For office use only
T1 _____
T2 _____
T3 _____
T4 _____

Team Control Number
79347

Problem Chosen
D

For office use only
F1 _____
F2 _____
F3 _____
F4 _____

2018
The Interdisciplinary Contest in Modeling
Summary Sheet

An Evolving Network with LSA for Vehicle Charging Stations

Summary

In this paper, we propose a systematic model for policy makers and merchants to help them make decisions to realize the switch on this topic. In the beginning, we crawl data and perform necessary data processing with linear regression to enrich our data. Then we design, implement several models and perform experiment to obtain the result. Finally, we do quantitative analysis with respect to our result. The whole scheme we construct can be partitioned into a series of solutions responding to different tasks:

In **Task 1**, we convert the problem of determining the quantity, location and distribution of charging stations to facility location problem in operational research, where we construct a model by introducing **A Circular-city Model with Box-Muller Transform** and **An Approximation Algorithm for Hard Capacitated k-facility Location Problem**. **Task 2** consists of three sub-tasks. For task 2A, we apply the same model used in Task 1 with data replacement and parameters tuning. As for Task 2B, it is, in fact, a supply-and-demand problem. We construct a model of **Local Search Algorithm with Effect Coefficient Matrix** to reach the trade-off between supply and demand of electricity, where optimization objective is a score function. For Task 2C, we modify the model basing on the one for Task 2B. For each time we operate the local search algorithm which is defined as a loop consisting of 10 steps, a batch of charging stations will be built. The transition to car electrification gradually makes progress as the loop iterates (iterate the loop once means building a new batch of charging stations). And finally, when boundary conditions are satisfied, we obtain a 100% switch. Consideration on **Task 3** and **Task 4** is summarized in section 3.4 and section 4.3 respectively.

American metros	Denver	Houston	Miami	Chicago
Charger numbers	156565	539574	296740	756040
South Korean provinces	North Chungcheong	South Chungcheong	Gangwon	Gyeonggi
Charger numbers	81763	164080	226259	122052
American metros	Baltimore	San Diego	New York	Hagerstown
Charger numbers	181067	184947	1076788	54384
South Korean provinces	North Gyeongsang	South Gyeongsang	North Jeolla	South Jeolla
Charger numbers	647020	252768	96516	249018

Conclusions: We responded to all the tasks, constructed a systematic approach to boost car electrification and obtained meaningful and valid data to support our model. For example, we derived quantity of charges of regions of South Korea and U.S., partial result is shown above.

Keywords:
Facility Location Problem, Box-Muller Transform, Local Search Algorithm, Effect Coefficient Matrix

An Evolving Network with LSA for Vehicle Charging Stations

ICM 2018 Team # 79347

Hao Shao, Zihao Ding, Yuhang Zang

February 12, 2018

Content

Summary Sheet

Cover

Content 1

1. Introduction 2
- 1.1 Background 2
- 1.2 Problem Restatement 2
- 1.3 Assumptions 3

2. Data 3
- 2.1 Data Acquisition 3
- 2.2 Data Preprocessing 4

3.Model Construction 5
- 3.1 Notation Interpretation (for Model I) 5
- 3.2 Model I 5
  - 3.2.1 Introduction: Algorithm for Hard Capacitated K-Facility Location Problem 5
  - 3.2.2 Introduction: A Circular-city Model with Box-Muller Transform 5
  - 3.2.3 Solution and Result 5
- 3.3 Model II 10
  - 3.3.1 A Local Search Algorithm with Effect Coefficient Matrix 11
  - 3.3.2 Iterator Model for Gradual Car Electrification Transition 13
  - 3.3.3 Solution and Result 13
- 3.4 Result of Model III 17

4. Evaluation 19
- 4.1 Strengths 19
- 4.2 Weakness 19
- 4.3 Future Work (For Task 4) 19

Reference

Appendix

An Evolving Network with LSA for Vehicle Charging Stations

1 Introduction

In this section, necessary introduction is delivered for readers to get the gist of our solution paper.

1.1 Background

We conceived a network to imitate the distribution of Tesla charging stations in this solution paper, making it come true of "Driving on E" as requirements.

1.2 Problem Restatement

We converted the tasks to make it easier to construct models, and this model can be partitioned into a series of solutions responding to different tasks:

Solution to Task 1 (under circumstances in United States):
The Task 1 actually deals with 2 questions. One question is, how to estimate the **quantity** of Tesla charging stations required, if the transition to 'Driving on E' is immediately 100% finished. Another question is, based on this quantity we just obtained, how can we provide a more generic location distribution plan of the Tesla charging stations for urban areas as well as a generic location distribution plan for rural areas.

Solution to Task 2B and Task 2C (under circumstances in South Korea):
Because Task 2A requires us to estimate the quantity of charging stations and predict the location distribution of charging stations. The idea we used to build this model is similar to the one used in Task 1. For Task 2B, it is, in fact, a supply-and-demand problem. We have to construct a model to obtain the trade-off between charging station construction (electricity supply) and car purchase (electricity demand) with a optimization object. For Task 2C ,we have to figure out an approach to precisely depict the whole progress until we finally realize a complete transition, where every step should be quantitatively analyzed.

Solution to Task 3:
We have to study the former model already constructed to evaluate the model's applicability for countries with appreciable inner difference. If it is not applicable, we have to construct a classification model to help them select the suitable development plan so as to realize this transition.

1.3 Assumptions

In a gesture to conveniently clarify our models and alleviate the workload, with effectiveness of our solution to the largest possible extent guaranteed, we decided to put forward a series of **overall** assumptions, which are applied in our model throughout the entire paper and will not be restated in the ensuing paragraph.

Assumption # 1:
Cities (or metropolitans) are with circular borders. Their respective radii are calculated to assure that the area of circles are equal to the area of real cities.

Assumption # 2:
We assume that all the people who live in urban area in U.S. must live in one of the designated 333 metropolitans listed in file[I] in appendix.

Assumption # 3:
By default, a charging station possesses altogether 1,000 chargers of both types.

Assumption # 4:
The default topographic feature of urban area is plain, which means all other terrains do not exist in our ideal map.

Assumption # 5:
For all the countries, we only take their mainlands into consideration.

2 Data

Data play a role of foundation in our model construction. In this section, we introduce d where and how we fetched the data. Also, how we preprocessed these data to help our model construction will also be clarified.

2.1 Data Acquisition

We obtained the data of urban areas (metropolitans) in U.S. from the websites in reference[a] and [b], with a web scraping program implemented in Python (in appendix[II]). Firstly, from website in reference[a], we obtain land area data of 497 metropolitans.

Further, because of some obstacles in data acquisition from website in reference[b], we discarded some urban areas which are all small cities. We obtain the data of population, GDP per capita and income per Capita for 333 metropolitans out of the 497 ones.

We decided to use data of these 333 metropolitans for our model construction.

Then, we acquired the data in household car parc of 153 metropolitans out of 333 ones from website in reference[c]. The remaining household car parc data of 180 metropolitans can be derived through data preprocessing.

2.2 Data Preprocessing

Necessary data preprocessing is performed in order to obtain the remaining household car parc data of 180 metropolitans with a model of linear regression. We use linear regression to realize curve fitting, so as to acquire the remaining data we want. More specifically:

Linear Regression Equation:

$$Household\_Car\_Parc=(-5.60739831)e\text{-}05*Area+(-1.72475505)e\text{-}08*Population+(7.66397741)e\text{-}06*GDP+(-2.19480196)e\text{-}05*Income+(1.69689609881)$$

Note: e-05 represents *10^(-5) in program.

And finally, we illustrate a small portion of our preprocessed data. A possible structure is shown as below:

A Small Portion of Preprocessed Data in metro_data.tsv

Name of Metropolitan	Area (in km²)	Population	GDP Per Capita (in USD)	Income Per Capita (in USD)	Household Car Parc
Houston	4299.40	6482592	65332	50288	1.59
Madison	391.1	634269	64767	47021	1.5
Cincinnati	2040.20	2146410	53609	47005	1.3
Fresno	443.6	963160	35821	33861	1.63
Savannah	428.6	372569	39510	39612	1.43
Beaumont	237.5	406506	50577	40500	1.62
Birmingham	1372.40	1141309	48405	45311	1.48

Actually, this section concerns only about data preprocessing (a common way to represent). To obtain data that can be used directly by our computer program requires further data processing.

3 Model Construction

3.1 Notation Interpretation (For Model I)

Notations	Interpretation
F	A set of charging station (candidates).
D	A set of user (candidates).
f_i	The cost to build i-th charger
c_{ij}	The cost for customer j to connect to charging station i. (The farer from j to i, the higher the cost)
u_i	The capacity of charging station i.
e_i	The demand of user i for electricity
x_{ij}	A bool value: Is customer j connected to charging station i, True(1) or False(0), $X_{ij} \in \{0, 1\}$
y_i	A bool value: Is charging station i being used, True (1) or False (0), $y_i \in \{0, 1\}$

3.2 Model I:

3.2.1 Introduction: Algorithm for Hard Capacitated K-Facility Location Problem

The problem of hard capacitated k-facility location, which is known as an operational research problem, gives a set of clients with demands, a set of facilities with capacities and constant number k. It costs f_i to open facility i, and c_{ij} for facility i to serve one unit of demand from client j. Because we have to provide a generic location distribution plan of the Tesla charging stations for urban area, where we **introduce** this algorithm.[1] Refer to reference [1] for more detailed information.

We use the linear programming method of this algorithm in 3.2.3 of our solution.

3.2.2 Introduction: A Circular-city Model with Box-Muller Transform

Given we had already assumed that all the cities are with circular borders, we **created** a circular-city model with Box-Muller Transform. The **Box–Muller transform**, by George Edward Pelham Box and Mervin Edgar Muller,[2] is a pseudo-random number sampling method for generating pairs of independent, standard, normally distributed (zero expectation, unit variance) random numbers, given a source of uniformly distributed random numbers.[d] Refer to reference [2] for more detailed information.

We used the Box-Muller Transform to discretize a continuous random variable in 3.2.3 of our solution.

3.2.3 Solution and Result:

The Model I is mainly serves for Task 1. In Task 1, we are asked to estimate the quantity of Tesla charging stations required by the whole United States together with the distribution

between urban areas and rural areas of these charging stations, if the transition to 'Driving on E' is immediately 100% finished. The problem of determining the quantity, location and distribution of charging stations can be converted into facility location problem in operational research. Because it is difficult to directly estimate the quantity of charging stations in U.S., we partition the U.S. into different regions and estimate the quantity respectively. Then we merge the data according to the experiment result. The partition has relation to terrain, car parc (per capita), and affluential extent.

According to section 2, we have already acquired the data of area, population, GDP per capita and car parc per capita of 333 metropolitans in U.S.. When it comes to rural areas, we perform similar approach for data acquisition.

Now, we introduce how to estimate the quantity, location and distribution of charging stations of a specific region:

➤It can be concluded as an optimization object to maximize the quantity of charging stations in a specific region, under the circumstance that users' demand be met and cost of building the charging station be minimized.

$$\sum_{i \in F} y_i * f + \sum_{i \in F} \sum_{j \in D} C_{i,j} * x_{i,j} \qquad (1)$$

➤Hence, the facility location problem is actual a linear programming problem, its corresponding optimization object is formula

1.

$$\sum_{i \in F} x_{i,j} = e_i, \forall j \in D \qquad (2)$$

$$\sum_{j \in D} x_{i,j} \leqslant u_i * y_i, \forall i \in F \qquad (3)$$

$$x_{i,j} \in [0, e_i] \quad y_{i,j} \in \{0, 1\} \qquad (4)$$

➤Formula 2 assures that electricity is fetched as required for every user connected to the charging station. Formula 3 reflects that electricity provided by a charging station for a user is less than its ability to provide. Formula 4 specifies the value range of the variable.

When dealing with the problem, the following content should be taken into consideration:

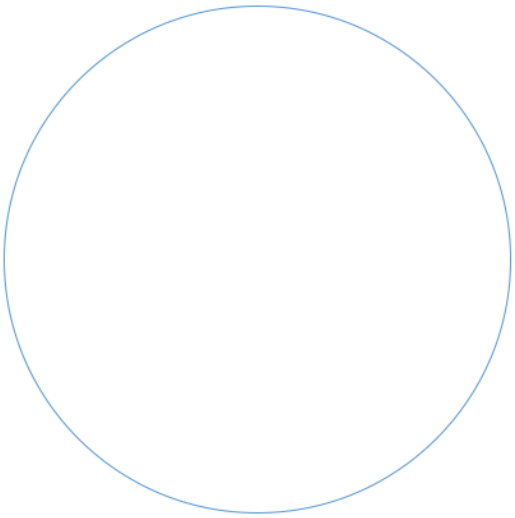
Chargers are of two types: fast charger and destination charger. The rated power of fast charger is higher and can charge up to 80% percent of the full battery within half an hour, which is suitable for emergency and long journey. However, destination charge does little harm to the battery and can charge up to the full battery within 6-8 hours. In our model, the rate of charging has influence on battery capacity and demand of users. Capacity and user demand of fast chargers are smaller than destination chargers. Charges are categorized to into household and public. Users who have household charges will have little demand for public charges.

Variable $\boldsymbol{y}$ is a vector whose elements are 1-bit binary numbers (0 or 1). Each element corresponds to a charging station to build. 1 stands for state of built, 0 stands for state of not built. After calculation, the vector $\boldsymbol{y}$ converge to a value, and its norm (number of non-zero elements of vector $\boldsymbol{y}$) is the quantity of charges to build.

Data Acquisition and Discretization Processing:

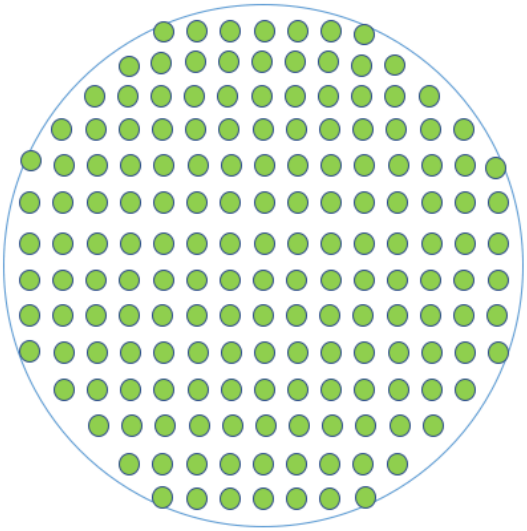
We have now theoretically converted Task 1 to Location Facility Problem in Operational Research. However, in real-world calculation, we have to acquire some precise value of some parameter. Here we introduce how to determine these values:

1. According to area of the region, calculate the radius to make sure the area of circle is equal to the area of region. And we get the radius for each region.



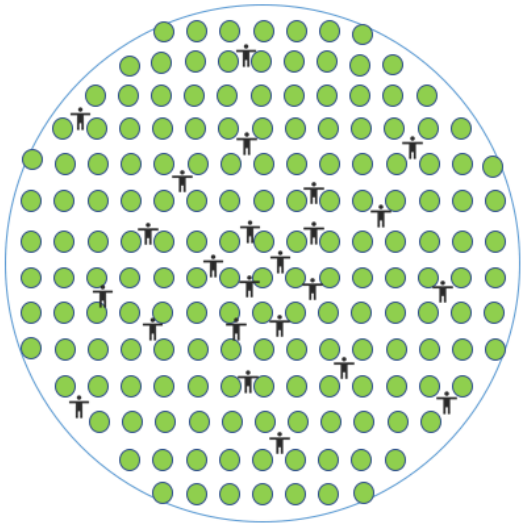
(a) Using a circular to simulate a city

2. We assume these circle regions are charging stations which are uniformly distributed and then, get the $(\boldsymbol{x}, \boldsymbol{y})$ coordinates of each charging stations.

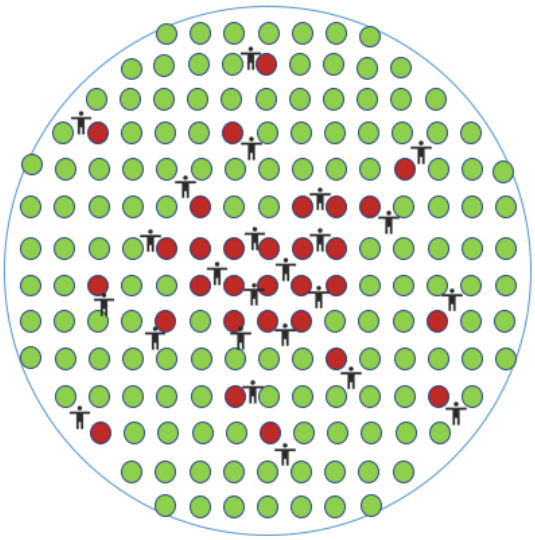


(b) The position of the charging station obeys the uniform distribution

3. Calculate the number of people who require chargers, according to population, GDP per capita, car parc per capita of the region.
4. Perform Box-Muller Algorithm and introduce Normal Distribution to get these users' coordinates.



(c) The location of the user obeys the Gauss distribution



(d) Solving linear programming and obtaining the number of charging stations

The mathematical background of Box-Muller is as follows:

For any continuous distribution, if its cumulative distribution function (c.d.f.) can be derived, we can obtain the inverse function of this c.d.f.. Then we introduce the even distribution, we can realize discretized sampling, based on which, we can obtain sampling with respect to some simple distribution:

➤Assume that random variable x, y are subject to a two-dimensional Standard Normal Distribution, we have:

$$f(x,y) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} \frac{1}{\sqrt{2\pi}} e^{\frac{-y^2}{2}} = \frac{1}{2\pi} e^{\frac{-(x^2+y^2)}{2}}$$

➤A polar coordinates form: $x = r\sin(\theta)$, $y = r\cos(\theta)$, then we have $r^2 = x^2 + y^2$.

➤To derive c.d.f.:

$$P(r \le R) = \int_{r=0}^{r=R} \int_{\theta=0}^{\theta=2\pi} \frac{1}{2\pi} e^{r^2} r dr d\theta = \int_{r=0}^{r} e^{-r^2} r dr$$

➤Substitute r with s, where

$$s = \frac{1}{2} r^2$$

we obtain:

$$P(r \le R) = \int_{s=0}^{r^2/2} e^{-s} ds = 1 - e^{-r^2/2}$$

➤We then perform random sampling on θ (θ is subject to uniform distribution and its range is $0 \le \theta \le 1$), we can realize:

$$\theta = 2\pi U_1$$

➤Then, $1 - e^{-r^2/2} = 1 - U_2$, $r = \sqrt{-1ln(U_2)}$

The algorithm is as follows:

➤Select two random number, which are subject to uniform distribution, where:

$$0 \leq U_1, U_2 \leq 1$$

➤Let $\theta = 2\pi U_1$, $r = \sqrt{-1ln(U_2)}$

➤$x = r\sin(\theta)$, $y = r\cos(\theta)$ are variables that are subject to Standard Gaussian Distribution.

Implementation in Python is as follows:

```
# The Box-Muller algorithm for constructing
pseudo-random Gaussian-distributed numbers

from pylab import *
import numpy as np

def boxmuller(n):

    x = np.zeros((n,2))
    y = np.zeros((n,2))

    for i in range(n):
        x[i,:] = np.array([2,2])
        x2 = x[i,0]*x[i,0]+x[i,1]*x[i,1]
        while (x2)>1:
            x[i,:] = np.random.rand(2)*2-1
            x2 = x[i,0]*x[i,0]+x[i,1]*x[i,1]

        y[i,:] = x[i,:] * np.sqrt((-2*log(x2))/
x2)

    y = np.reshape(y,2*n,1)
    return y

y = boxmuller(1000)
hist(y,normed=1,fc='c')
x = arange(-4,4,0.1)
plot(x,1/
np.sqrt(2*np.pi)*np.exp(-0.5*x**2),'g',lw=6)
xlabel('x',fontsize=24)
ylabel('p(x)',fontsize=24)
show()
```

We have the following parameter to tune in our model. These parameters are converged after linear regression.

1. the cost of building a charging station
2. capacity of a charger (fast charging and destination charging are different)
3. users' demand for chargers (fast charging and destination charging are different, having and not having household charger are different)
4. The conversion ratio of (number of people who require chargers/population)

After we obtain the mathematical description and solution of linear programming, the related constant data, initial value of variable, we write program to calculate the result of every region with data of quantity, location and distribution of charging stations, and finally merge the data to obtain a result of the entire U.S.

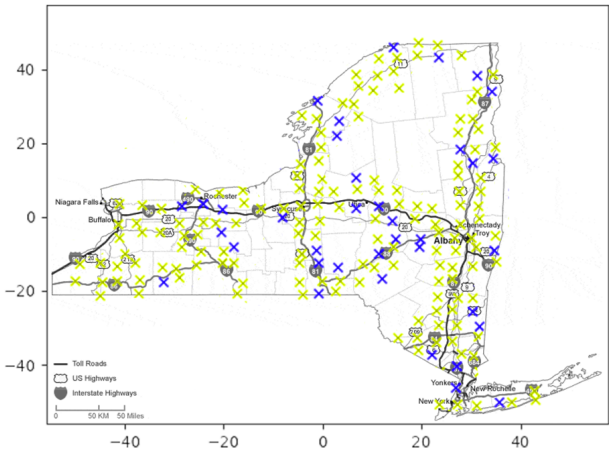
Solution Result and Analysis

The model with Normal Distribution can be vulnerable in calculation due to random seed. To guarantee the robustness of our model, we calculate with different random seed 10 times, and acquire 10 results of total quantity of charges requirement in U.S.. The bias of the 10 results is less than 7.58%, so the robustness of out model is considered acceptable.

By solving Task 1 with this model, we calculated the charges we have to build in U.S.:

18,028,574 chargers in urban areas
2,800,405 chargers in rural areas
20,828,979 chargers in total

The result is listed in task_one_US_charging_stations.csv:
Take New York City and its neighbors as example, the quantity of chargers in New York City and its neighbors is **1076788**, its distribution is shown as below:



Nonetheless, we would like to remind readers that:

1. This model postulates a series of ideal circumstances for purpose of convenient calculation. Furthermore, during this procedure only approximate optimal solution could be obtained, so the outcome of the model is just a robust data which has same order of magnitude with true value.

2. At the same time, we consider that if we could acquire detailed data to eliminate some of the assumption, along with the improvement of modeling, the outcome will be closer to true value.

3. In the model we first divide America into urban area and rural area, then calculate the quantity, place and distribution of charging stations. Hence, the model is dependent on data like area, population, car parc per capita and GDP per capita of each region.

3.3 Model II:

The Model II is mainly serves for Task 2.

The Task 2 switches our battle field from U.S. to South Korea and consists of three sub-tasks — Task 2A, 2B and 2C. It requires us to provide the quantity of charging stations and to predict the location distribution of charging station under circumstance of 100% car electrification transition. Also, it requires us to construct a model to reach the trade-off between charging station construction (electricity supply) and car purchase (electricity demand). Further, we are asked to figure out an approach to precisely depict the whole progress until we finally realize a 100% switch, where every small progress should be quantitative analyzed.

Because Task 2A is a problem to estimate the quantity of charging stations and predict the location distribution of charging station. The idea we use to construct the model in Task 1 is applicable to this Task 2A. So, for task 2A, we apply the same model used in Task 1 just with data replacement and parameters tuning. The modeling process is trivial. We will directly give the result later in this paper.

For Task 2B and Task 2C, we directly clarify our model in 3.3.1 and 3.3.2 respectively.

3.3.1 A Local Search Algorithm with Effect Coefficient Matrix

As for Task 2B, it is, in fact, a supply-and-demand problem. We construct a model of **local search algorithm with effect coefficient matrix** (local search algorithm: use several local optimal solutions to approximate to a global optimal solution) to reach the trade-off between charging station construction (electricity supply) and car purchase (electricity demand), where optimization objective is an ingeniously designed score function.

A quantitative analysis process is as follow:
(including introduction of Effect Coefficient Matrix and Local Search Algorithm):

➤Firstly, we define a matrix, **called Effect Coefficient Matrix (ECM).** An **ECM *J*** is a two-dimensional matrix with ***m*** rows representing ***m*** suppliers (suppliers: ***A*-*Z***) (***m*** charging stations), with ***n*** columns representing ***n*** demanders (demanders: ***a*-*z***) (***n*** electrical cars). Each element in the matrix represents a **effect coefficient**. The J_{Aa} represents the effect coefficient of supplier ***A*** to demander ***a***, whose value is **between 0 and 1**, it is decided by the complexity of topographic situation through the distance. The farer the distance, more complicated the terrain is and the bigger the J_{Ij} value is.the coefficient is defined as follows:

$$J_{Ij} = \frac{1}{\frac{d^{1.5}}{200\beta} + 1}$$

ECM for Super Charger

$$J_{Ij} = \frac{1}{\frac{d^{1.2}}{400\beta} + 1}$$

ECM for Destination Charger

The **Effect Coefficient Matrix *J*** (assume the matrix is $m \times n$) is used to evaluate and record the ability for ***m*** suppliers and ***n*** demanders to interact. The ECM's elements are coefficients. The **higher** a coefficient, for example, the coefficient J_{Aa}, the **higher** the interactive ability between supplier ***A*** and demander ***a***.

The Local Search Algorithm is defined as follow:

➤Step 1, if first running (else if **N = [1, 1 ,1 ,1 ,… , 1], exit**): **initialize the 1×*m* vector *N*, to set all its elements to 0.**

$$N = [0, 0, 0, 0, \ldots, 0]$$

The elements of $\boldsymbol{N}$, n ∈ **{0, 1}**, 1 represents the charging station in this position has been built, and 0 represents the opposite.

➤Step 2, calculate vector $\boldsymbol{H}$.

$$\boldsymbol{H} = \boldsymbol{e_h J^T N} + \boldsymbol{e_i I t} + \boldsymbol{e_g I}$$

$\boldsymbol{e_h}, \boldsymbol{e_i}, \boldsymbol{e_g}$ are all constant. $\boldsymbol{J^T}$ is the transpose of ECM, $\boldsymbol{I}$ is a vector whose elements are all 1. $\boldsymbol{t}$ is variable, its value is equal to the iterator (initial value is 0). $\boldsymbol{H}$ is a vector, depicting the demanding amount of demander.

➤Step 3, calculate vector $\boldsymbol{D}$.

$$\boldsymbol{D} = \boldsymbol{H} - \boldsymbol{vJ}$$

$\boldsymbol{v}$ represents capacity of all the suppliers, which is a vector.

➤Step 4, calculate the vector $\boldsymbol{S}$. This calculation is a function mapping from vector $\boldsymbol{D}$ to vector $\boldsymbol{S}$, calculating the demanding amount to supplier, from demanding amount of demander.

$$\boldsymbol{S} = \boldsymbol{JD^T}$$

➤Step 5, calculate the still-demand-amount vector $\boldsymbol{S'}$ by truncating the elements in $\boldsymbol{S}$ whose value is smaller than 0, or is bigger than corresponding elements in vector **v**. And we derive a new vector $\boldsymbol{S'}$.

➤Step 6, calculate the vector $\boldsymbol{w}$ which represents all the waste amount of demanders for failure to making full use of the supplement from suppliers.

$$\boldsymbol{w} = \boldsymbol{\lambda}(\boldsymbol{v} - \boldsymbol{S'})$$

$\boldsymbol{\lambda}$ is defined as waste coefficient. The high the $\boldsymbol{\lambda}$ is, the less the country cares about resources saving.

Notice: we consider that λ has another meaning — the extent of unwillingness to electrify cars. It comes from the simple logic that if a country has little interest in resources saving, it's less likely to enforce a policy like car electrification. The coefficient will be useful in later analysis in our paper.

➤Step 7, based on step 6, we calculate the score vector $\boldsymbol{x}$.

$$\boldsymbol{x} = \boldsymbol{S'} - \boldsymbol{w}$$

It functions as score in that the equation is simple to understand: more still demand and less waste of resources means a cost-saving plan of suppliers.

➤Step 8, maximize the optimization object:

$$\sum_i A_i x_i M_i + f(\|A\|)$$

M is a Mask-code Matrix. ***A*** is Selection Vector whose elements ***a*** satisfies ***a*** ∈ **{0, 1}**. Find a batch of ***A*** which can maximize the object.

➤Step 9, execute a computer program:

for *i* in [*1*, *m* - ||***N***||]:

A = ***0*** * ***I***, Select the index of first to *i*-th biggest value in ***x***, set the value of corresponding index position in ***A*** as 1

score = $\sum_i A_i x_i M_i + f(\|A\|)$

```
if score > max_score:
        max_score = score
        best_plan = this_plan
```

➤Step 10, update the value of ***N*** and ***t***:

$$N = A + N$$
$$t = t + 1$$

3.3.2 Iterator Model for Gradual Car Electrification Transition

For Task 2C, we modify the model base on the one for Task 2B. We define step 1 to step 10 of model in 3.3.1. as a loop, and a batch of charging stations will be built for each loop. The transition to car electrification gradually makes progress as the loop iterates. And finally, we realize a 100% switch. The model is trivial to repeat.

3.3.3 Result

Task 2A:

By solving Task 2A with corresponding model, we calculated the charges we have to build in South Korea:

2,173,741 chargers in urban areas
256,607 chargers in rural areas
1,917,134 chargers in total

Task 2B:

Notice: The following statement only focuses on North Chungcheong Province instead of the whole South Korea. Suggested development plan: We would better build chargers in places where people and vehicle gathers at early stage.

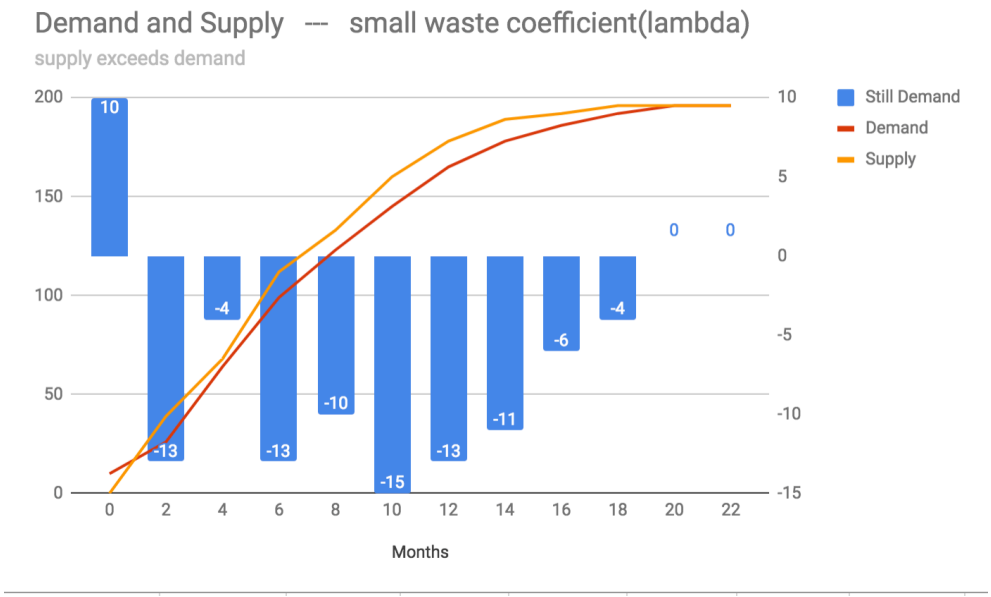
Basing on our research, it can be conclude that solution to supply-demand problem is the key factor to determine whether electrification transition will go smoothly. So, first and foremost, you have to reach a trade-off between charging station construction (electricity supply) and car purchase (electricity demand).

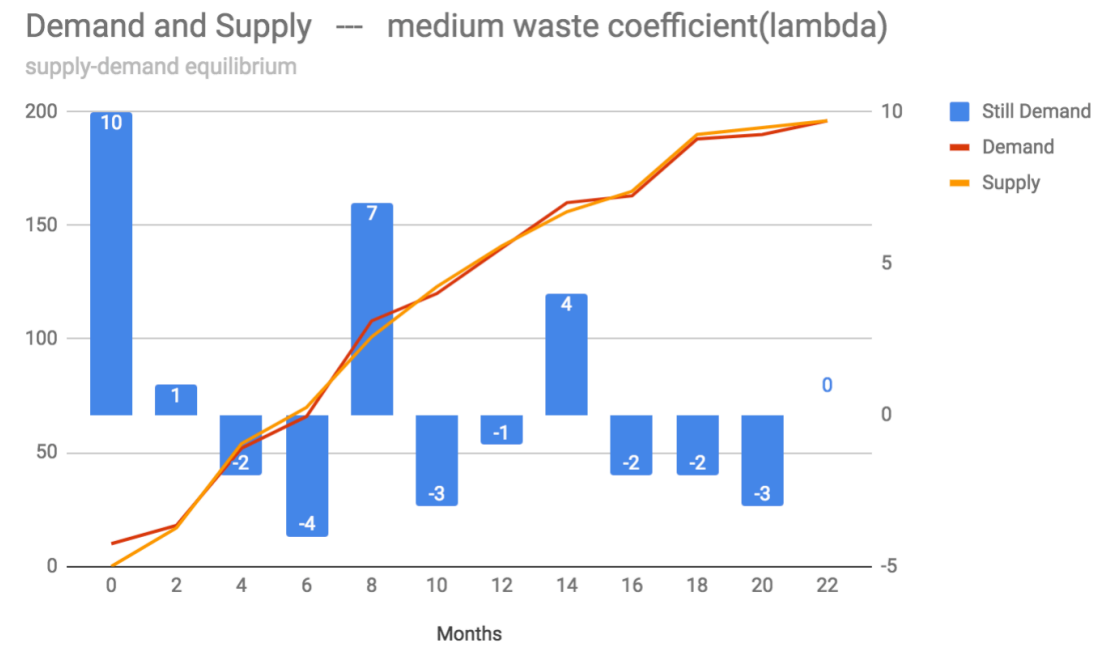
However, the solutions for different countries are likely to differ. Building chargers first or waiting for people purchasing more electrical cars. Essentially, it is a problem determined by a country's economic situation and its decision maker's willingness when it comes to promoting electrical cars. Establishing charging network is definitely fast and effective, while you have to suffer a long period of idle charger and resources waste.

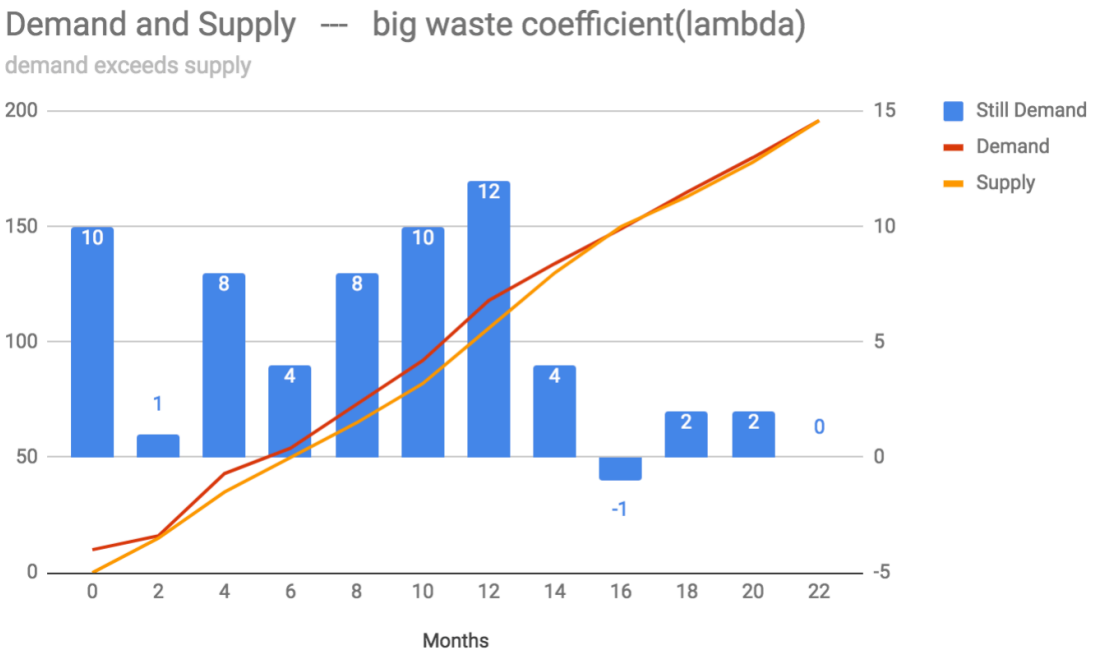
South Korea is a prosperous country and has strong driving force for green environmental protection. Therefore, we think it belongs to the former development mode, that is to set up charging to drive people's desire to purchase electric cars.

Here we think the key factors that affect the plan for building a charging station are the country's financial affluence and the desire to drive electric vehicles.

We define the quantized value in the model as λ (waste coefficient), the smaller the value, the richer the national capital, and the stronger the desire to drive electric vehicles.







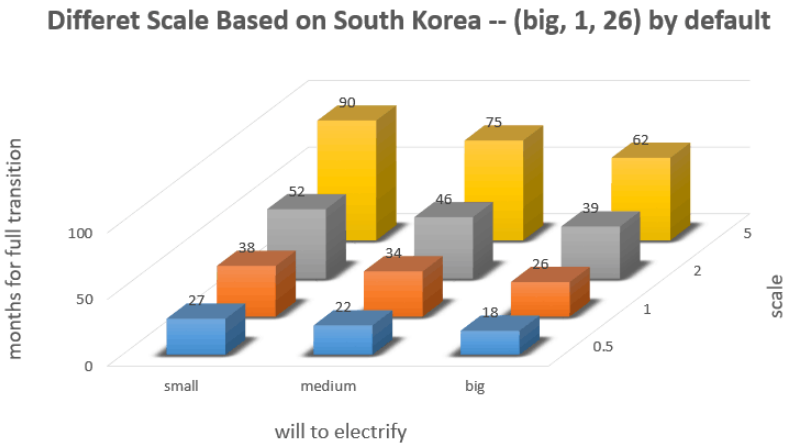
Task 2C:

The timeline we propose for the full evolution to electric vehicles in Korea:

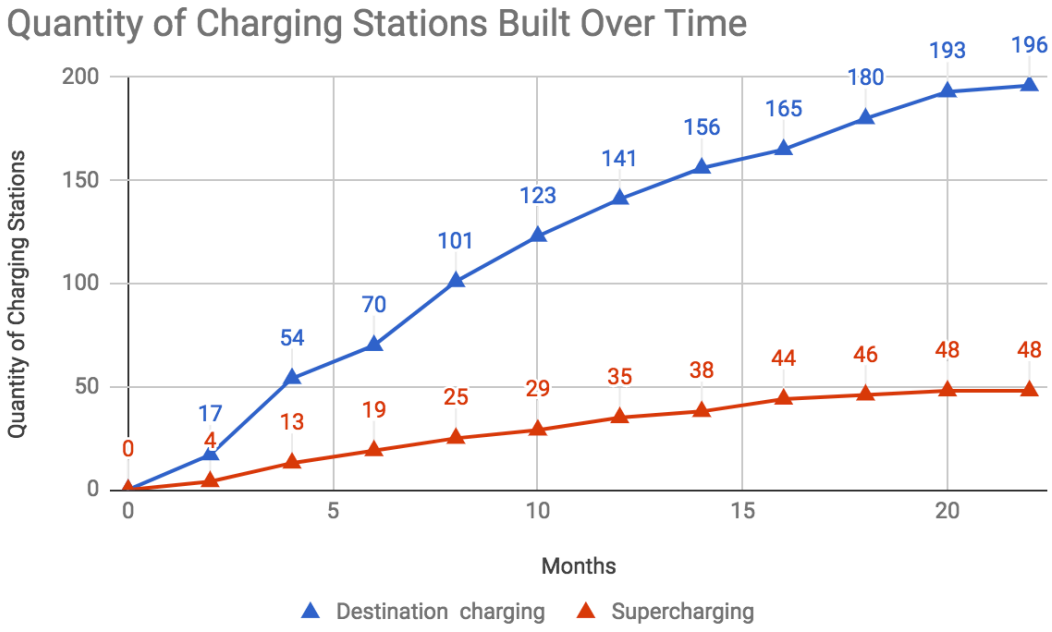
Destination charging stations	0	159	350	514	717	870
Supercharging stations	0	45	92	145	215	263
Months	12	14	16	18	20	22
Destination charging stations	1009	1104	1240	1345	1453	1523
Supercharging stations	303	331	369	413	440	463

For details:

percentage	months
10%	2
30%	6
50%	10
70%	16
100%	26



scale-axis means the multiple of South Korea in population and car parc



The key factors that shaped our proposed growth plan timeline:

1. The population size of the country as a whole
2. The country's capital affluence and the desire to drive electric vehicles

How to affect it? See the following diagram in detail:

We select North Chungcheong province in South Korea, and observe the development of the chargers distribution. (The progresses are repectively 0%, 10%, 30%, 50%, 70%, 100%)

3.4 Result of Model III

The pattern of development we have drawn from Korea before does not apply to countries with vastly different geographic distributions, wealth distributions, population distributions and policies.

Regarding the factors that triggered different growth networks, we think the major ones are: the country's population and total number of cars, the average complexity of the terrain, national engineering capacity, the state's financial well-being and its desire to motivate electric vehicles.

National Population and Total Car Numbers: determine the number of chargers that countries need to eventually build.

The complexity of the terrain: In our model, the more complex the terrain, the less the ability of the charging station and the people to exert influence on each other national engineering capacity : determine the country in each time period can at most pedicure charger station numbers.

The State's Funding Situation and the Enthusiasm for Driving Electric Vehicles: The Larger Value Shows More Hopes Made by More Constructing Chargers to Drive Up the Need to Replace Electric Vehicles, And Not At The Same Time Regardless of the Waste Caused by Freezer Charges

We can use the country's population and total vehicle volume in Model1 to calculate the total number of charger numbers that the country needs to convert all cars to electric vehicles. Based on the country's engineering capacity, we can calculate the country's shortest time needed to complete construction, but the quickest solution is not necessarily suitable for the country's national conditions.
We define this shortest time as T_fast.

In our Model 2, we can get the optimal solution for all chargers and the optimal time in our country through the country's financial affluence and the desire to promote the motive power and terrain complexity of electric vehicles.
We define this optimal time as T_optimal.

We define u: u = T_optimal / T_fast

Then we define different development patterns based on the value of u:
u > 1.8: Slow development mode, build the chargers first and hope people buy the cars
1.3 < u <= 1.8: Balance development model
1.0 <= u <= 1.3: Fast development model, build chargers inresponse to car purchases

We have classified some countries according to our model and the results are as follows:

slow development mode: Brazil, Thailand, Vietnam, Mexico, Cuba
balanced development model: China, South Korea, Italy, Russia
fast development model: Canada, Germany, Australia, United Kingdom, United States

4 Evaluation

4.1 Strengths

➤When dealing with task 1, it's convenient for us to adapt to different circumstances, including urban areas and rural areas, by simulating the actual location of passengers and charging stations. The distribution, thus, can be obtained.

➤To solve for the answer of model 2 in Task 2, it's a waste of computation resources to find the global optimal solution for a NP problem. Algorithm as local search, under this circumstance, is an efficient way to acquire approximate optimal solution.

➤The calculations of the relationship between suppliers and demanders are done with matrix. Thus, model 2 has strong fault tolerance.

4.2 Weaknesses

➤When dealing with task 1, we assume that the users' demand for fast charger/ destination charger is a constant, this may not be consistent with reality.

➤When dealing with task 2B, the two constant parameters for two types of suppliers is empirically defined as 200 and 400 respectively, which may be vulnerable when drastic change occurs to variable.

➤When dealing with task 2B, the ability for demander and supplier to interact is defined by matrix. When we construct model for building charging stations, we do not consider the impact from traditional petroleum industries.

4.3 Future Work (For Task 4)

All the changes will lead to the decrease of demand for chargers. The charges in our network will become sporadic.

5 Reference

[1] Aardal, K., van den Berg, P., Gijswijt, D. and Li, S. (2015). Approximation algorithms for hard capacitated k-facility location problems. *European Journal of Operational Research*, 242(2), pp.358-368.

[2] G. E. P. Box and Mervin E. Muller, *A Note on the Generation of Random Normal Deviates*, The Annals of Mathematical Statistics (1958), Vol. 29, No. 2 pp. 610–611

[a] https://en.wikipedia.org/wiki/List_of_United_States_urban_areas

[b] https://www.opendatanetwork.com/

[c] http://www.governing.com/gov-data/car-ownership-numbers-of-vehicles-by-city-map.html

[d] https://en.wikipedia.org/wiki/Box–Muller_transform

[e] https://www.wired.com/story/tesla-model-3-delivery-timeline/

[f] http://money.cnn.com/2017/09/11/news/china-gas-electric-car-ban/index.html

[g] http://www.govtech.com/fs/Building-Out-Electric-Vehicle-Infrastructure-Where-Are-the-Best-Locations-for-Charging-Stations.html

[h] https://www.tesla.com/destination-charging

[i] https://www.tesla.com/supercharger

[j] https://geology.com/cities-map/new-york.shtml

6 Appendix

[I]metro_data.tsv: The data of metropolitans in U.S. for Task 1

[II]korea.tsv: The data of regions in South Korea for Task 2

[III]task_one_US_charging_stations.csv: The result of Task 1

[IV]task_two_South_Korea_charing_stations.csv: The result of Task 2

```
##The program is for The Local Search Algorithm
import numpy as np

# define constants
FAST_SPREAD_DIS = 150
SLOW_SPREAD_DIS = 60
MAX_EXECUTE_NUM = 8
alpha = 0.05
constant = 5
origin = (0, 0)
TERRAIN_SCORE = 1

eh = 0.1
ei = 0.1
eg = 1
emax = 5
ALPHA = 1.0
NATATION_SCALE = 8.0
SCALE_WEIGHT = 1.0
timer = 0

# load data
PREFIX = 'Gangwon_'
fast_stations = np.load(PREFIX + 'station_slow.npy')
slow_stations = np.load(PREFIX + 'station_quick.npy')
pops = np.load(PREFIX + 'people.npy')

# get some info from data
FAST_STATION_NUM = fast_stations.shape[0]
SLOW_STATION_NUM = slow_stations.shape[0]
ALL_STATION_NUM = FAST_STATION_NUM + SLOW_STATION_NUM
POPS_NUM = pops.shape[0]
FAST_CAPAICTY = fast_stations[0, 2]
SLOW_CAPAICTY = slow_stations[0, 2]

st_index_dict = {}
pops_index_dict = {}

for i in range(SLOW_STATION_NUM):
    st_index_dict[i] = ((slow_stations[i, 0], slow_stations[i, 1]), 0)

for i in range(FAST_STATION_NUM):
    st_index_dict[i+SLOW_STATION_NUM] = ((fast_stations[i, 0], fast_stations[i, 1]), 1)

for i in range(POPS_NUM):
    pops_index_dict[i] = ((pops[i, 0], pops[i, 1]), pops[i, 2])

# function to compute Euclid distance
def l2dis(p1, p2):
    dis = (p1[0] - p2[0]) ** 2 + (p1[1] - p2[1]) ** 2
    return dis
# compute all distances in all charging stations and pops
dis_cache = np.zeros(ALL_STATION_NUM, POPS_NUM)

for i in range(ALL_STATION_NUM):
    for j in range(POPS_NUM):
        dis_cache[i, j] = l2dis(st_index_dict[i][0], pops_index_dict[i][0])

J = np.zeros((ALL_STATION_NUM, POPS_NUM))
V = np.zeors((1, ALL_STATION_NUM))
N = np.zeros((1, ALL_STATION_NUM))
```

```
# compute J value for fast charging
def flast_spread(distance, terrain=TERRAIN_SCORE):
    return distance ** 1.5 / (200.0 * TERRAIN_SCORE) + 1
# compute J value for destination charging
def slow_spread(distance, terrain=TERRAIN_SCORE):
    return distance ** 2 / (400.0 * TERRAIN_SCORE) + 1
# make up for step 9
def scale(n):
    return n * SCALE_WEIGHT / NATATION_SCALE * (n - NATATION_SCALE) ** 2 + 1

for i in range(ALL_STATION_NUM):
    if i < SLOW_STATION_NUM:
        for j in range(POPS_NUM):
            J[i, j] = slow_spread(dis_cache[i, j])
    else:
        for j in range(POPS_NUM):
            J[i, j] = flast_spread(dis_cache[i, j])

for i in range(ALL_STATION_NUM):
    if i < SLOW_STATION_NUM:
        V[0, i] = SLOW_CAPAICTY
    else:
        V[0, i] = FAST_CAPAICTY

while (np.sum(N) < ALL_STATION_NUM):
    # H: 1 * n
    H = N.dot(J) * eh + ei * timer * np.ones(1, POPS_NUM) + eg * np.ones(1, POPS_NUM)
    H[H > emax] = emax
    D = H - V.dot(J)
    S = J.dot(D.T)
    S[S < 0] = 0

    for i in range(ALL_STATION_NUM):
        S[0, i] = min(S[0, i], V[0, i])

    W = ALPHA * (V - S)
    X = S - W
    M = 1 * np.ones(1, ALL_STATION_NUM) - N
    A = np.zeros(1, ALL_STATION_NUM)
    # get the result of sort
    sort_index = np.argsort(-X)
    res = X[sort_index]
    # choose the biggest i numbers
    best_plan = None
    best_score = -0x7FFFFFFF
    # find optimal plans
    for i in range(1, ALL_STATION_NUM - np.sum(N) + 1):
        score = np.sum(res[:i]) + scale(i)
        if score > best_score:
            best_score = score
            best_plan = i

    N[sort_index[:i]] = 1
    print 'timer: ', timer, 'Choose: ', best_plan

    timer += 1
    # update vector N
    N = A + N
```

```
# encoding=utf-8
"""
model1: The problem of location of hard capacity facilities

Author: daisenryaku@163.com
Date: 2018-2-12
"""

import numpy as np
from pulp import *
import matplotlib
import matplotlib.pyplot as plt


class Dataset(object):
    def __init__(self, urban_name, population, land_area, gdp, car_avreage, quick=False):
        """
        Init the dataset class
        Args
          - urban_name: the name of the metro
          - land_area: the area of the metro
          - population: the population of the metro
          - gdp: the per capita GDP of the metro
        """
        self.urban_name = urban_name
        self.land_area = land_area
        self.population = population
        self.gdp_max = 38786
        self.gdp = gdp / self.gdp_max  # value range (0, 1)
        self.car_average = car_avreage
        self.quick = quick

        # the area of the metro -> the radius of the corresponding circle, (km)
        self.r = int(np.math.floor(np.sqrt(self.land_area / np.pi)))
        # get the site of the charging station
        self.charging_station = self.station_transform()
        # Obtaining the proportions of rechargeable piles
        self.station_scale = self.land_area * 0.05
        self.station_choice = []
        # Charging pile capacity, fast and slow filling the corresponding capacity / demand
        if self.quick:
            self.capacity = 0.02
            self.need = 0.1
        else:
            self.capacity = 0.04
            self.need = 0.4
        # Obtaining customer distribution
        self.charging_people, self.pop_scale = self.pop_transform()
        # Gain the proportions of customer contraction
        self.pop_scale = np.ones((self.charging_people, 1)) * self.pop_scale
        # the (x, y) coordinate of customer
        self.population_random = np.zeros((self.charging_people, 2))
        self.population_random[0: int(0.1*self.charging_people) , :] =
np.random.multivariate_normal((29, -52), [[100, 0],[0, 100]], int(0.1 * self.charging_people))
        self.population_random[int(0.1*self.charging_people): int(0.2*self.charging_people) , :]
= np.random.multivariate_normal((12, -32), [[20, 0],[0, 20]], int(0.1 * self.charging_people))
        self.population_random[int(0.2*self.charging_people): int(0.3*self.charging_people) , :]
= np.random.multivariate_normal((15, -22), [[100, 0],[0, 100]], int(0.1 * self.charging_people))
        self.population_random[int(0.3*self.charging_people): int(0.7*self.charging_people) , :]
= np.random.multivariate_normal((0, 0), [[120, 0],[0, 120]], int(0.4 * self.charging_people))
        self.population_random[int(0.7*self.charging_people): int(0.8*self.charging_people) , :]
= np.random.multivariate_normal((-5, -28), [[50, 0],[0, 50]], int(0.1 * self.charging_people))
        self.population_random[int(0.8*self.charging_people): int(0.9*self.charging_people) , :]
= np.random.multivariate_normal((36, 38), [[160, 0],[0, 160]], int(0.1 * self.charging_people))
        self.population_random[int(0.9*self.charging_people): int(1.0*self.charging_people) , :]
= np.random.multivariate_normal((78, 42), [[100, 0],[0, 100]], int(0.1 * self.charging_people))
```

```
      print(self.urban_name, np.shape(self.charging_station),
np.shape(self.population_random))

    def station_transform(self):
        """
        Ensuring small computation (1~147)
        Return:
            - the (x, y) coordinate of charging station
        """
        split_base = self.r // 20+1 if self.r > 20 else 1
        return np.array([(x, y) for x in range(-self.r, self.r, split_base) for y in
range(-self.r, self.r, split_base) if (x ** 2 + y ** 2) <= self.r ** 2])

    def pop_transform(self):
        """
        Ensuring small computation (~100)
        Return:
            - the (x, y) coordinate of customer
        """
        base_pop = int(self.population / 4 * self.car_average * 0.7 * self.gdp)
        pop_scale = 10**(len(str(base_pop))-3)
        return base_pop // pop_scale, pop_scale

    def draw(self):


        """
        Draw the position of the person, the position of the charging station on the map
        """
        f1 = plt.figure(1)
        plt.scatter(self.charging_station[:, 0], self.charging_station[:, 1], c='b',
marker='.')

        self.charging_station = self.charging_station[self.station_choice]
        plt.scatter(self.charging_station[:, 0], self.charging_station[:, 1], c='g',
marker='.')

        plt.scatter(self.population_random[:, 0], self.population_random[:, 1], c='r',
marker='X')
        plt.show()

    def solve(self, show=False):
        """
        Solving linear programming
        Args:
            - show (bool): Decide whether to print out the intermediate results
       Return:
            - unique_num (int): The amount of use of the charging station
        """
        prob = LpProblem("mcm2018", LpMinimize)
        # define the constant
        len_station, len_population = len(self.charging_station),
len(self.population_random)
        f = 10 / len_station
        c = np.ones((len_station, len_population))
        u = self.capacity
        for i in range(len_station):
            for j in range(len_population):
                x1, y1 = self.charging_station[i]
                x2, y2 = self.population_random[j]
```

```
                c[i][j] = np.sqrt((x1-x2)**2+(y1-y2)**2)

        # define the variable
        F = [str(id) for id in range(len(self.charging_station))]
        D = [str(id) for id in range(len(self.population_random))]
        y = LpVariable.dicts("station_choice", F, 0, 1, LpInteger)
        x = LpVariable.dicts("connect_choice", (F, D), 0, 1, 'Continuous')

        # add the object function
        prob += lpSum([f * y[i] + x[i][j] * c[int(i)][int(j)] for j in D for i in F])

        # add constraints
        for j in D:
            prob += lpSum([x[i][j] for i in F]) == self.need
        for i in F:
            prob += lpSum([x[i][j] for j in D]) <= u * y[i]

        # solve
        prob.solve()

        # show output
        if show:
            print("Status:", LpStatus[prob.status])
            print("Total Cost = ", value(prob.objective))

        # look the answer
        if show:
            for v in prob.variables():
                print(v.name, "=", v.varValue)

        self.station_choice = [bool(int(y[i].value())) for i in F]
        unique_num = np.count_nonzero(self.station_choice) * self.station_scale
        if show:
            print(unique_num)
        return unique_num

    def save(self):
        """
        save the data with the '.npy' format
        """
        if self.quick:
            station_name = self.urban_name + '_station_quick.npy'
        else:
            station_name = self.urban_name + '_station_slow.npy'

        self.charging_station = self.charging_station[self.station_choice]
        station_scale = np.ones((len(self.charging_station), 1)) * self.station_scale
        station_save = np.concatenate((self.charging_station, station_scale), axis=1)

        people_name = self.urban_name + '_people.npy'
        people = np.concatenate((self.population_random, self.pop_scale), axis=1)

        np.save(station_name, station_save)
        np.save(people_name, people)


if __name__ == '__main__':
    dataset = Dataset('North Chungcheong', 1588633, 7433, 32456, 0.509, quick=True)
    dataset.solve()
    dataset.save()
    dataset.draw()
```