Handout to Honored Leaders from Around the World

Dear Chairman/President:

Thank you very much for reading our letter during the energy summit. Through careful data analysis and thorough research, we hope to bring your attention to the issue related to the carbon emission produced by motor vehicles. Each year, almost 10 GtC of carbon emission is generated and threatens our environment. Our group wish to present our research results to show you that electric vehicle is could be a economically feasible and technically practical alternative to the conventional motor vehicles.

By referring to the database of Tesla, leading firm in this electric vehicle industry, we researched about distribution of charging facilities, including destination charging facilities that allow cars to charge for a long time, and superchargers that charge more rapidly. In our search for a better distribution, we realized that the cost of driving a Tesla electric car is actually far less driving a car using fossil fuels. For instance, by extracting data about electricity charging cost and the battery capacity of Tesla Model 3, the estimated cost of travel per mile is merely $0.037, far less than the cost of $0.2 if we are driving a vehicle powered by fossil fuels. Therefore, the use of electric car is financially beneficial for average users.

We do understand that you may have concerns about the feasibility of the construction of charging facilities. We used different case studies, from urban areas such as New York or Singapore, to suburban area like Wilson City in North Carolina, NC, to countries where population is concentrated in the capital city like Uruguay. After considering the intertwined geographical, political and economical factors, we conclude that the following factors are worth attention if you plan to adopt electric vehicle in your country. Firstly is the population density distribution. Intuitively, more populated place should be equipped with more charging facilities. It is then critical to evaluate the demand for electricity to determine which type of charging facilities to be installed. Next, it is also necessary to consider the possibility of non-local residents using electric vehicles. This is especially important in places where tourism is a significant contributor to the Gross Domestic Product. Installing charging facilities provides not only convenience for travellers but also other drives of electric cars the incentive for visiting. We also used the technique of Benders Decomposition to solve linear programming to find an allocation of charging facilities with minimum cost, which may be helpful as you plan for your country.

We, as a team, sincerely hope that our strategy and proposal would be helpful as you plan for your nation to migrate away from gasoline vehicles.

Yours sincerely
Team Control Number: 93237

1. Introduction

The transportation sector largely depends on liquid fossil fuels, a significant part of the economy. In the United States, the transportation sector accounts for 7.4% of total employment. However, in the past years, attention has been paid globally to reduce carbon footprint through a series of initiatives such as the Paris Climate Agreement. Considering the extensive use of fossil fuels and the waste generated in mobile vehicles, it is in the environmental and economic interest of mankind to break the dependence on oil without compromising mobility and efficiency. Alternatively powered vehicles, such as Tesla Model 3, are therefore gradually occupying the market of mobile vehicles. Nevertheless, several practical concerns have to be resolved to accelerate the migration towards electric vehicles. One primary issue to be addressed is the number and locations of charging stations to be placed, a critical question challenging both policy makers and Tesla.

A lot of work has already proposed models about the distribution of charging stations of electric vehicles, providing several approaches to improve the size and placement of charging infrastructure. Nonetheless, there seems to have no particular article analyzing the existing distribution of Tesla charging facilities, and the projection into future development. Therefore, in this paper, we are going to assess the capability of existing charging facilities provided by Tesla, including destination charging facilities and supercharging facilities, their sufficiency in face of future expansion of the corporation, and variation in placement strategy in different nations or geographies. Several factors are of primary concern: the transportation flow of different locations, the capacity of each type of charging stations.

We are also making key assumptions in this paper to streamline the process of modelling, including:

1. The cost of building each charging station of the same type is invariant.
2. Each charging station is able to accommodate all the vehicles in need of charging within its specified capacity.
3. Energy used is directly proportional to distance travelled.
4. Distance/energy covered by at-home charging is completely ignored.
5. Straight-line distances are used in calculating distances between demand nodes and charging stations.
6. Population and the total number of vehicles remain relatively unchanged in the long term.

2. Examination of Current Tesla Charging Station Networks

2.1 Model

We will use the common Benders Decomposition technique to approach the Capacitated Fixed Charge Facility Location Problem. Benders Decomposition is used to compute the minimum total cost of building facilities and transportation cost of all vehicles, under a few constraints such as meeting all demands of customers at different facilities and taking into consideration of the capacities of each facility.

In this model, we are provided with a set of candidate sites J where a charging station could possibly be located, as well as a set of demand nodes I where electric vehicles are located. The actual charging stations hence form a subset of the candidate sites.
Each consumer will need to travel from one of the demand nodes to a charging station, incurring a travelling cost. In addition, constructing a charging station at a candidate site will also result in a fixed cost.

Below is an abstract mathematical representation of Benders Decomposition.

Inputs

f_j = fixed cost of locating a facility at candidate site $j \in J$

h_i = demand at node $i \in I$

d_{ij} = distance from demand node i to candidate location $j \in J$

k_j = capacity of a facility at candidate site $j \in J$

α = cost per unit distance per unit demand

Decision Variables

$X_j = 1$, if we locate at candidate site j

$\quad 0$, if not

Y_{ij} = fraction of demand at node $i \in I$ that is served by a facility at location $j \in J$

For convenience, we also define the following input and decision variables:

Inputs

$c_{ij} = \alpha d_{ij}$

$\quad$ = the unit shipping cost between candidate location $j \in J$ and demand node $i \in I$

Decision Variables

$Z_{ij} = h_i Y_{ij}$

= the amount of demand at node $i \in I$ that is satisfied by a facility at location $j \in J$

We will

MINIMIZE

$$\sum_{j \in J} f_j X_j + \sum_{i \in I} \sum_{j \in J} c_{ij} Z_{ij} \quad (1)$$

SUBJECT TO:

$$\sum_{j \in J} Zij = h_i \quad \forall i \in I \quad (2)$$

$$Z_{ij} \leq k_i X_j \quad \forall i \in I; j \in J \quad (3)$$

$$\sum_{j \in J} Zij = k_i X_j \quad \forall j \in J \quad (4)$$

$$X_j \in \{0, 1\} \quad \forall j \in J \quad (5)$$

$$Z_{ij} \geq 0 \quad \forall i \in I; j \in J \quad (6)$$

The object function described in (1) minimizes the sum of fixed charging station location costs and transportation costs. In this equation, the cost per unit time and travel time is now combined as a single cost parameter c_{ij}. In principle, (1) is a non-linear function of the flow between demand node i and a charging station at candidate site j. In particular $c_{ij} Z_{ij}$ is a concave function.

Since (1) is stated in terms of flow variables, the demand at node i is captured by the flow variable Z_{ij}. Constraint (2) ensures that all demand at node i is satisfied. Constraint (4) is a capacity constraint. Constraint (5) ensures integrality while (6) ensures nonnegativity.

2.2 Variables

We make the following statements based on relevant data collected:

a. The travelling costs are counted over the duration of one year.
b. The cost of travel per mile is 0.037 dollars, based on the data of effective charging rate as well as the battery capacity information provided by Tesla[1,2].

[1] Lambert, F., & Fred Lambert @FredericLambert Fred is the Editor in Chief and Main Writer at Electrek. Tesla Model 3 production specs revealed: up to 310 miles range, 140 mph top speed, and more.

c. The cost of building a destination charging station is $2,800[3] and that of a supercharger station is $270,000[4].
d. Each supercharger has a useful life of 12 years[5], and we assume that's the same for destination charging station. Hence, we will depreciate the fixed cost over 12 years.
e. Each consumer will drive 13,476 miles per year on average[6]. (Distance travelled between demand nodes and charging stations is ignored.)
f. Exactly half of the consumers will choose either of the battery packs (50kWh and 75kWh)[7]. Thus, the average capacity of each electric vehicle is estimated to be 62.5kWh, which provides a range of 265 miles.
g. An average destination charging station has capacity for 2 cars to charge and each takes 3.91 hours (16kW).
h. An average supercharger station has capacity for 8 cars to charge and each takes 0.521 hours (120kW).
i. 81% of consumers will use at-home charging, and hence we only consider the remaining electricity to be supplied by superchargers and destination charging stations[8].
j. Current market share of electric vehicles is 0.3%.
k. The charging stations are allowed to be utilized, assuming operating at full capacity, for an estimated 12 hours. This is because not all consumers will be charging throughout the day; for example, destination charging stations at workplaces may be more crowded during daytime when people are at work.

Using the data above, we set the following values for variables in our model:

$f_j = \frac{2800}{12} = 233$, if j is a destination charging station

$\frac{270000}{12} = 22500$, if j is a supercharger station

Retrieved February 12, 2018, from https://electrek.co/2017/07/29/tesla-model-3-production-specs-revealed/

[2] F, C. (2017, January 16). Tesla Supercharger calculator aims to compute true cost of travel. Retrieved February 12, 2018, from https://www.teslarati.com/video-cost-actually-traveling-tesla-supercharger-network/

[3] Jen, E. (2016, February 25). Becoming a Tesla Destination Charging Participant. Retrieved February 12, 2018, from https://www.teslarati.com/becoming-tesla-destination-charging-participant/

[4] Keeney, T. (2016, July 11). Supercharger: A Charge Could Cost Half the Price of Gas. Retrieved February 12, 2018, from https://ark-invest.com/research/supercharger-cost-comparison

[5] Rothenberg, P. (2015, January 22). Letter to the SEC. Retrieved February 12, 2018, from https://www.sec.gov/Archives/edgar/data/1318605/000119312515017866/filename1.htm

[6] United States Department of Transportation. (2016, July 13). Average Annual Miles per Driver by Age Group. Retrieved February 12, 2018, from https://www.fhwa.dot.gov/ohim/onh00/bar8.htm

[7] Lambert, F. (2017, August 8). Tesla Model 3 battery packs have capacities of ~50 kWh and ~75 kWh, says Elon Musk. Retrieved February 12, 2018, from https://electrek.co/2017/08/08/tesla-model-3-battery-packs-50-kwh-75-kwh-elon-musk/

[8] 81% of Electric Vehicle Charging is Done at Home. (n.d.). Retrieved February 12, 2018, from https://insideevs.com/most-electric-vehicle-owners-charge-at-home-in-other-news-the-sky-is-blue/

$$k_j = \frac{365.2422\times12\times2}{3.91\times\frac{13476}{265}} = 44.1, \quad \text{if } j \text{ is a destination charging station}$$

$$\frac{365.2422\times12\times8}{0.521\times\frac{13476}{265}} = 1324, \quad \text{if } j \text{ is a supercharger station}$$

$$\alpha = \frac{13476}{265} \times 0.037 = 1.88$$

2.3 Methodology

2.3.1 Data Sources and Extraction

We obtain the geographic coordinates of existing Tesla destination charging stations[9] and Superchargers[10] by searching for relevant html nodes containing the addresses of the stations store these the addresses as strings in a file. We then implement the Google Maps Geocoding API in Python scripts to return the geographic coordinates of the stations. Similar methods are employed to find geographic coordinates of parking lots, residential areas and so on. (See Appendix for more)

Modelling the entire national charging station network with one linear system poses several serious challenges such as the inhomogeneity of variables like unit transportation costs and costs of land to build charging stations, and difficulty in obtaining the data to justify and improve our model.

Fig 1: The Distribution of Tesla superchargers across the U.S

[9] US Tesla Destination Charging | Tesla. (n.d.). Retrieved February 12, 2018, from https://www.tesla.com/findus/list/chargers/United States

[10] US Tesla Superchargers | Tesla. (n.d.). Retrieved February 12, 2018, from https://www.tesla.com/findus/list/superchargers/United%20States

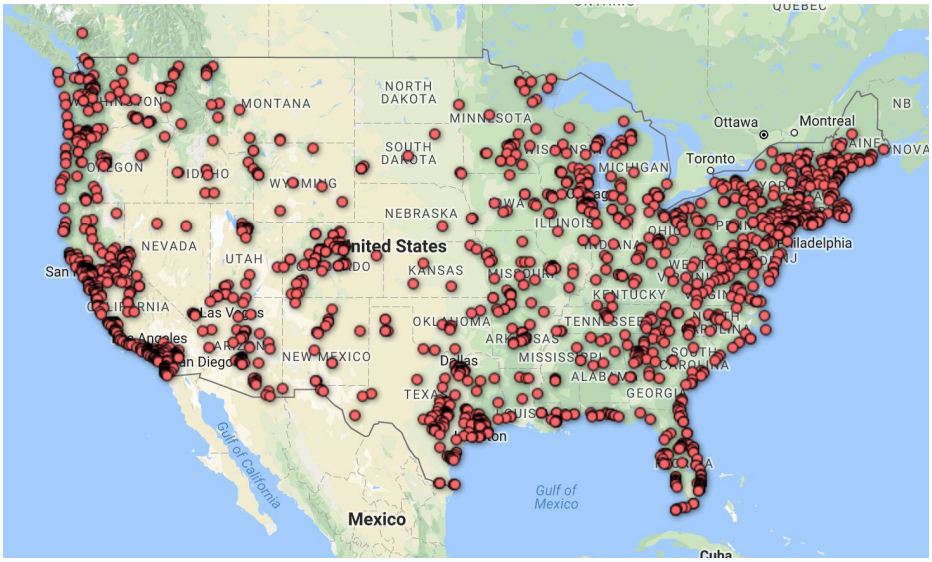

Fig 2: The Distribution of Destination Charging Stations across the U.S

Therefore intuitively, our team decides to extract case studies corresponding roughly to rural, suburban and urban areas of the country and use models with different parameters to examine each case. For categorization into urban, suburban and rural areas, we will adhere to a modified US Census Bureau urban-rural classification[11]. We will redefine urbanized areas of 50000 or more people as urban, urban clusters of 2500 to 50000 people as suburban and all the rest population, housing and territory as rural.

For each case study, we first assume 0.3% of all current vehicles are electric (Statement j), and that 19% of them will contribute to demand nodes (Statement i). This gives us information of all demand nodes within the city. We also extract locations of all existing Tesla supercharger stations and destination charging stations. Providing these inputs into the model therefore allows us to analyze the current situation, such as whether all demands are met and the extent of utilization of existing charging stations.

Subsequently, we analyze the scenarios when 30% and 100% of vehicles have become electric, (modifying or removing the constraint from Statement j). We will first examine whether current charging stations are able to fulfill all demands should such a switch takes place. If more charging stations need to be added, we will generate candidate sites randomly within the approximate geographic boundary of the city. We will generate these sites by randomly choosing demand nodes and adding a candidate site at that location, and thus, there's a greater probability of candidate sites being generated in densely populated areas. We can then use our model to

[11] Branch, G. P. (2012, September 01). Urban and Rural. Retrieved February 12, 2018, from https://www.census.gov/geo/reference/urban-rural.html

determine the number of charging stations to be built, their optimal locations and estimated total cost.

2.4 Case Study

2.4.1 Urban area: New York

New York is selected to represent urban areas. Available population density datasets for New York are mostly on the level of boroughs. The few number of boroughs may not represent the population distribution or more importantly the distribution of demand for transport in different parts of New York. Our team therefore proposes to approximate the distribution of demand for transport using the geographic locations of the various parking lots and garages and their respective capacities. Here we make the assumption that the locations of parking lots are meaningful representations of where high demand could occur, therefore making them suitable candidates for demand nodes. We also make the assumption that the capacity of each parking lot is a good gauge for the density of motor vehicles around it, as it is in the interest of urban planning to provide sufficient but not excessive parking space, considering the high cost of land in urban areas such as New York City. A total number of 818 data points about parking lots and their capacities are extracted from New York Open Data[12]. The location of the candidate charging station is more arbitrary, since there is no real limit on the site of superchargers. In other words, the locations of charging stations are determined primarily for the purpose of sufficing the need of all demand nodes with minimum cost.

[12] Calgary, O. (n.d.). Parking garage list. Retrieved February 12, 2018, from https://data.cityofnewyork.us/Business/parking-garage-list/5bhr-pjxt/data

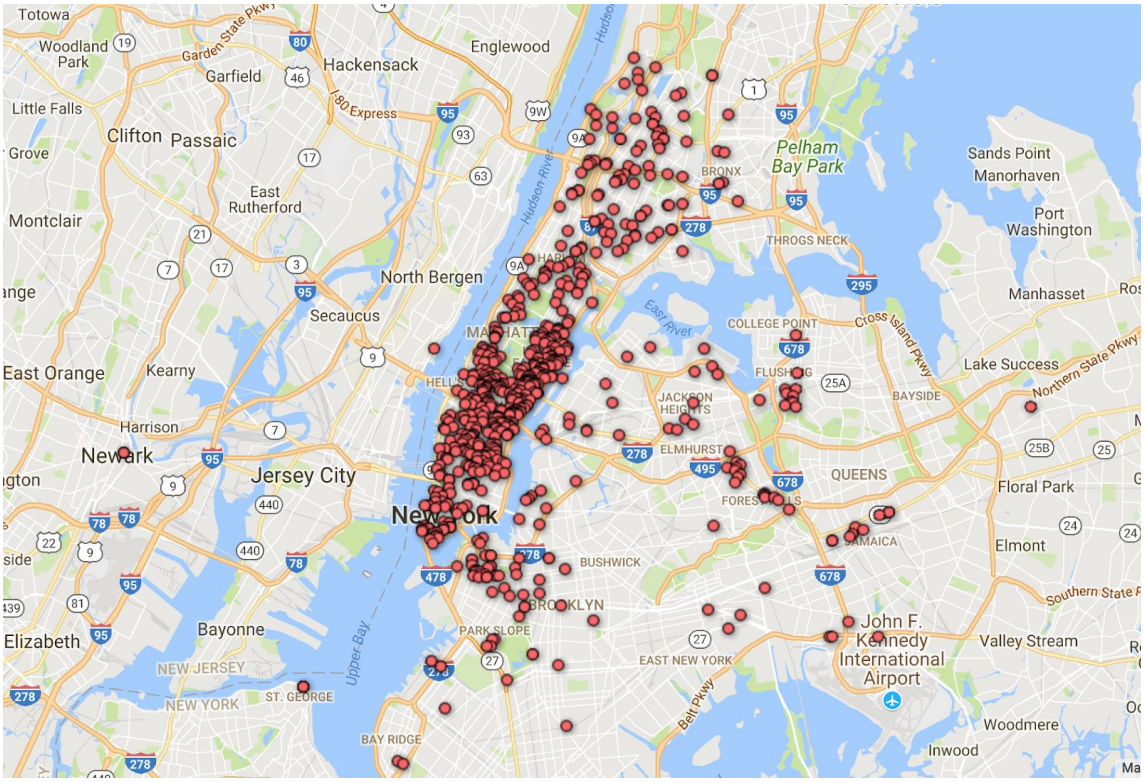

Fig 3: The Distribution of Parking Lots in New York City

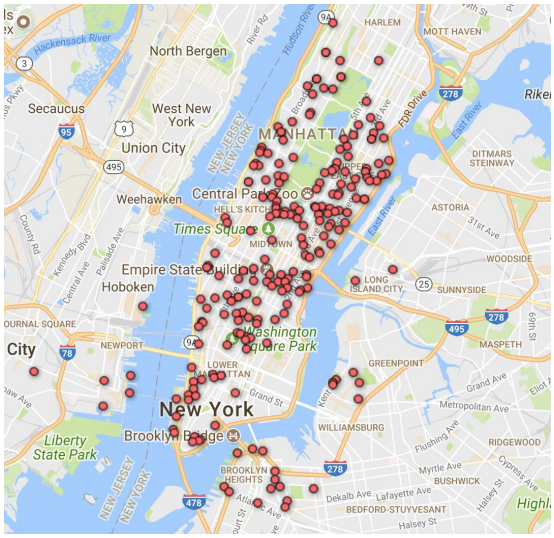

Fig 4: Current Distribution of Destination Charging Stations in New York City

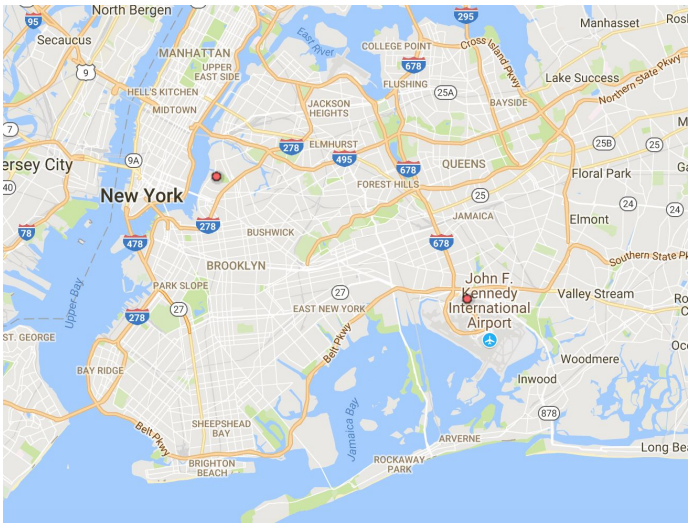

Fig 5: Current Distribution of Superchargers in New York City

Experimentally, our algorithm takes a very long time to execute when the number of candidate sites are large. Therefore, we have decided to merge candidate sites that are close to each other, in order to reduce the number of nodes. In particular, we are merging nodes that are within 0.006

degrees of latitude and longitude, reducing the number of charging stations in consideration from 227 to 49.

Our algorithm returns the following results:

- At the current level, when 0.3% of all vehicles are electric vehicles, demand from all consumers (estimated by parking lots) can be fulfilled using 0 existing superchargers and 2 existing destination charging stations, with a minimal total cost of $857.04 per year:

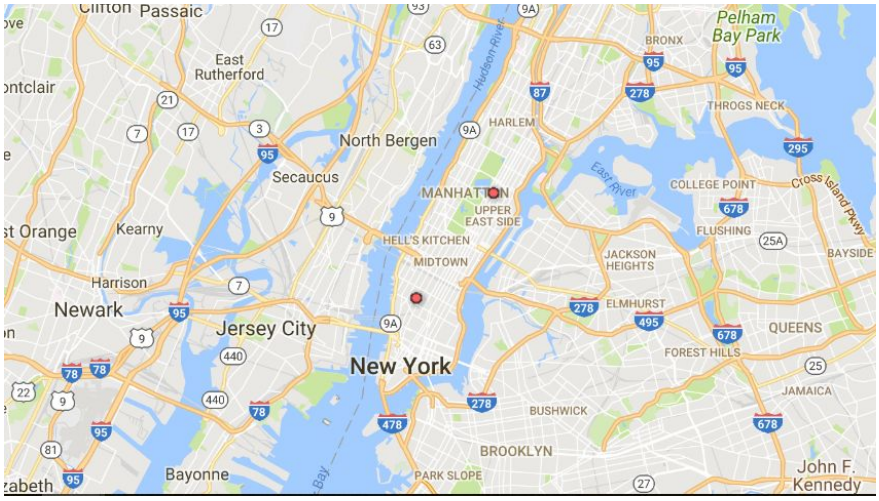

Fig 6: Distribution of Destination Charging Stations Required in New York City (Current)

- If the market share of electric vehicles increases to 30%, demand from all consumers can be fulfilled using 1 existing supercharger and 170 existing destination charging stations, with a minimal total cost of $68046.10 per year:

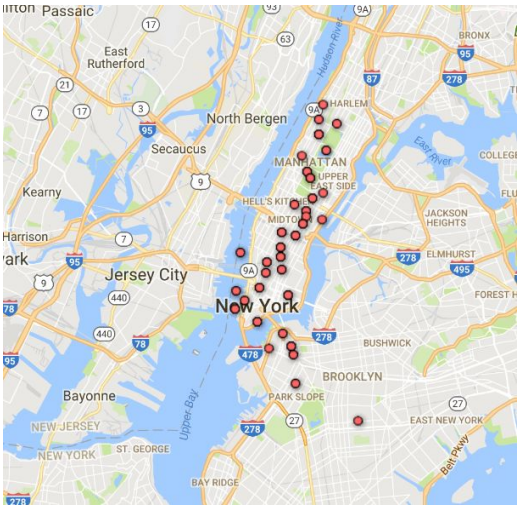

Fig 7: Distribution of Destination Charging Stations Required in New York City (30%)

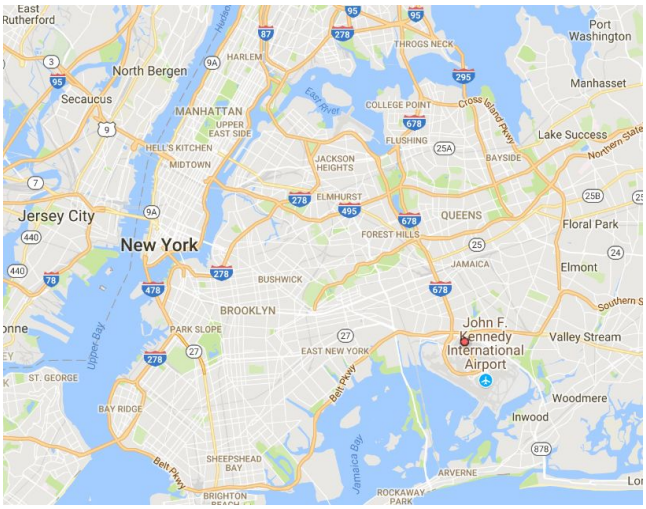

Fig 8: Distribution of Superchargers Required in New York City (30%)

- If all consumers switch to electric vehicles, current charging stations will not be able to fulfill all demand.
  Randomly generating 250 candidate sites for destination charging stations and 10 candidate sites for superchargers, we obtain an optimal solution using 5 superchargers and 453 destination charging stations, with a minimal total cost of $277588.05 per year:

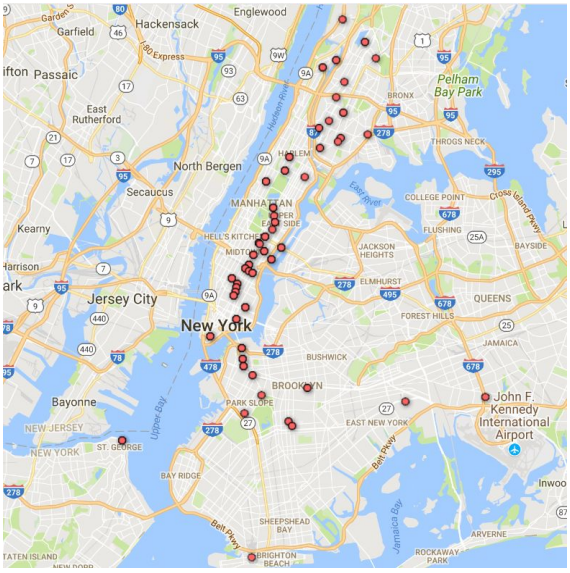

Fig 9: Distribution of Destination Charging Stations Required in New York City (100%)

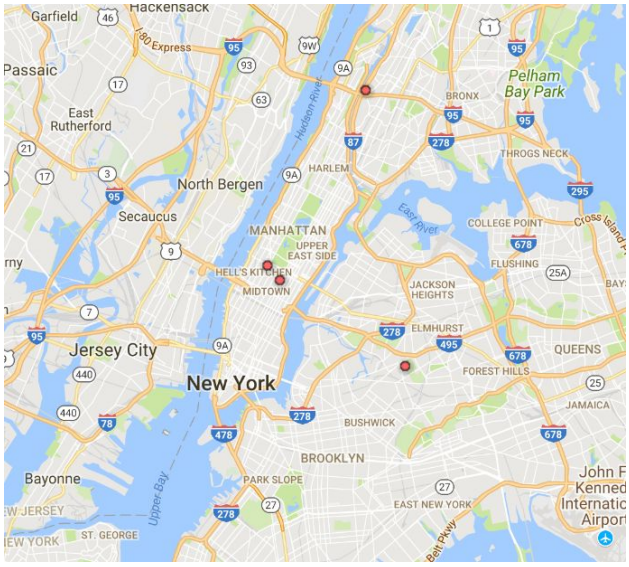

Fig 10: Distribution of Superchargers Required in New York City (100%)

Note that some of the data points in Fig 9 and Fig 10 represent multiple charging stations at the same location, as we have merged charging stations with similar locations into one supply node (with proportionally greater cost and capacity).

2.4.2 Suburban area: Wilson

We select Wilson City, North Carolina as the representation of suburban area since it has a population of slightly more than 49,000, according to the United States Census Bureau[13].

Information on parking lots is not available online, probably due to the small number of parking decks, or most parking decks are unregistered. Therefore, we make use of the population density as a measure of the demand at each demand node, with the assumption that the rate of motor

[13] QuickFacts. (2016). Retrieved February 12, 2018, from https://www.census.gov/quickfacts/fact/table/wilsoncitynorthcarolina,US/PST045216

vehicle ownership is constant throughout different parts of Wilson. We also assumed that with in the area of same population density, the distribution of car is relatively even. Data about population density is extracted from Statistics Atlas[14]. Though the location is therefore less accurate and specific than the parking lots as used in analyzing New York City, the relatively high population density of Wilson implies that the region is small and the error is negligible in this approximation[15].

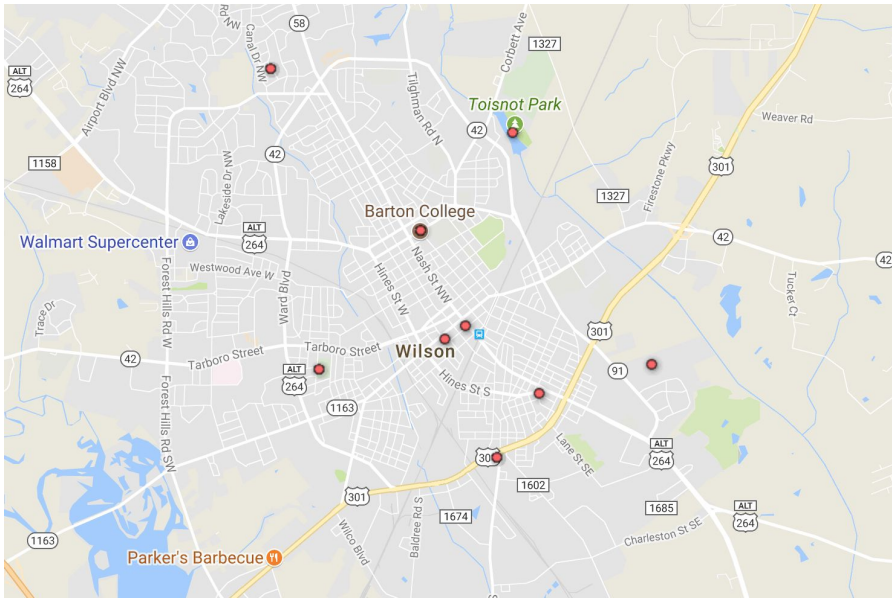

Fig 11: Data Points Chosen to Represent the Population Density in Different Sections of Wilson

Wilson currently has no charging stations. Therefore, we randomly generated candidate sites for 53 destination charging stations and 7 superchargers, each within a random, relative short distance from the demand nodes (with a maximum difference of 0.01 degrees in latitude and longitude).

Using our model, we obtain an optimal solution using 1 supercharger and 49 destination charging stations, with a minimal total cost of $37934.32 per year:

[14] Population of ZIP Code 27896, North Carolina (ZIP Code). (n.d.). Retrieved February 12, 2018, from https://statisticalatlas.com/zip/27896/Population

[15] List of U.S. states and territories by population density. (2018, February 12). Retrieved February 12, 2018, from https://en.wikipedia.org/wiki/List_of_U.S._states_and_territories_by_population_density

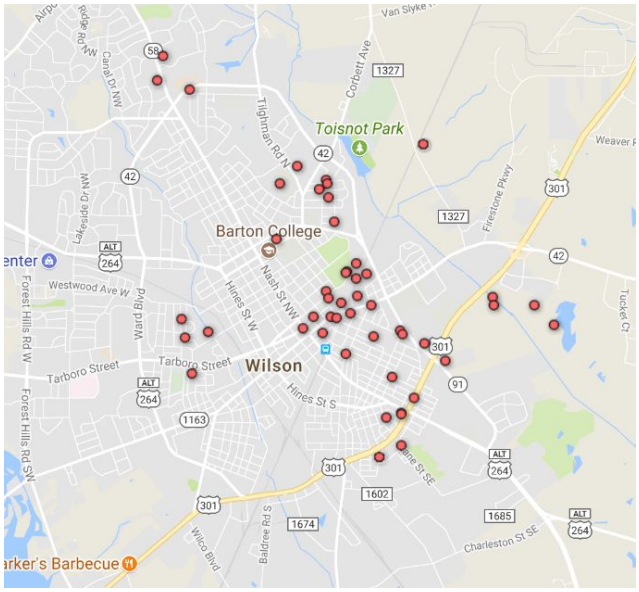

Fig 12: Distribution of Destination Charging Stations Required in Wilson (100%)

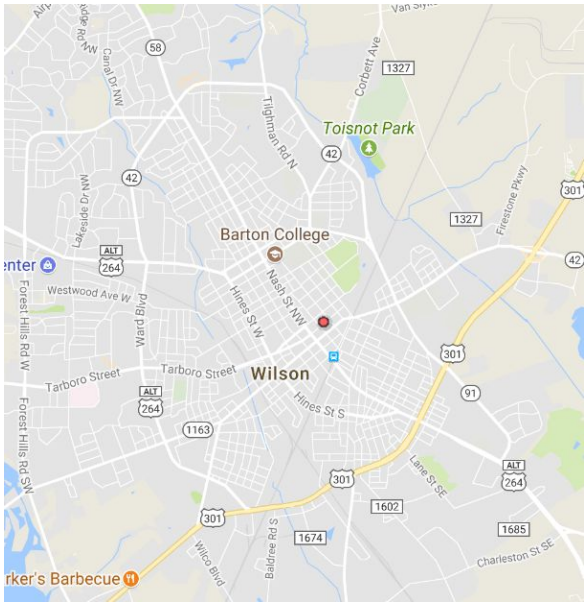

Fig 13: Distribution of Superchargers Required in Wilson (100%)

3. Possible Distribution of Charging Stations in Other Countries

Uruguay is the selected country for case study. Background research indicates that the population density is merely 19.6 persons per square kilometer, with only the capital city Montevideo having a population density exceeding 1000 persons per square kilometer[16]. Such a low population density leads to the reasonable simplification that the population distribution is more or less even across the whole nation, with the exception of Montevideo, which should be treated separately as a urban area. The rest of the country could all be regarded as rural area without addressing the differences among different cities.

3.1 Plans for locations of charging stations

Due to lack of data on populations and parking lots in Uruguay, as well as time constraints, it is extremely hard to use our model to obtain an optimal solution for the construction and allocation of superchargers and destination charging stations. However, we can use the case studies of New York City and Wilson in Task 1 as analogies, in order to determine cities in which charging stations should be placed and the approximate quantities.

[16] Uruguay - Population density (people per sq. km). (n.d.). Retrieved February 12, 2018, from https://tradingeconomics.com/uruguay/population-density-people-per-sq-km-wb-data.html

For example, the population of Montevideo is 1.31 million as of 2011[17], while that of New York City is 8.18 million.[18] Since we have deduced in Task 1 that New York City would require 5 superchargers and 453 destination charging stations should all the consumers switch to electric vehicles, using the same ratio, we can approximate that 1 supercharger and 73 destination charging stations would be optimal for Montevideo. Similarly, we can compare Wilson, NC to other suburban or rural areas in Uruguay, such as the cities of Paysandú and Rivera. The optimal number of destination charging stations for these areas would be similar to that of Wilson which is 49.

However, due to the possible lack of economic capability and the high fixed cost of supercharger stations, it might be difficult to construct a large number of superchargers in Uruguay, even in urban areas. Furthermore, the proportion of residents who own and frequently use cars in Uruguay is likely to be much smaller than that of the United States, due to their economic status and different lifestyles. In reality, only destination charging stations might be feasible for Uruguay.

3.2 Proposal for evolving the charging network

Given the distinct population distribution in Uruguay, it is obvious that Tesla cars, after being imported, will concentrate in the city of Montevideo. Therefore, the most reasonable plan for the expansion of network is the starting from the only urban area, Montevideo. It is advised that destination charger should be the primary choice since its cost its much lower. Though its capacity of 2 is lower than that of supercharger, which has a capacity of 8, at the beginning when there are only limited number of electric vehicles, destination chargers should be sufficient to provide the necessary service. After the need in Montevideo being mostly covered, the following target cities should be cities with next high population densities such as Maldonado, Canelones and San Jose. In these cities, destination charger is also given the priority to be constructed[19]. Meanwhile, the Travel and Tourism industry accounts for 9.4% of the country's GDP. Therefore, key tourist attractions, such as Punta del Este, are also important sites to install chargers. In these cases, superchargers should be installed to provide convenience for travellers. Superchargers may be distributed with approximately constant intervals along the highways with the most traffic volume.

---

[17] Wikipedia. (2018, February 11). Montevideo. Retrieved February 12, 2018, from https://en.wikipedia.org/wiki/Montevideo

[18] Wikipedia. (2018, February 12). New York City. Retrieved February 12, 2018, from https://en.wikipedia.org/wiki/New_York_City

[19] (info@geo-ref.net), H. F. (n.d.). Oriental Republic of Uruguay. Retrieved February 12, 2018, from http://www.geo-ref.net/en/ury.htm

4. Other Factors in Different Geographies

For Tesla to expand its business and network globally, it is both necessary and meaningful to grasp a better understanding of the wealth distribution, essential policies and general public perception in different countries. In this section of the paper, we will be making a detailed examination of specific contexts of different countries to assess the feasibility of our proposed model.

4.1 Evaluate the Business Prospect of Tesla in the context of Singapore

Singapore, the country ranked 10th globally in terms of GDP per capita[20], is a potential target of business expansion. This country is worth analyzing due to its special policy background and geography. Characterized by its small size, Singapore has a high population density as well as building density. As a result, available land for the construction of destination charging station is highly limited. Therefore, our primary focus will be on the construction of supercharging station. It is difficult to analyze the influence of policies in Singapore due to the presence of both supportive and discouraging policies. For instance, the government-issued Certificate of Entitlement (COE) is the quota license that grants the holder the right to use a vehicle in Singapore for 10 years[21]. While the demand for COE far exceeds the number of COE granted each year, the cost of a COE could often be as high as 80,000 dollars. Through limiting the allowed number of cars in usage, Singapore government aims at encouraging the public to make better use of public transport facilities such as Mass Rapid Transit (MRT) and Light Rail Transit (LRT). On the other hand, Singapore government is highly conscious of the reduction of air pollution. In the *Singapore Green Plan* 2012, the government explicitly encourages the use of natural gas in motor vehicles[22]. It is reasonable to infer the use of electric vehicle like Tesla Model 3, as a more environment-friendly alternative to conventional vehicles, will be supported. There have been cases of Tesla cars imported to Singapore being granted tax breaks[23].

In terms of Singapore's demographic distribution, there are 5.607 million people residing in the country with merely 278 mi² of land. The high population density guarantees that we may safely assume the whole country to be urban area. Such homogeneity provides convenience for analysis [24]. It is also unlikely for Singaporeans to travel overseas by driving a motor vehicle since it is an island located on the Pacific ocean, only connected to Malaysia and Indonesia by land.

---

[20] GDP per capita. (n.d.). Retrieved February 12, 2018, from https://tradingeconomics.com/country-list/gdp-per-capita

[21] Certificate of Entitlement (COE). (n.d.). Retrieved February 12, 2018, from https://www.lta.gov.sg/content/ltaweb/en/roads-and-motoring/owning-a-vehicle/vehicle-quota-system/certificate-of-entitlement-coe.html

[22] Document management. Engineering document format using PDF. (2012). doi:10.3403/30148508

[23] Tan, C. (2016, October 26). Two Tesla cars imported into Singapore granted tax breaks. Retrieved February 12, 2018, from http://www.straitstimes.com/singapore/transport/two-tesla-cars-imported-into-singapore-granted-tax-breaks

[24] Singapore. (2018, February 10). Retrieved February 12, 2018, from https://en.wikipedia.org/wiki/Singapore

Therefore, key assumption aforementioned should be adjusted. For instance, the average miles travelled per year , represented by would be significantly lower than the case in the U.S due to the low likelihood of long-distance travelling. As a result, the lower cost of travelling per year would permit drivers to travel for a longer distance to the charging station. The total minimal cost may be achieved with a smaller number of superchargers. On the other hand, the travelling modes by Singaporeans may also be characterized by short distance commutation, especially back and to work. Hence, it would be desirable to place all super chargers at places with high transportation volume, such as the area near Changi Airport or the financial district near Marina Bay.

An important feature of Singapore's landscape is that its roads are rarely straight. Extensive curved roads apparently undermine our assumption in calculating the distances between demand nodes and facility nodes. The straight-line distance will be a great under-estimation of the actual distance travelled. Thanks the small size of Singapore's land area, it is still possible for perform a closer estimation of the actually travelled by considering the actual road length. A percentage error could be estimated to be around 20%.

4.2 Evaluate the Business Prospect of Tesla in the context of Indonesia

Though neighbouring the aforementioned Singapore, Indonesia presents a picture that is in stark contrast. With a total land area of 735,358 mi^2, it is one of the biggest country in Asia[25]. Different landscapes in different cities increases the complexity of the analysis, making it necessary to analyze the situation by categorizing the country into urban, suburban and rural areas again just as we did in the analysis for the United States. However, the GDP per capita in Indonesia is merely $3570.3[26], making the purchase of Tesla electric vehicles in rural areas highly unlikely. To make the discussion about network of charging stations meaningful, it is important to establish a foundation of Tesla's business in the country. For Tesla, t is critical to develop affordable car models for Indonesians, otherwise such expensive cars would only be limited to the wealthiest group of people in the society. Thanks to the low labour cost and low land cost in Indonesia, constructing manufacturing plants in Indonesia is one possible strategy to be employed by Tesla.

When we consider the distribution of charging stations, it is worth noticing that aata collected by the Ministry of Villages, Disadvantaged Regions and Transmigration shows that no less than 52 percent of villages, do not have access to electricity[27]. This is not to say that the construction of

[25] Geography of Indonesia. (2018, February 11). Retrieved February 12, 2018, from https://en.wikipedia.org/wiki/Geography_of_Indonesia

[26] GDP per capita (current US$). (n.d.). Retrieved February 12, 2018, from https://data.worldbank.org/indicator/NY.GDP.PCAP.CD

[27] Mempercepat Perwujudan Pemerataan dan Kesejahteraan. (2017, February 07). Retrieved February 12, 2018, from

charging stations in rural areas where three out of five Indonesians live shall be completely ignored[28]. Currently, it is more feasible to resolve this problem by constructing superchargers close to highways, since in most cases Tesla users are making a short-term stay or a road trip. Constructing destination charging station may lead to a waste of resources.

One particular issue imposing constraints on the construction of charging station in Indonesia is the lack of quality transport infrastructure. As a archipelago consisting of 17,000 islands and a mountainous topography on its land, Indonesia faces difficulty, both financially and technically, in constructing roads[29]. As such, the installation of charging stations should definitely be placed near major traffic conjunctions in order to maximize the usage of electricity. Parking lot is definitely not an appropriate parameter since the number of parking decks and their capacities would be too small for meaningful analysis.

Some more unconventional choices like the flying cars, or self-driving cars are compatible with electric cars so they should not be viewed as rivalry but partners. For instance, flying cars being powered by electricity could avoid the problem of refilling fuel and attract a great number of customers. Nonetheless, currently there are still safety concerns regarding such cutting-edge technologies. Only after numerous tests being performed could the general public accept these technologies better.

After all, the electric vehicle has a promising prospect. As it helps to reduce carbon footprint, it is likely that many government will have policy intensives encouraging people to switch to electric cars. Current obstacles hindering the expansion of electric vehicles possibly include the high cost (as Tesla is still regarded an expensive brand of car, especially in foreign countries), or the power of electric cars being weaker than normal cars. However, as technology matures, such problems are likely to be resolved and electric vehicle opens the gate for many other potential modes of transportation.

---

http://mediaindonesia.com/news/read/91079/mempercepat-pembangunan-infrastruktur-mewujudkan-pemerataan-dan-kesejahteraan/2017-02-08

[28] Preprod. (n.d.). Retrieved February 12, 2018, from https://www.ifad.org/

[29] Dikanaya Tarahita and Muhammad Zulfikar Rakhmat. (2017, May 17). Solving Indonesia's Infrastructure Gap. Retrieved February 12, 2018, from https://thediplomat.com/2017/05/solving-indonesias-infrastructure-gap/

5. Influence of Other Technological Development on Transportation

The emergence of different transportation technologies and related modes of business certainly have an impact on people's attitude towards electric vehicles. In this section, the paper will analyzed the advantage and disadvantage of several alternative environment-friendly solution to the carbon emission problems by vehicles.

First of all, the car share services have the advantage of low cost, especially attractive to the group that do not travel much, such as students studying at college. However, such car share services have great constraint on time and location, resulting in some people not able to use the car when they need it. It is unlikely that car sharing services would have negative influence on the promotion of electric vehicles, since the group of people willing to use the car share service probably have a lot travelling demand, therefore not planning to purchase a car at all. Users of car share service are not the primary target pool of customers of electric vehicles.

Secondly, other alternative fuel, for instance, compressed natural gas (CNG) is one such alternative fuel that produces less pollution and greenhouse gases. Some cars are now powered by both compressed natural gas and fuel, leading to a significant decrease of pollution generated. There have been policy and tax incentives in certain countries regarding the alternative fuel cars, encouraging drivers to migrant to less pollutive choices. Since normal cars can be modified to use CNG as fuel, existing cars need not be abandoned. Compared to electric cars, this choice is more popular among people who have already owned a car. Though there are also disadvantages related to CNG, such as danger in compressing and transporting gases, less power provided by CNG, it is a convenient alternative to traditional means of transportation.

References

Lambert, F., & Fred Lambert @FredericLambert Fred is the Editor in Chief and Main Writer at Electrek. Tesla Model 3 production specs revealed: up to 310 miles range, 140 mph top speed, and more. Retrieved February 12, 2018, from https://electrek.co/2017/07/29/tesla-model-3-production-specs-revealed/

F, C. (2017, January 16). Tesla Supercharger calculator aims to compute true cost of travel. Retrieved February 12, 2018, from https://www.teslarati.com/video-cost-actually-traveling-tesla-supercharger-network/

Jen, E. (2016, February 25). Becoming a Tesla Destination Charging Participant. Retrieved February 12, 2018, from https://www.teslarati.com/becoming-tesla-destination-charging-participant/

Keeney, T. (2016, July 11). Supercharger: A Charge Could Cost Half the Price of Gas. Retrieved February 12, 2018, from https://ark-invest.com/research/supercharger-cost-comparison

Rothenberg, P. (2015, January 22). Letter to the SEC. Retrieved February 12, 2018, from https://www.sec.gov/Archives/edgar/data/1318605/000119312515017866/filename1.htm

United States Department of Transportation. (2016, July 13). Average Annual Miles per Driver by Age Group. Retrieved February 12, 2018, from https://www.fhwa.dot.gov/ohim/onh00/bar8.htm

Lambert, F. (2017, August 8). Tesla Model 3 battery packs have capacities of ~50 kWh and ~75 kWh, says Elon Musk. Retrieved February 12, 2018, from https://electrek.co/2017/08/08/tesla-model-3-battery-packs-50-kwh-75-kwh-elon-musk/

81% of Electric Vehicle Charging is Done at Home. (n.d.). Retrieved February 12, 2018, from https://insideevs.com/most-electric-vehicle-owners-charge-at-home-in-other-news-the-sky-is-blue/

US Tesla Destination Charging | Tesla. (n.d.). Retrieved February 12, 2018, from https://www.tesla.com/findus/list/chargers/United States

US Tesla Superchargers | Tesla. (n.d.). Retrieved February 12, 2018, from https://www.tesla.com/findus/list/superchargers/United%20States

Branch, G. P. (2012, September 01). Urban and Rural. Retrieved February 12, 2018, from https://www.census.gov/geo/reference/urban-rural.html

Calgary, O. (n.d.). Parking garage list. Retrieved February 12, 2018, from https://data.cityofnewyork.us/Business/parking-garage-list/5bhr-pjxt/data

QuickFacts. (2016). Retrieved February 12, 2018, from https://www.census.gov/quickfacts/fact/table/wilsoncitynorthcarolina,US/PST045216

Population of ZIP Code 27896, North Carolina (ZIP Code). (n.d.). Retrieved February 12, 2018, from https://statisticalatlas.com/zip/27896/Population

List of U.S. states and territories by population density. (2018, February 12). Retrieved February 12, 2018, from https://en.wikipedia.org/wiki/List_of_U.S._states_and_territories_by_population_density

Uruguay - Population density (people per sq. km). (n.d.). Retrieved February 12, 2018, from https://tradingeconomics.com/uruguay/population-density-people-per-sq-km-wb-data.html

(info@geo-ref.net), H. F. (n.d.). Oriental Republic of Uruguay. Retrieved February 12, 2018, from http://www.geo-ref.net/en/ury.htm

GDP per capita. (n.d.). Retrieved February 12, 2018, from https://tradingeconomics.com/country-list/gdp-per-capita

Certificate of Entitlement (COE). (n.d.). Retrieved February 12, 2018, from https://www.lta.gov.sg/content/ltaweb/en/roads-and-motoring/owning-a-vehicle/vehicle-quota-system/certificate-of-entitlement-coe.html

Document management. Engineering document format using PDF. (2012). doi:10.3403/30148508

Tan, C. (2016, October 26). Two Tesla cars imported into Singapore granted tax breaks. Retrieved February 12, 2018, from http://www.straitstimes.com/singapore/transport/two-tesla-cars-imported-into-singapore-granted-tax-breaks

Singapore. (2018, February 10). Retrieved February 12, 2018, from https://en.wikipedia.org/wiki/Singapore

Geography of Indonesia. (2018, February 11). Retrieved February 12, 2018, from https://en.wikipedia.org/wiki/Geography_of_Indonesia

GDP per capita (current US$). (n.d.). Retrieved February 12, 2018, from https://data.worldbank.org/indicator/NY.GDP.PCAP.CD

Mempercepat Perwujudan Pemerataan dan Kesejahteraan. (2017, February 07). Retrieved February 12, 2018, from http://mediaindonesia.com/news/read/91079/mempercepat-pembangunan-infrastruktur-mewujudkan-pemerataan-dan-kesejahteraan/2017-02-08

Preprod. (n.d.). Retrieved February 12, 2018, from https://www.ifad.org/

Dikanaya Tarahita and Muhammad Zulfikar Rakhmat. (2017, May 17). Solving Indonesia's Infrastructure Gap. Retrieved February 12, 2018, from https://thediplomat.com/2017/05/solving-indonesias-infrastructure-gap/

Daskin, Mark S. *Network and discrete location: models, algorithms, and applications*. Hoboken, NJ: John Wiley & Sons, Inc., 2013.

Wikipedia. (2018, February 12). New York City. Retrieved February 12, 2018, from https://en.wikipedia.org/wiki/New_York_City

Wikipedia. (2018, February 11). Montevideo. Retrieved February 12, 2018, from https://en.wikipedia.org/wiki/Montevideo

Appendix

Python script that reads addresses from Tesla official website and stores the addresses into strings

```
import geopy
from urllib.request import urlopen

link = "https://www.tesla.com/findus/list/chargers/United+States"
f = urlopen(link)

def extract(line):
        line = line[line.find(">")+1:]
        line = line[:line.find("<")]
        return line

nameList = []
addressList = []
areaList = []
for line in f:
        line = line.decode('utf-8')
        if "locality" in line:
           areaList.append(extract(line))
        if "org url" in line:
           nameList.append(extract(line))
        if "street-address" in line:
           addressList.append(extract(line))
answerList = [", ".join([nameList[i],addressList[i],areaList[i]]) for i in range(len(nameList))]
answerList = [elem for elem in answerList if "coming soon" not in elem]
with open("stationNames.txt",'w') as temp:
   for line in answerList:
      temp.write(line + '\n')
```

Python script that stores addresses of New York parking lots into strings

```
tempList = []
with open("parking_garage_list.csv",'r') as f:
```

```
    for line in f:
        line = line.split(",")
        tempList.append(line[4]+" "+line[6]+" "+line[7]+" "+line[11])

with open("stationNames.txt",'w') as f:
    k = 0
    for entry in tempList:
        k += 1
        if k == 1:
            continue
        if entry.startswith("\""):
            entry = entry[1:]
        f.write(entry+"\n")
```

Python script that returns geographic coordinates of string addresses

```
from geopy.geocoders import GoogleV3
from geopy.exc import GeocoderTimedOut
import sqlite3
from time import sleep

class Cache:
    def __init__(self, fn='cache.db'):
        self.conn = conn = sqlite3.connect(fn)
        cur = conn.cursor()
        cur.execute('CREATE TABLE IF NOT EXISTS '
                    'Geo ( '
                    'address TEXT PRIMARY KEY, '
                    'latitude REAL, '
                    'longitude REAL'
                    ')')
        conn.commit()

    def address_cached(self, address):
        cur = self.conn.cursor()
        cur.execute('SELECT latitude FROM Geo WHERE address=?', (address,))
        res = cur.fetchone()
```

```
        if res is None: return False
        return True

    def save_to_cache(self, address, latitude, longitude):
        cur = self.conn.cursor()
        cur.execute('INSERT INTO Geo VALUES(?, ?, ?)',
                    (address, latitude, longitude))
        self.conn.commit()

def geocode(place):
    try:
        return geolocator.geocode(place)
    except GeocoderTimedOut:
        sleep(1)
        return geocode(place)

geolocator = GoogleV3("GOOGLE API KEY")
with open("stationNames.txt",'r') as source:
    for line in source:
        cache = Cache()
        location = cache.address_cached(line)
        if location:
            continue
        else:
            location = geocode(line)
            if location != None:
                cache.save_to_cache(line, location.latitude, location.longitude)
            else:
                conn = sqlite3.connect('cache.db')
                c = conn.cursor()
                c.execute('INSERT INTO Geo(address) VALUES(?)',(line,))
                conn.commit()
        sleep(0.3)
```

Java code that determines a lower bound of the cost as well as a set of candidate sites where facilities are located, limited to a given set of constraints that is gradually developed during the algorithm

```
import lpsolve.*;
import net.sf.javailp.*;
import net.sf.javailp.Solver;
import java.util.ArrayList;
public class MasterProblem {
	int n;  // Number of demand nodes
	int m;  // Number of candidate sites
	double[] h;  // Demand[n]
	double[] f;  // Fixed cost[m]
	double[] k;  // Capacity[m]
	double[][] c;  // Unit shipping cost (demand -> supply; actually alpha*d)  [n][m]
	//LpSolve solver;
	//int solverCurrentConstraintSize;  // Constraint capacity when solver was initialized (need to expand if necessary)
	SolverFactory factory;
	Problem problem;
	Result result;
	ArrayList<double[]> DX;
	ArrayList<Double> DConst;
	double obj;  // Result of objective function
	double[] var;  // Result of variables [m+1]  (0-indexed)
	public MasterProblem(int n, int m, double[] h, double[] f, double[] k, double[][] c) throws LpSolveException {
	this.n = n;
	this.m = m;
	this.h = h;
	this.f = f;
	this.k = k;
	this.c = c;
	DX = new ArrayList<>();
	DConst = new ArrayList<>();
	//solverCurrentConstraintSize = 500;
	//solver = LpSolve.makeLp(solverCurrentConstraintSize, m);
	factory = new SolverFactoryLpSolve();
	factory.setParameter(Solver.VERBOSE, 0);
	factory.setParameter(Solver.TIMEOUT, 100);
	initializeSolver();
	}
```

```
protected void initializeSolver() throws LpSolveException {
problem = new Problem();
Linear linear = new Linear();
for (int j = 0; j < m; j++) {
linear.add(f[j], Integer.toString(j+1));
}
linear.add(1, Integer.toString(m+1));
problem.setObjective(linear, OptType.MIN);
linear = new Linear();
for (int j = 0; j < m; j++) {
linear.add(k[j], Integer.toString(j+1));
}
double sumh = 0;
for (int i = 0; i < n; i++)
sumh += h[i];
problem.add(linear, ">=", sumh);
for (int j=1; j<=m+1; j++) {
problem.setVarType(Integer.toString(j), Integer.class);
problem.setVarLowerBound(Integer.toString(j), 0);
if (j != m+1) problem.setVarUpperBound(Integer.toString(j), 1);
}
}
public void solve() throws LpSolveException {
Solver solver = factory.get();
Result result = solver.solve(problem);
// Todo: deal with null cases
obj = result.getObjective().doubleValue();
var = new double[m+1];
for (int j=1; j<=m+1; j++)
var[j-1] = result.getPrimalValue(Integer.toString(j)).doubleValue();
}
public void addD(double cst, double[] coeff) {  // coeff should be positive!!!
Linear linear = new Linear();
for (int j = 0; j < m; j++) {
linear.add(coeff[j], Integer.toString(j+1));
}
linear.add(1, Integer.toString(m+1));
problem.add(linear, ">=", cst);
}
```

```
	public double getObj() {
	return obj;
	}
	public double[] getVariables() {
	return var;
	}
}
```

Java code that determines the allocation of demand at each node to facilities (i.e. the transportation problem), given a fixed set of facilities chosen

```
import lpsolve.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
public class TransportationProblem {
	int n;  // Number of demand nodes
	int m;  // Number of candidate sites
	double[] h;  // Demand[n]
	double[] f;  // Fixed cost[m]
	double[] k;  // Capacity[m]
	double[][] c;  // Unit shipping cost (demand -> supply; actually alpha*d)  [n][m]
	double[] x;  // Choice of stations[m]  (Must be integer)
	LpSolve solver;
	double obj;  // Result of objective function
	double[] var;  // Result of variables [n*m]   (0-indexed)
	double[] u;  // Result of demand-side duals
	double[] w;  // Result of supply-side duals
	public TransportationProblem(int n, int m, double[] h, double[] f, double[] k, double[][] c,
double[] x) throws LpSolveException {
	this.n = n;
	this.m = m;
	this.h = h;
	this.f = f;
	this.k = k;
	this.c = c;
	this.x = x;
```

```
solver = LpSolve.makeLp(n+m, n*m);
solver.setVerbose(0);  // Disable output
initializeSolver();
}
protected void initializeSolver() throws LpSolveException {
double[] obj_func = new double[n*m+1];
for (int i=0; i<n; i++)
for (int j=0; j<m; j++)
        obj_func[i*m+j+1] = c[i][j];
solver.setObjFn(obj_func);
solver.setMinim();
for (int i=0; i<n; i++) {
double[] row = new double[n*m+1];
for (int j=0; j<m; j++)
        row[i*m+j+1] = 1;
solver.addConstraint(row, LpSolve.EQ, h[i]);  // Sum over j equals h_i
}
for (int j=0; j<m; j++) {
double[] row = new double[n*m+1];
for (int i=0; i<n; i++)
        row[i*m+j+1] = 1;
solver.addConstraint(row, LpSolve.LE, k[j]*x[j]);  // Sum over i leq k_j*X_j
}
//for (int i=0; i<n*m; i++)
//        solver.setLowbo(i, 0);
// Exception in thread "main" lpsolve.LpSolveException: ERROR in set_lowbo: status =
-1 (Model has not been optimized)
}
public void solve() throws LpSolveException {
solver.solve();
obj = solver.getObjective();
var = solver.getPtrVariables();
double[] duals = new double[(n+m)*2+n*m+1];
solver.getDualSolution(duals);
u = Arrays.copyOfRange(duals, n+m+1, n*2+m+1);
w = Arrays.copyOfRange(duals, n*2+m+1, n*2+m*2+1);
for (int i=0; i<m; i++)
w[i] = -w[i];
}
```

```
	public double getObj() {
	return obj;
	}
	public double[] getVariables() {
	return var;
	}
	public double getTotalCost() {
	double sum = obj;
	for (int i=0; i<m; i++) {
	sum += f[i] * x[i];
	}
	return sum;
	}
	public HashMap<Integer, Double> getSupplyContributions(int dmd) {  // Returns proportions of demand served by each supply node
	HashMap<Integer, Double> map = new HashMap<>();
	for (int j=0; j<m; j++)
	if (var[dmd*m+j] > 1e-6) {
		map.put(j, var[dmd*m+j]/h[j]);
	}
	return map;
	}
	public double[] getUs() {
	return u;
	}
	public double[] getWs() {
	return w;
	}
}
```

Java code that implements the algorithm

```
import lpsolve.LpSolveException;
import net.sf.javailp.SolverFactoryLpSolve;
import java.util.ArrayList;
import java.util.Arrays;
public class Solver {
```

```
	int n;  // Number of demand nodes
	int m;  // Number of candidate sites
	double[] h;  // Demand[n]
	double[] f;  // Fixed cost[m]
	double[] k;  // Capacity[m]
	double[][] d;  //  Distance (demand -> supply)  [n][m]
	double alpha;  //  Unit cost
	double[][] c;  // Unit cost  [n][m]
	double[] x;  // Choice of stations[m]  (Must be integer)
	double[][] y;  // Proportion of demand served by charging station  [n][m]
	double sol;  // Solution
	MasterProblem master;
	TransportationProblem tpt;
	public Solver(int n, int m, double[] h, double[] f, double[] k, double[][] d, double alpha)
throws LpSolveException {
	this.n = n;
	this.m = m;
	this.h = h;
	this.f = f;
	this.k = k;
	this.d = d;
	this.alpha = alpha;
	c = new double[n][m];
	for (int i=0; i<n; i++)
	for (int j=0; j<m; j++)
		c[i][j] = d[i][j] * alpha;
	x = new double[m];
	y = new double[n][m];
	}
	public double solve() throws LpSolveException {
	double low = -Integer.MAX_VALUE;
	double high = Integer.MAX_VALUE;
	master = new MasterProblem(n, m, h, f, k, c);
	System.out.println(low + " " + high);
	int t = 0;  // If lower and upper bounds remain unchanged after 30 iterations, stop
	int t2 = 0;  // If upper bound remains unchanged after 100 iterations, stop
	double prevlow = low;
	double prevhigh = high;
	while (low < high - 1e-6 && t < 30 && t2 < 100) {
```

```
        master.solve();
        low = master.getObj();
        x = Arrays.copyOf(master.getVariables(), m);
        tpt = new TransportationProblem(n, m, h, f, k, c, x);
        tpt.solve();
        // Convert [n*m] to [n][m]
        double[] var = tpt.getVariables();
        for (int i=0; i<n; i++)
                for (int j=0; j<m; j++)
                y[i][j] = var[i*m+j];
        high = Math.min(high, tpt.getTotalCost());
        System.out.println(low + " " + high);
        if (low < high - 1e-6) {
                double cst = 0;
                double[] u = tpt.getUs();
                double[] w = tpt.getWs();
                for (int i=0; i<n; i++)
                cst += h[i] * u[i];
                double[] coeff = new double[m];
                for (int j=0; j<m; j++)
                coeff[j] = k[j]*w[j];
                master.addD(cst, coeff);
        }
        if (low == prevlow && high == prevhigh) t++; else t=0;
        if (high == prevhigh) t2++; else t2=0;
        }
        // The exact solutions are stored in x and y
        sol = high;
        return sol;
        }
        public double[] getFacilitySelection() {
        return x;
        }
        public double[][] getDemandAllocation() {
        return y;
        }
}
```

Java code that reads the coordinates from input files, formats them and calls the Solver to solve the problem

```
import lpsolve.*;
import java.io.*;
import java.util.*;
public class Driver {
	public static class demandNode implements Comparable<demandNode> {
	double x;
	double y;
	double size;
	public demandNode(double x, double y, double size) {
	this.x = x; this.y = y; this.size = size;
	}
	@Override
	public int compareTo(demandNode o) {
	if (Double.compare(x, o.x) != 0)
		return Double.compare(x, o.x);
	if (Double.compare(y, o.y) != 0)
		return Double.compare(y, o.y);
	return -Double.compare(size, o.size);
	}
	}
	public static class supplyNode implements Comparable<supplyNode>{
	double x;
	double y;
	boolean isSuper;
	int multi;  // Multipliers
	public supplyNode(double x, double y, boolean isSuper, int multi) {
	this.x = x; this.y = y; this.isSuper = isSuper; this.multi = multi;
	}
	@Override
	public boolean equals(Object o) {
	if (o == null || !(o instanceof supplyNode))
		return false;
	supplyNode sp = (supplyNode) o;
	return (x == sp.x) && (y == sp.y) && (isSuper == sp.isSuper) && (multi == sp.multi);
```

```
}
@Override
public int compareTo(supplyNode o) {
if (Double.compare(x, o.x) != 0)
        return Double.compare(x, o.x);
if (Double.compare(y, o.y) != 0)
        return Double.compare(y, o.y);
if (isSuper ^ o.isSuper)
        return (isSuper? -1: 1);  // Superchargers always greater than destination chargers
return -Integer.compare(multi, o.multi);
}
}
public static class DemandComparator implements Comparator<demandNode> {
double dx;
public DemandComparator(double dx) {this.dx = dx;}
@Override
public int compare(demandNode o1, demandNode o2) {
boolean samex = false;
if (sameX(o1.x, o2.x, dx))
        samex = true;
if (Double.compare(o1.x, o2.x) != 0 && !samex)
        return Double.compare(o1.x, o2.x);
if (Double.compare(o1.y, o2.y) != 0)
        return Double.compare(o1.y, o2.y);
return -Double.compare(o1.size, o2.size);
}
}
public static class SupplyComparator implements Comparator<supplyNode> {
double dx;
public SupplyComparator(double dx) {
this.dx = dx;
}
@Override
public int compare(supplyNode o1, supplyNode o2) {
boolean samex = false;
if (sameX(o1.x, o2.x, dx))
        samex = true;
if (Double.compare(o1.x, o2.x) != 0 && !samex)
        return Double.compare(o1.x, o2.x);
```

```
        if (Double.compare(o1.y, o2.y) != 0)
                return Double.compare(o1.y, o2.y);
        if (o1.isSuper ^ o2.isSuper)
                return (o1.isSuper ? -1 : 1);  // Superchargers always greater than destination chargers
        return -Integer.compare(o1.multi, o2.multi);
        }
        }
        public static boolean sameX(double x1, double x2, double dx) {
        return Math.floor((x1 - minx) / dx) == Math.floor((x2 - minx) / dx);
        }
        static final double DCSCost = 2800;
        static final double SuperCost = 270000;
        static final int StationLife = 12;
        static final double DCSCars = 2;
        static final double DCSDuration = 62.5/16;
        static final double SuperCars = 8;
        static final double SuperDuration = 62.5/120;
        static final double vehicleRange = 265;
        static final double yearlyMiles = 13476;
        static final double availableHoursPerDay = 12;
        static final double costPerMile = 0.037;
        static final double publicChargingProportion = 0.19;
        //static final double electricMarketShare = 0.003;
        static final double electricMarketShare = 1;
        static final double fDCS = DCSCost / StationLife;
        static final double fSuper = SuperCost / StationLife;
        static final double kDCS = 365.2422 * availableHoursPerDay * DCSCars / DCSDuration / (yearlyMiles / vehicleRange);
        static final double kSuper = 365.2422 * availableHoursPerDay * SuperCars / SuperDuration / (yearlyMiles / vehicleRange);
        static final double alpha = costPerMile * (yearlyMiles / vehicleRange);
        static ArrayList<demandNode> demand;
        static double minx;
        static double maxx;
        static double miny;
        static double maxy;
        static ArrayList<supplyNode> supply;
        static double[] facilitySelection;
```

```
static double[][] demandAllocation;

public static void initialize() {
demand = new ArrayList<>();
supply = new ArrayList<>();
}
public static void readDemand(String demandInput) {
// Extract locations from csv file
minx = Integer.MAX_VALUE;
maxx = Integer.MIN_VALUE;
miny = Integer.MAX_VALUE;
maxy = Integer.MIN_VALUE;
try {
Scanner s = new Scanner(new File(demandInput));
s.useDelimiter(",");
s.nextLine();
while (s.hasNext()) {
        if (!s.hasNextDouble()) {  // Skips faulty lines without leading doubles
        s.nextLine();
        continue;
        }
        double x = s.nextDouble();
        double y = s.nextDouble();
        String str = s.next();
        s.nextLine();
        int z;
        /*try {
        z = Integer.parseInt(str.replaceAll("\"Vehicle Spaces: ", ""));
        } catch (NumberFormatException ee) {
        z = 1;
        }*/
        try {
        z = Integer.parseInt(str.substring(0, str.indexOf("/")));
        } catch (NumberFormatException ee) {
        z = 1;
        }
        demand.add(new demandNode(x, y, z));
        minx = Math.min(minx, x); maxx = Math.max(maxx, x);
        miny = Math.min(miny, y); maxy = Math.max(maxy, y);
```

```
}
s.close();
} catch (FileNotFoundException e) {
e.printStackTrace();
}
}
public static void readSupply(String supplyInput, boolean bounded, boolean isSuper) {
// Extract locations from csv file
try {
Scanner s = new Scanner(new File(supplyInput));
s.useDelimiter(",");
s.nextLine();
while (s.hasNext()) {
        if (!s.hasNextDouble()) {  // Skips faulty lines without leading doubles
        s.nextLine();
        continue;
        }
        double x = s.nextDouble();
        double y = s.nextDouble();
        s.nextLine();
        if (bounded)
        if (!(x>=minx && x<=maxx) || !(y>=miny && y<=maxy))
        continue;
        supply.add(new supplyNode(x, y, isSuper, 1));
}
s.close();
} catch (FileNotFoundException e) {
e.printStackTrace();
}
}
public static void solve() throws LpSolveException {
int n = demand.size();
int m = supply.size();
System.out.println(n + " " + m);
// Create demand array
double[] h = new double[n];
for (int i=0; i<n; i++)
h[i] = demand.get(i).size * electricMarketShare * publicChargingProportion;
// Create supply array
```

```
        double[] f = new double[m];
        double[] k = new double[m];
        for (int i=0; i<m; i++) {
        if (supply.get(i).isSuper) {
                f[i] = fSuper * supply.get(i).multi;
                k[i] = kSuper * supply.get(i).multi;
        } else {
                f[i] = fDCS * supply.get(i).multi;
                k[i] = kDCS * supply.get(i).multi;
        }
        }
        // Create distance array
        double[][] d = new double[n][m];
        for (int i=0; i<n; i++)
        for (int j=0; j<m; j++)
                d[i][j] = DistanceCalculator.distance(demand.get(i).x, demand.get(i).y,
supply.get(j).x, supply.get(j).y);
        // Solve
        Solver sol = new Solver(n, m, h, f, k, d, alpha);
        sol.solve();
        facilitySelection = sol.getFacilitySelection();
        demandAllocation = sol.getDemandAllocation();
        }
        public static void mergeDemand(double dx, double dy) {
        Collections.sort(demand);
        ArrayList<demandNode> newNodes = new ArrayList<>();
        double sumx = 0;
        double sumy = 0;
        double sumsize = 0;
        int sumn = 0;
        int prevIndex = -1;
        for (int i=0; i<demand.size(); i++) {
        if (i == 0 || !sameX(demand.get(prevIndex).x, demand.get(i).x, dx) || (demand.get(i).y -
demand.get(prevIndex).y > dy)) {
                // Sum existing ones
                if (sumn != 0) {
                newNodes.add(new demandNode(sumx / sumn, sumy / sumn, sumsize));
                sumx = 0;
                sumy = 0;
```

```
                sumsize = 0;
                sumn = 0;
                }
                prevIndex = i;
        }
        sumx += demand.get(i).x;
        sumy += demand.get(i).y;
        sumsize += demand.get(i).size;
        sumn++;
        }
        if (sumn != 0) {
        newNodes.add(new demandNode(sumx / sumn, sumy / sumn, sumsize));
        }
        demand = newNodes;
        }
        public static void mergeSupply(double dx, double dy) {
        Collections.sort(supply);
        ArrayList<supplyNode> newNodes = new ArrayList<>();
        double sumxDCS = 0;
        double sumyDCS = 0;
        int sumsizeDCS = 0;
        int sumnDCS = 0;
        double sumxSuper = 0;
        double sumySuper = 0;
        int sumsizeSuper = 0;
        int sumnSuper = 0;
        int prevIndex = -1;
        for (int i=0; i<supply.size(); i++) {
        if (i == 0 || !sameX(supply.get(prevIndex).x, supply.get(i).x, dx) || (supply.get(i).y -
supply.get(prevIndex).y > dy)) {
                // Sum existing ones
                if (sumnDCS != 0) {
                newNodes.add(new supplyNode(sumxDCS / sumnDCS, sumyDCS / sumnDCS,
false, sumsizeDCS));
                sumxDCS = 0;
                sumyDCS = 0;
                sumsizeDCS = 0;
                sumnDCS = 0;
                }
```

```
                if (sumnSuper != 0) {
                newNodes.add(new supplyNode(sumxSuper / sumnSuper, sumySuper / sumnSuper, true, sumsizeSuper));
                sumxSuper = 0;
                sumySuper = 0;
                sumsizeSuper = 0;
                sumnSuper = 0;
                }
                prevIndex = i;
        }
        if (supply.get(i).isSuper) {
                sumxSuper += supply.get(i).x;
                sumySuper += supply.get(i).y;
                sumsizeSuper += supply.get(i).multi;
                sumnSuper++;
        } else {
                sumxDCS += supply.get(i).x;
                sumyDCS += supply.get(i).y;
                sumsizeDCS += supply.get(i).multi;
                sumnDCS++;
        }
        }
        if (sumnDCS != 0) {
        newNodes.add(new supplyNode(sumxDCS / sumnDCS, sumyDCS / sumnDCS, false, sumsizeDCS));
        }
        if (sumnSuper != 0) {
        newNodes.add(new supplyNode(sumxSuper / sumnSuper, sumySuper / sumnSuper, true, sumsizeSuper));
        }
        supply = newNodes;
        }
        public static void outputSuperchargers(String output) {
        int sum = 0;
        try {
        FileWriter file = new FileWriter(output);
        PrintWriter pt = new PrintWriter(file);
        pt.println("latitude,longitude");
        for (int i=0; i<supply.size(); i++) {
```

```
            if (facilitySelection[i] < 0.5) continue;
            supplyNode s = supply.get(i);
            if (s.isSuper) {
            sum += s.multi;
            for (int j = 0; j < s.multi; j++)
            pt.println(s.x + "," + s.y);
            }
}
pt.close();
} catch (IOException e) {
e.printStackTrace();
}
System.out.println("Number of superchargers: " + sum);
}
public static void outputDCS(String output) {
int sum = 0;
try {
FileWriter file = new FileWriter(output);
PrintWriter pt = new PrintWriter(file);
pt.println("latitude,longitude");
for (int i=0; i<supply.size(); i++) {
            if (facilitySelection[i] < 0.5) continue;
            supplyNode s = supply.get(i);
            if (!s.isSuper) {
            sum += s.multi;
            for (int j = 0; j < s.multi; j++)
            pt.println(s.x + "," + s.y);
            }
}
pt.close();
} catch (IOException e) {
e.printStackTrace();
}
System.out.println("Number of DCS: " + sum);
}
public static void generateRandomDCS(int sum) {
double dx = 0.01;
Random random = new Random();
for (int i=0; i<sum; i++) {
```

```
	int ind = random.nextInt(demand.size());
	supply.add(new supplyNode(demand.get(ind).x + random.nextDouble()*dx,
demand.get(ind).y + random.nextDouble()*dx, false, 1));
	}
	}
	public static void generateRandomSuper(int sum) {
	double dx = 0.01;
	Random random = new Random();
	for (int i=0; i<sum; i++) {
	int ind = random.nextInt(demand.size());
	supply.add(new supplyNode(demand.get(ind).x + random.nextDouble()*dx,
demand.get(ind).y + random.nextDouble()*dx, true, 1));
	}
	}
	public static void main(String[] args) throws LpSolveException {
	initialize();
	readDemand("data/wilson_population.csv");
	readSupply("data/supercharger.csv", true, true);
	readSupply("data/destination.csv", true, false);
	generateRandomDCS(53);
	generateRandomSuper(7);
	// If data size is too large, need to merge some demand nodes and/or supply nodes
	//if (demand.size() > 150)
	//	mergeDemand(0.0001, 0.0001);
	//if (supply.size() > 50)
	//mergeSupply(0.025, 0.025);
	//mergeSupply(0.0053, 0.0053);
	//	mergeSupply(0.008, 0.008);
	solve();
	outputSuperchargers("data/Wilson_100_superchargers_new.csv");
	outputDCS("data/Wilson_100_DCS_new.csv");
	}
}
```