However, this would entail initiating the algorithm for every p until the exact number of CSs needed to cover 100% is identified. Considering the long run times, this is not a feasible endeavor.

Hence, the following section investigates how the GA and GAAL can be used in combination to identify quality solutions in a time-efficient manner.

5.3.3 Combining the GA and GAAL

Combining the two algorithms could be a time-efficient method for producing high-quality solutions. This can be done by executing the GA to optimize a given number of CSs, then adding new CSs to this solution with the greedy algorithm until the desired flow volume is captured. To further maximize the quality of the solution, the GA may be deployed to solve for the new number of CSs identified after the greedy-adding is performed. To further save time, this solution can be included as a chromosome in the initial population for the GA when it is initialized the second time, supplying the algorithm with a good starting point and enabling a rapid convergence. The alternative would be to run the GA for each p until the desired flow has been identified, which is highly time-consuming, if not infeasible, to perform for all three vehicle ranges.

We tested the efficiency of combining the algorithms by trying to find the number of CSs needed to capture 100% flow for each EV range. First, the GA is initialized to find a solution from the GA below 100%. If the first execution resulted in 100% flow, there would be no way of knowing how many CSs in the solution are redundant to capture all the flow, and the algorithm must be executed again with a lower p. Thus, an output solution below 100%, while close to it, is the ideal starting point. From the results in Table 13, we have solution outputs from the GA that are close to 90% for each range. The CS locations for 70, 40, and 25 CSs for 170, 325, and 585km range, respectively, were inserted into the GAAL, adding CSs to the solutions until the flow no longer increases or it reaches 100%. If the GAAL gets stuck before the flow reaches 100%, which occurs when two or more stations are needed to recharge a path, the GA is deployed with the current best solution as a chromosome in the initial population. If the flow is still not 100% from the second iteration of the GA, the process continues until it is reached. The results for each range are displayed in Table 14.

Algorithm Iteration	Flow Captured	Number of CSs	Runtime (sec)
1. GA	94.16%	70	16 026
1. GAAL	97.49%	98	7 485
2. GA	98.83%	98	16 500
2. GAAL	100%	108	6 792

Table 14: 100% flow capture for 170km range

Algorithm Iteration	Flow Captured	Number of CSs	Runtime (sec)
1. GA	97.26%	40	15 100
1. GAAL	98.70 %	59	8 900
2. GA	99.97%	59	34 800
2. GAAL	100%	64	3 000

Table 15: 100% flow capture for 325km range

Algorithm Iteration	Flow Captured	Number of CSs	Runtime (sec)
1. GA	99.85%	25	8 800
1. GAAL	100%	38	4 100

Table 16: 100% flow capture for 585km range

For the 585km EV range, the solution able to capture 100% flow was identified by just one execution of each algorithm. For 325km, two iterations were needed and it was found that 64 CSs can cover all the flow. With the second execution of the GA taking more than 34 000 seconds, which amounts to 9.5 hours, it is clear that this combining of the algorithms was the only feasible way to identify the optimal solutions for 100% flow coverage. For 170km, two iterations were also needed to capture 100% flow by locating 108 CSs. With a runtime of over 14 hours, this would not have been practical to perform by "guessing" the right number of CSs with the GA.

6 Discussion

This section discusses the implications of the results presented in section 6 and certain weaknesses and limitations that affect the results. Following this, the reader will be presented with suggestions for further research.

6.1 Implications of Results

An important aspect of the allocation process of charging stations across the Norwegian road network is the relationship between the captured flow and the number of CSs. Naturally, it is desirable to cover the most demand in a cost-efficient manner. Figure 7 attempts to illustrate this relationship by showing how the flow coverage changes as the number of allocated CSs increases for all given EV ranges.

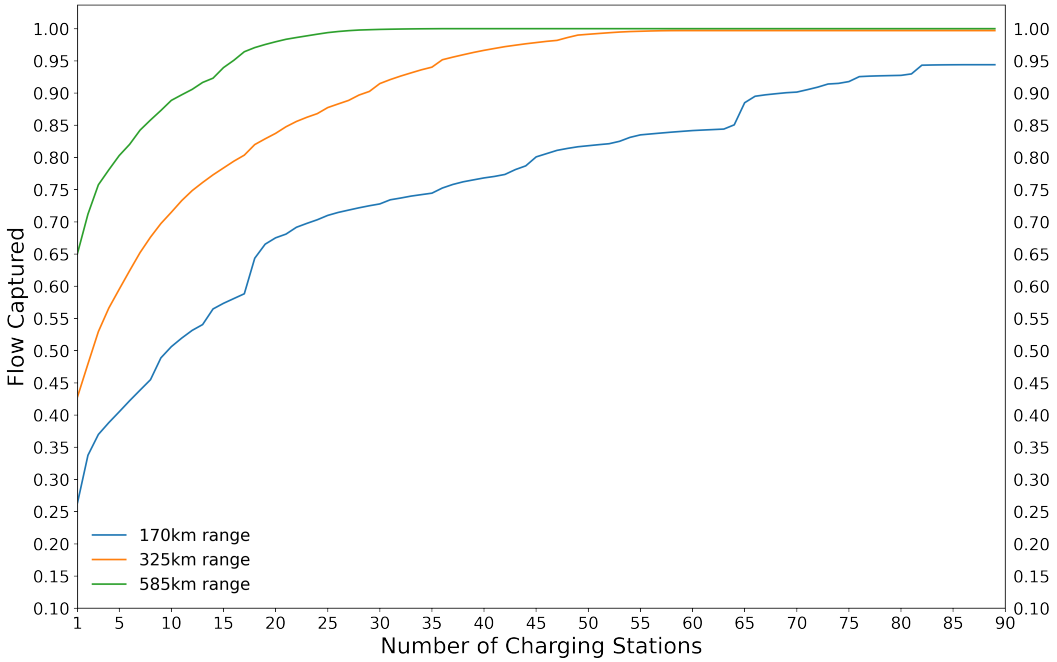


Figure 11: Flow captured for different numbers of CSs

It should be noted that the numbers used in the above figure are results from the GAAL. The GA would likely identify a better allocation capturing a higher amount of flow for each number of CSs. Notwithstanding, deploying the GA for these results would not be feasible. For this reason, the results displayed only estimate the number of CSs needed to capture each flow in the graph, offering some insights into the relationship between the number of CSs and the flow captured. Unsurprisingly, there is a non-linear relationship between the number of CSs and the captured flow, as the marginal gain of adding new CSs decreases as the number of initialized stations increases.

In the current system of existing CSs in Norway, 100% of the flow for 325 and 585 km EV ranges is captured. However, the current infrastructure only captures 87.9% of the flow for an EV range of 170km, which is arguably insufficient. Figure 12 illustrates the locations of all current CS locations (left) compared to an alternative allocation that provides approximately the same flow coverage using the GA. It was found that 55 CSs were sufficient to capture a slightly higher flow than the current infrastructure.

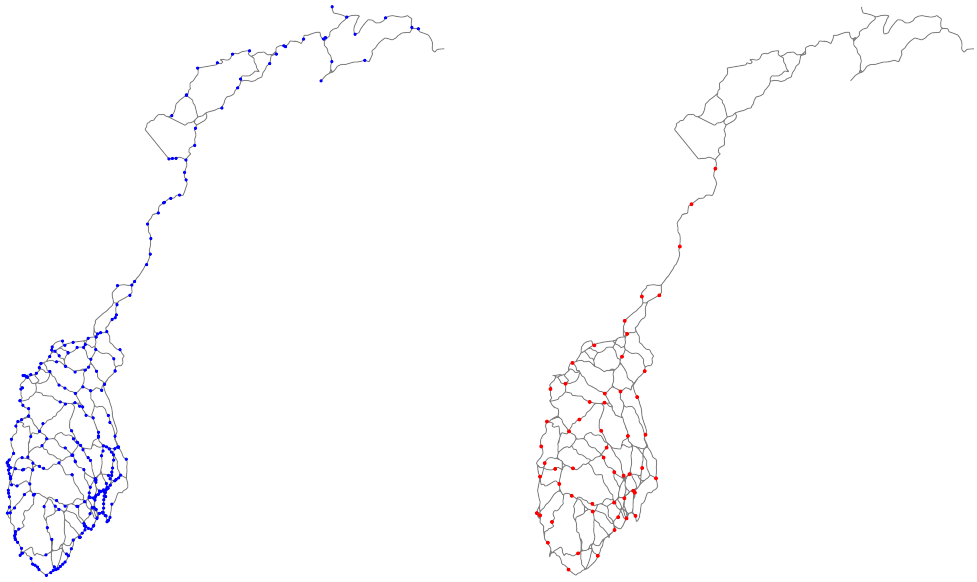


Figure 12: Comparison of current system and a similar flow coverage provided by the GA

The left part of this figure shows the 330 current CS locations in Norway in the network. Despite having a high number of CS with an even distribution across the country, this current infrastructure is only capable of capturing 87.9% flow with a 170km range according to our model. The sub-figure to the right shows the estimated locations of 55 CSs as calculated by the GA, which captures 89.5% flow for the same EV range. This finding is surprising when considering the visual representation of the two maps, especially considering how the GA solution has entirely omitted to locate CSs north of "Bodø," failing to recharge large geographical parts of the country. This is partly due to the modeled flows, which are estimated to be negligible for this area. When the GA accounts for this, it focuses on placing the stations where it can capture more flow.

Another reason is that the estimates for the current system rest on the simplifying assumptions of the model, such as no deviations from the shortest path between an origin and destination node. This assumption can result in a very different objective value for seemingly identical solutions, as many nodes are less than 1km apart. When a flow reaches a destination node, it is assumed to return precisely the way it came, and the model neglects the opportunity to recharge here at a neighboring node only meters away. When the GA locates each CSs, it can account for these minor differences and adjust accordingly by moving a station within the range of a passing flow to capture it. As a result, the way the model is constructed, this comparison with the current system does not provide much insight. If we could model the possibility for slight deviations from the shortest paths, the current system should indeed be able

to cover 100% flow for the shortest range.

Furthermore, another potential weakness in the solution provided by the GA is the possible reason for the lack of CSs allocated in the most northern part of the country. In the current system, there is seemingly a fair amount of CSs in the northern region. It may therefore seem like the GA is more concerned with capturing the flow in the middle and southern regions of Norway, whereas the current system is seemingly lacking the sufficient infrastructure to fully cover these areas. One hypothesis that may explain why the GA does not locate any CSs in the northern parts of the country is that the generated flow from the O-D nodes in the northern parts of the country is exceedingly low. As explained in section 3.1.2, the flow generated is decreasing with both drive time and the length of path. Furthermore, it was seen in Figure 3 that the vast majority of O-D nodes lie in the middle and southern parts of the country. Thus, it becomes apparent that the flow generated by traveling between the northern and southern/middle O-D nodes is significantly lower than if one were to travel exclusively in the middle parts of the country. This being said, the proposed allocation is not necessarily better than how the current system is modeled, as it is contingent on several assumptions and estimations that may not be entirely correct.

However, we can compare the current system with the solutions that capture 100% flow. This may provide some insights into the current system's inefficiencies and where CSs could be redundantly placed. With the results from Table 14, we can visualize the location of all CSs required to capture 100% flow for an EV range of 170km. This is arguably the most interesting range to compare with the current system, as a 100% flow coverage for an EV range of 170km also ensures full coverage for EV ranges beyond this. Figure 13 below shows the proposed locations of all 108 CSs required to capture 100% flow for an EV range of 170km.

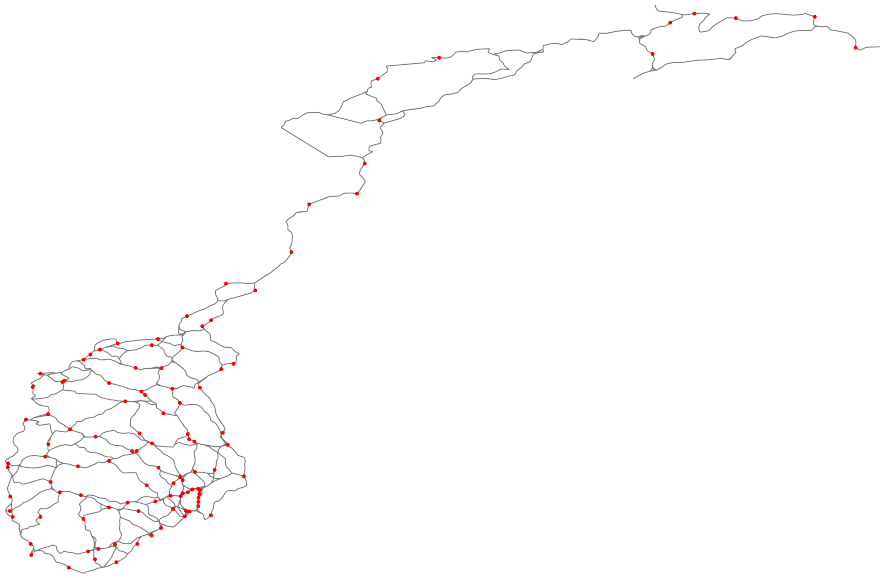


Figure 13: Location of CSs to capture 100% for EV range of 170km

Unlike the solution in Figure 12, CSs are now located throughout the whole country, including the most northern parts. Although the solution is able to capture 100% flow with only 108 stations, certain factors should be considered before accepting the locations as an optimal solution. The first is the fact that the model does not account for the capacity at each CS. Nevertheless, more stations appear to be placed in more densely populated areas, which to some extent satisfies the capacity demands in the areas associated with more flow.

There are several assumptions that must hold if the proposed solution can be said to be an optimal solution. One of the core assumptions that the model is based on are the flow volumes assumed to travel between each O-D pair. Estimating the flows correctly is a difficult task and often relies on a gravity model with many implied assumptions. The gravity model used for this thesis can be considered a simple version, where the flow on an O-D path increases in relation to the number of dwellings and holiday homes, while decreasing in the drive time and distance of the path. Despite this logic holding true in a general sense, how much each of these factors influences the exact flow is not sufficiently modeled for this thesis. To do this requires substantial research on the geographical area being investigated.

Furthermore, the flows are estimated with the assumption that the distributions of EVs are the same across all the O-D nodes, which is unlikely to be true. It is reasonable to assume that the concentration is higher in urban areas, resulting in an overestimation of the flows of EVs traveling between the rural nodes of the network. In addition, the selection of the O-D pairs, both which to choose and the number to include, could be a weakness in our implementation. We made assessments based on general tendencies and aspects

assumed to affect Norwegians' travel habits. For instance, some of the most popular municipalities for holiday homes was chosen and most of the largest cities.

6.2 Further Research

In light of the discussion above, it is clear that there are several model assumptions and limitations that could and should be further investigated in order to improve the feasibility of the proposed solutions.

Firstly, since the generated flows are such a significant part of the solution, further research to identify an accurate implementation of a gravity model for Norwegian intercity and long distance travel would be highly beneficial for the quality of the results proposed by the model. The implemented gravity model is a simple model where the generated flow is a function of drive time, path distance, number of dwellings and holiday homes. Furthermore, finding data or using general qualitative assessments to account for the differences in EV distributions are also suggested for further research to better model the flow volumes of EVs in the network.

Secondly, due to the importance of the number of O-D nodes, as well as their respective locations, more extensive research may be necessary to select a set of O-D locations that more realistically represents the reality of long distance traveling occurring in Norway.

Finally, The assumption of no deviation from the shortest paths is also a significant weakness in this model. If a path cannot be recharged by a given solution, it is assumed that the entire flow cannot be captured. This is a simplification of reality, as most EV drivers would accept to deviate from the path in order to avoid becoming stranded. It should therefore be further researched how to model the human behaviour better. One idea could be to implement the use of deviation paths, to reflect that multiple paths can be used to reach the same destination.

7 Conclusion

In this thesis, the FRLM has been applied to investigate an optimal allocation of charging stations in Norway for electric vehicles. Vehicle ranges of 170, 325 and 585km was considered and a . Since the FLRM cannot by solved as a traditional MILP, the use of alternative methods, like heuristics, is a necessity. Although the genetic algorithm generally produced better results than the greedy-adding algorithm, it is too computationally demanding to be used by itself. Thus, it was found that a combination of the greedy-adding algorithm and the genetic algorithm is better suited for solving this problem, rather than relying on one heuristics approach alone.

Furthermore, this thesis highlights potential inefficiencies in the current system and proposes one solution for each vehicle range where 100% flow coverage is obtained. Although the model employed has certain weaknesses and does not consider the capacity at each charging station location, it can help relevant decision-makers in determining *where* charging stations should be located in order to maximize the demand covered.

References

Baumol, W. J., & Wolfe, P. (1958). A warehouse-location problem. *Operations research*, *6*(2), 252–263.

Berman, O., & Simchi-Levi, D. (1988). Finding the optimal a priori tour and location of a traveling salesman with nonhomogeneous customers. *Transportation Science*, *22*(2), 148–154.

Black, W. R. (2003). *Transportation: a geographical analysis*. Guilford Press.

Boeing, G. (2017). Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, *65*, 126–139.

Cacchiani, V., Hemmelmayr, V. C., & Tricoire, F. (2014). A set-covering based heuristic algorithm for the periodic vehicle routing problem. *Discrete Applied Mathematics*, *163*, 53–64.

Capar, I., & Kuby, M. (2012). An efficient formulation of the flow refueling location model for alternative-fuel stations. *Iie Transactions*, *44*(8), 622–636.

Christiansen, H. (2018). Det er enklere enn noensinne å dra på elbilferie. slik planlegger du turen. *Aftenposten*. Retrieved 2022-06-16, from `https://www.aftenposten.no/motor/i/na9pzd/det-er-enklere-enn-noensinne-aa-dra-paa-elbilferie-slik-planlegger-du-turen`

Chung, S. H., & Kwon, C. (2015). Multi-period planning for electric car charging station locations: A case of korean expressways. *European Journal of Operational Research*, *242*(2), 677–687.

Church, R., & ReVelle, C. (1974). The maximal covering location problem. In *Papers of the regional science association* (Vol. 32, pp. 101–118).

Chvatal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of operations research*, *4*(3), 233–235.

Daskin, M. (1997). Network and discrete location: models, algorithms and applications. *Journal of the Operational Research Society*, *48*(7), 763–764.

Dejax, P. J. (1988). A methodology for warehouse location and distribution systems planning. In *Freight transport planning and logistics* (pp. 289–318). Springer.

Efthymiou, D., Chrysostomou, K., Morfoulaki, M., & Aifantopoulou, G. (2017). Electric vehicles charging infrastructure location: a genetic algorithm approach. *European Transport Research Review*, *9*(2), 1–9.

Elbil.no. (2022). Norwegian ev policy. Retrieved 2022-05-11, from `https://elbil.no/english/norwegian-ev-policy/`

Figenbaum, E. (2018). Electromobility status in norway: mastering long distances–the last hurdle to mass adoption. *TØI report*(1627/2018).

GeoNorge. (2022). National roads database - road network for routing. *Kartkatalogen*. Retrieved 2022-01-13, from https://kartkatalog.geonorge.no/metadata/statens-vegvesen/nvdb-ruteplan-nettverksdatasett/8d0f9066-34f9-4423-be12-8e8523089313

Gwalani, H., Tiwari, C., & Mikler, A. R. (2021). Evaluation of heuristics for the p-median problem: Scale and spatial demand distribution. *Computers, Environment and Urban Systems*, *88*, 101656.

Hodgson, M. J. (1990). A flow-capturing location-allocation model. *Geographical Analysis*, *22*(3), 270–279.

Kuby, M., & Lim, S. (2005). The flow-refueling location problem for alternative-fuel vehicles. *Socio-Economic Planning Sciences*, *39*(2), 125–145.

Kuby, M., & Lim, S. (2010). Heuristic algorithms for siting alternative-fuel stations using the flow-refueling location model. *European Journal of Operational Research*, *204*(1), 51–61.

Kuby, M., Lim, S., & Upchurch, C. (2009). A model for location of capacitated alternative-fuel stations. *Geographical Analysis*, *41*(1), 85–106.

Liu, K., Yamamoto, T., & Morikawa, T. (2017). Impact of road gradient on energy consumption of electric vehicles. *Transportation Research Part D: Transport and Environment*, *54*, 74–81.

Ministry of Transport. (2017). Meld. st. 33 (2016–2017) [Stortingsmelding]. *Regjeringen.no*. Retrieved 2022-01-11, from https://www.regjeringen.no/no/dokumenter/meld.-st.-33-20162017/id2546287/

MirHassani, S., & Ebrazi, R. (2013). A flexible reformulation of the refueling station location problem. *Transportation Science*, *47*(4), 617–628.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Moore, J. L., Folkmann, M., Balmford, A., Brooks, T., Burgess, N., Rahbek, C., ... Krarup, J. (2003). Heuristic and optimal solutions for set-covering problems in conservation biology. *Ecography*, *26*(5), 595–601.

Nwogugu, M. (2006). Site selection in the us retailing industry. *Applied mathematics and computation*, *182*(2), 1725–1734.

OpenChargeMap. (2022). OpenChargeMap API. Retrieved 2022-04-17, from https://api.openchargemap.io/v3

Owen, S. H., & Daskin, M. S. (1998). Strategic facility location: A review. *European journal of operational research*, *111*(3), 423–447.

Pagany, R., Ramirez Camargo, L., & Dorner, W. (2019). A review of spatial localization methodologies for the electric vehicle charging infrastructure. *International Journal of Sustainable Transportation*, *13*(6), 433–449.

Perkowski, J. (2017). How china is raising the bar with aggressive new electric vehicle rules. *Forbes.com*. Retrieved 2022-06-01, from https://www.forbes.com/sites/jackperkowski/2017/10/10/china-raises-the-bar-with-new-electric-vehicle-rules/

?sh=2a44162d77ac

Petroff, A. (2017). These countries want to ban gas and diesel cars. *CNN Money*, *11*.

Plane, D. R., & Hendrick, T. E. (1977). Mathematical programming and the location of fire companies for the denver fire department. *Operations Research*, *25*(4), 563–578.

Ritchie, H., Roser, M., & Rosado, P. (2020). Co and greenhouse gas emissions. *Our world in data*.

Røed, G. (2021). Så stor er elbil-andelen i storbyene. *Motor.com*. Retrieved 2022-06-19, from https://www.motor.no/bompenger/sa-stor-er-elbil-andelen-i-storbyene/101148

Statistics Norway. (2022a). 07849: Registered vehicles, by type of transport and type of fuel (M) 2008 - 2021. Retrieved 2022-05-06, from https://www.ssb.no/statbank/table/07849/

Statistics Norway. (2022b). 06265: Dwellings, by type of building (M) 2006 - 2022. Retrieved 2022-04-03, from https://www.ssb.no/en/statbank/table/06265/

Teitz, M. B., & Bart, P. (1968). Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations research*, *16*(5), 955–961.

Yao, J., Zhang, X., & Murray, A. T. (2019). Location optimization of urban fire stations: Access and service coverage. *Computers, Environment and Urban Systems*, *73*, 184–190.

8 Appendices

8.1 A: Technical Considerations

This section will briefly go through the most important technical aspects that we encountered during the implementation of the code. The algorithms were executed on a computer with 16GB RAM and an Intel Core i5-8500 3.00 GHz CPU.

During the implementation of the heuristics algorithms many steps were conducted in order to optimize the runtime of the greedy-adding, greedy-adding with substitution and the genetic algorithm. The main network that was used in this thesis consists of 1 583 nodes and 1 918 edges. Although this may not be considered large in a general context, it is relatively big compared to previously used networks that have been solved by using the FLRM. The main challenge in this regard was the initially long runtime for the genetic algorithm. The first step that was taken to reduce the runtime of this algorithm was to implement *multiprocessing* for the greedy-deletion crossover part of the algorithm. By implementing multi-processing for this part, the genetic algorithm could perform the greedy-deletion crossover for multiple chromosomes simultaneously instead of looping through one chromosome at a time. This ultimately led to a 600% decrease in the runtime of the genetic algorithm. Unfortunately, this was not possible to implement for the greedy algorithms, as multiprocessing only works for loops that can be *parallelized*.

However, we were successful in further reducing the runtime of all algorithms by performing relatively simple steps. By using tuples and dictionaries wherever possible, we were able to reduce the overall runtime of all heuristics algorithms by approximately 300%. For the genetic algorithm, this was in addition to the already 600%. If the steps above was not performed, there is no way it would have been feasible to use the genetic algorithm. The greedy-adding algorithm could have been possible to use, but this is not certain. This being said, it might seem like Python is not the optimal programming language to use for processor-intensive tasks as those performed in this thesis. A more fitting programming language is C, as this is a structure-oriented language, while Python is a object oriented language.

8.2 B: Links to Code and source data

All code, figures and data required to replicate the results is available on one of the authors' Github page:
https://github.com/aperatinos/Optimal-Location-of-Electric-Vehicle-Charging-Stations-in-Norway-using-the-FLRM

The dataset used to create the road network of Norway can be found here:
https://kartkatalog.geonorge.no/metadata/statens-vegvesen/nvdb-ruteplan-nettverksdat

34f9-4423-be12-8e8523089313