Strategic EV Charger Network in Illinois

Tian Sun

December 2023

1 Abstract

Optimizing Electric Vehicle (EV) Charger Hub Locations in Illinois: A Comprehensive Analysis, presents a detailed examination of the strategic placement of EV charging stations using advanced methodologies. Employing advanced network analysis, spanning tree algorithms, and Steiner tree methodologies, the study integrates crucial real-world factors such as population density, urban layouts, and regional traffic flows, underscoring the importance of comprehensive, multi-faceted data in infrastructure planning. The findings not only illuminate the complex dynamics of charger hub optimization but also propose a scalable model for future EV infrastructure development, emphasizing the need for adaptive and data-driven approaches in the rapidly evolving landscape of electric mobility.

2 Introduction

2.1 Background

In the wake of a global shift towards sustainable transportation, the adoption of electric vehicles (EVs) has been rapidly increasing. This shift, while beneficial for the environment, presents a new set of challenges and opportunities, particularly in the realm of infrastructure development. The State of Illinois, with its diverse geography, bustling urban centers, and extensive road network, stands at the forefront of this transition. This project aims to address a critical aspect of this transition: the strategic placement of electric vehicle charger hubs across Illinois.

The goal is to develop an optimized network of charging stations that not only meets the current demand but is also scalable for future growth. This involves a multidimensional analysis encompassing geographic, demographic, economic, and technological factors. By leveraging geographical information systems (GIS), data analytics, and predictive modeling, this project seeks to identify key locations throughout Illinois where EV charging hubs would be

most effective and beneficial.

Through this project, we aim to create a blueprint for efficient, accessible, and sustainable EV infrastructure, setting a standard not just for Illinois, but for regions worldwide grappling with similar challenges in the era of electric mobility.

2.2 Dataset Description

The dataset is from the US Zip Codes Database conducted by simplemaps Interactive Maps & Data.[1]

The US Zip Codes Database provides a simple, accurate and up-to-date data of US Zip Codes. It's been built from the ground up using authoritative sources including the U.S. Postal Service, U.S. Census Bureau, National Weather Service, American Community Survey, and the IRS.

The data is updated from October 30, 2023. It has 41,690 unique zip codes including ZCTA, unique, military, and PO box zips. It includes attributes from latitude and longitude to household income. The GIS information is accurate as it is aggregated from official sources and precisely geocoded to latitude and longitude. Here is a brief view of the dataset.

	zip	lat	lng	city	state_id	state_name	zcta	parent_zcta	population	density	county_fips	county_name	county_weights	county_names_a
601	18.18027	-66.75266	Adjuntas	PR	Puerto Rico	True	NaN	17126.0	102.6	72001	Adjuntas	{"72001": 98.73, "72141": 1.27}	Adjuntas|Utuad
602	18.36075	-67.17541	Aguada	PR	Puerto Rico	True	NaN	37895.0	482.5	72003	Aguada	{"72003": 100}	Aguad
603	18.45744	-67.12225	Aguadilla	PR	Puerto Rico	True	NaN	49136.0	552.4	72005	Aguadilla	{"72005": 99.76, "72099": 0.24}	Aguadilla|Mo
606	18.16585	-66.93716	Maricao	PR	Puerto Rico	True	NaN	5751.0	50.1	72093	Maricao	{"72093": 82.27, "72153": 11.66, "72121": 6.06}	Maricao|Yauco|Sabar Gran
610	18.29110	-67.12243	Anasco	PR	Puerto Rico	True	NaN	26153.0	272.1	72011	Añasco	{"72011": 96.7, "72099": 2.81, "72083": 0.37, ...	Añasco|Moca|La Marias|Aguad

Figure 1: Dataset overview

2.3 Dataset Preprocessing

As this project only focus on the electric vehicles (EVs) in Illinois State, we will select observations with "state_id" matches "IL".

Next, as we only need GIS information about location of each zip code for calculate distance between counties in the future, we will select columns "zip," "lat," and "lng." For better understanding, we rename the columns as

[1] https://simplemaps.com/data/us-zips

"zip_code," "latitude," and "longitude."

To avoid problems in generating networks, we simply remove any columns with missing values and reset the index. Here is a brief view of the clean dataset.

	zip_code	latitude	longitude
0	60002	42.46740	-88.09419
1	60004	42.11206	-87.97895
2	60005	42.06518	-87.98676
3	60007	42.01431	-87.99828
4	60008	42.07390	-88.02290
...	...	...	...
1391	62995	37.41945	-88.88244
1392	62996	37.16624	-89.15984
1393	62997	37.98357	-89.59013
1394	62998	37.51649	-89.44698
1395	62999	37.89122	-89.05275

1396 rows × 3 columns

Figure 2: Clean dataset overview

3 Network Overview

3.1 Distance Calculation

The first step we need to do is generate a basic view of the Illinois network. As our data only contains zip code with the associated latitude and longitude, we include extra math to help us calculate the distance between each zip code, which represents the weight of the edges in the network system. The simpliest way to calucate distance between two points is the **Euclidean distance**.

3.1.1 Euclidean Distance

The Euclidean distance between two points in Euclidean space is the length of a line segment connecting them. In Cartesian coordinates, if $p = (p_1, p_2, ..., p_n)$

and $q = (q_1, q_2, ..., q_n)$ are two points in Euclidean n-space, the distance (d) between p and q is given by the Pythagorean formula:

$$d(p, q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

The Euclidean distance is the "ordinary" straight-line distance between two points in space and is used in various areas such as mathematics, physics, computer science, and engineering. The Euclidean distance formula is most appropriate for measuring distances in a flat, two-dimensional space. However, when it comes to calculating distances between locations on the Earth's surface, characterized by latitude and longitude, this approach is not suitable because the Earth is a three-dimensional sphere. Instead, we introduce **Haversine distance**.

3.1.2 Haversine Distance

The Haversine formula is used to calculate the distance between two points on the surface of a sphere, given their longitudes and latitudes. It's particularly useful for finding the distance between two points on the Earth, assuming it as a perfect sphere. The formula is:

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot \text{atan2}\left(\sqrt{a}, \sqrt{1-a}\right)$$

$$d = R \cdot c$$

Where:

- φ_1 and φ_2 are the latitudes of point 1 and point 2 (in radians),
- $\Delta\varphi$ is the difference in latitude between point 2 and point 1 (in radians),
- $\Delta\lambda$ is the difference in longitude between point 2 and point 1 (in radians),
- R is the radius of the Earth (mean radius = 6,371 km),
- d is the distance between the two points (along the surface of the sphere).

The Haversine formula accounts for the curvature of the Earth and is particularly important for navigation and geolocation applications where accuracy is crucial. While it assumes the Earth is a perfect sphere (which it is not, due to its slightly ellipsoidal shape), it still provides a close approximation for most practical purposes, which makes the distance more realistic in our project.

3.2 Network Visualization

With zip codes as vertices, and distance between zip codes as weight of edges, we can generate a network system for Illinois counties based on "networkx" package in Python. Here is a view of the Illinois graph. The shading part represents the edges between each zip code. For visualization purposes, we make the width of the edges and the size of the nodes pretty small, also remove all labels of zip code.

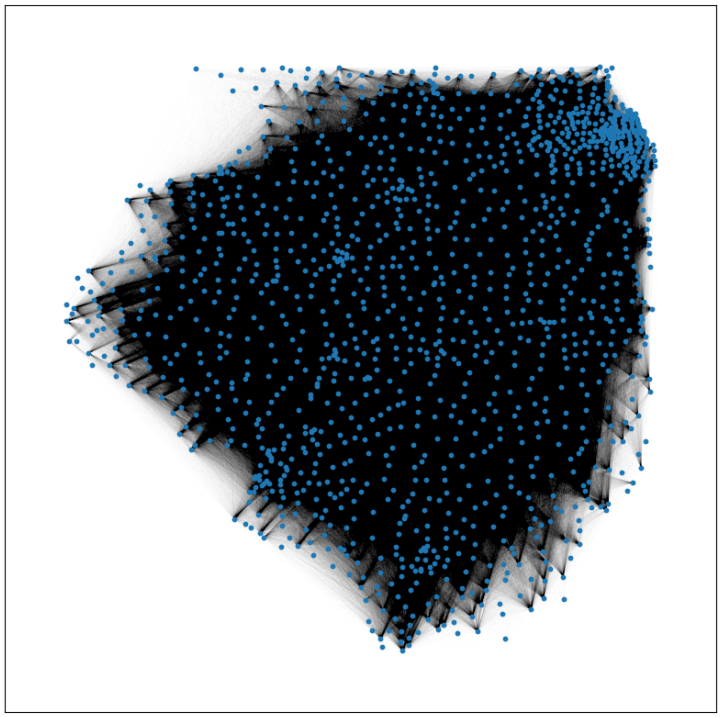


Figure 3: Illinois Graph

Note that there are a lot of shading part inside the graph as edges cross between each cycle. We should avoid that for our electric vehicles (EVs) problem as placing charger hubs in cycles will involves in waste in economic and efficiency. One general approach to avoid cycles inside the network tree graph is **Spanning tree**.

3.3 Spanning Tree

A spanning tree of a graph is a subgraph that includes all the vertices of the original graph and is a single connected tree. Essentially, a spanning tree connects all the vertices together with the minimum number of edges to ensure there is no cycle in the graph. This concept is particularly important in fields like computer networking, graph theory, and algorithm design. Key characteristics of a spanning tree are:

- **Connectivity:** It connects all the vertices of the graph together.
- **Minimum Number of Edges:** A spanning tree of a graph with N vertices has exactly $N - 1$ edges. Any more, and it would form a cycle; any fewer, and it wouldn't connect all the vertices.
- **No Cycles:** A tree is an acyclic graph, so a spanning tree cannot contain any cycles.
- **Subgraph:** A spanning tree is a subset of the graph it spans, containing all the vertices but only some of the edges.
- **Multiple Spanning Trees:** A graph can have multiple spanning trees. The number depends on its structure.

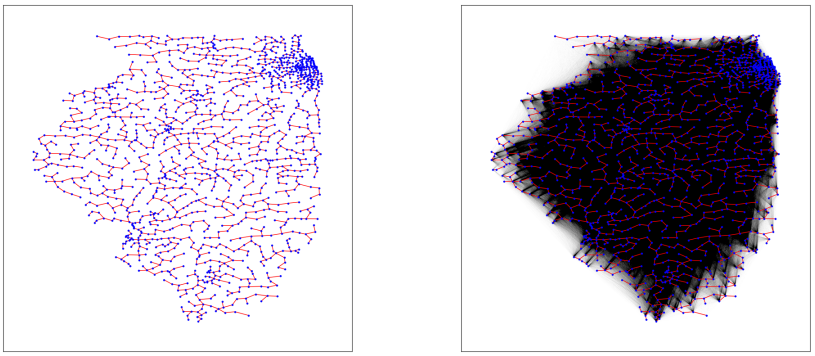
There are various algorithms to find a spanning tree in a graph, such as:

1. **Kruskal's Algorithm:** This algorithm builds the spanning tree by adding the shortest edge of the graph that doesn't form a cycle, repeating this until the spanning tree is complete.
2. **Prim's Algorithm:** This algorithm builds the spanning tree by starting from an arbitrary vertex and repeatedly adding the shortest edge that connects a vertex in the tree to a vertex outside the tree.

In our project, we will use a pre-defined function "minimum_spanning_tree" from "networkx" do calculate the spanning tree for the Illinois graph. By default, it will use the "Kruskal's Algorithm."

3.4 Spanning Tree Visualization

We are now able to make plots for spanning tree of Illinois graph. Here is a plot for vertices with only spanning tree. For comparison, we add an extra plot with spanning tree in red edge above the previous general graph.



(a) Illinois Spanning Tree Graph (b) Illinois Spanning Tree above Graph

Figure 4: Spanning Tree

We will continue our strategy based on this spanning tree.

4 Strategy Development

4.1 Randomly Selected k Vertices

Continue with our electric vehicles (EVs) problem, it will be much easier for us to solve by considering it as a mathematical problem than general. Assume that we want to locate k electric vehicle charging hubs on vertices in our transportation network $G(V, E)$. The objective is to find the location of these hubs (vertices) as it could accommodate most electric vehicle owners with least cost.

One possible solution is to follow the spanning tree we discuss previously:

1. Connect all vertices in the fastest way possible the electric charging station hubs (forming an "electric tree".) We can achieve this by identify a minimum cost spanning tree in the whole network. This guarantees each vertices can reach an electric hub (eventually).

2. Identify the shortest path from each other node (non-hub node) to any node in the tree. We can calculate the average time it takes every node to get to the closest electric hub using only the edges in the tree to check the cost.

Base on this strategy, let's first try randomly select $k = 10$ hubs to locate inside the spanning tree we generated and proceed with this calculation:

1. For each node in the minimum spanning tree, find the shortest distance to the closest among the 10 randomly selected nodes.
2. Sum these shortest distances for all nodes.
3. Calculate the average of these sums.

By perform these steps with the data, we get the result as:

- The sum of the shortest distances from each node to its nearest among the $k = 10$ randomly selected nodes is approximately **113,645.17** miles.
- The average of these shortest distances, calculated across all nodes in the minimum spanning tree, is approximately **81.41** miles.

This means that on average, each node in our network is about **81.41** miles away from its closest selected node among the $k = 10$ random selections. Note that this result will change for each different random selected. This inspires us to a different thought: if it is possible to consider it as an optimization problem and our objective is to find the best location $k = 10$ hubs.

4.2 Optimization Problem

The first step we do optimization problem is to find the parameters. Let's thinking about our electric vehicles (EVs) problem in another way. We'll treat it as a variant of the facility location problem. In this scenario, the selected vertices can be thought of as "facilities," and the goal is to minimize the total distance from each vertices to its nearest facility. Here is a formulation of the problem:

- **Decision Variables:** For each node i, introduce a binary variable x_i that is 1 if node i is chosen as one of the facilities and 0 otherwise. Additionally, for each node i and potential facility j, introduce a binary variable y_{ij} that is 1 if node i is assigned to facility j.
- **Objective Function:** Minimize the sum of the distances from each node to its assigned facility. This is the sum of $d_{ij} \times y_{ij}$ over all nodes i and facilities j, where d_{ij} is the distance from node i to node j.
- **Constraints:**
    1. **Facility Selection Constraint:** Exactly 10 facilities must be selected. This is the sum of $x_j = 10$.
    2. **Assignment Constraints:** Each node i must be assigned to exactly one facility that is open. This means $\sum_j y_{ij} = 1$ for all nodes i, and $y_{ij} \leq x_j$ to ensure a node is only assigned to an open facility.

We can solve this optimization problem with "gurobipy" package in Python. By perform these steps with the data, we get the result as:

- The optimized $k = 10$ vertices for electric charger hubs are **60104, 61108, 61448, 61541, 61832, 62018, 62447, 62684, 62878, 62918**.
- The sum of the shortest distances from each node to its nearest among the $k = 10$ randomly selected nodes is approximately **73,688.76** miles.
- The average of these shortest distances, calculated across all nodes in the minimum spanning tree, is approximately **52.79** miles.

Notice that the optimization result we get is far **SMALLER** than the result from randomly selected, means economic and efficient. Here is graph of the location of the optimized hubs based on the previous spanning tree plot.

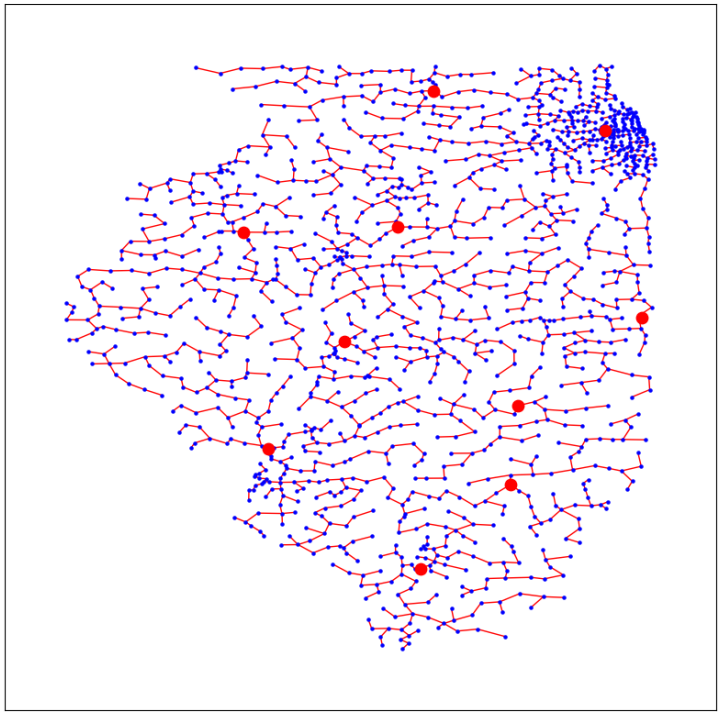


Figure 5: Illinois Optimized Location Graph

4.3 More Realistic Problem

Although optimization result is very good, we have to consider the problem realistically. Our previous work based on the minimum spanning tree is a little too strict as we assumes that all vertices are equally important, which is not true in real life. For a general map, there are big cities which have more people and more use of electric vehicles (EVs). We have to consider this assumption and tend to place more hubs to these cities, as they will definitely be used a lot.

According to the Illinois Zip Codes by Population data from Illinois Demographics BY CUBIT,[2] we select top $k = 10$ zip codes which has the highest population in Illinois. Here is a view of this list.

Illinois Zip Codes by Population

Below are 1,387 Illinois zip codes sorted by population from largest to smallest. The population data are from the 2022 American Community Survey.

Rank	Zip Code	Population
1	60629	108,997
2	60618	92,235
3	60632	86,372
4	60647	85,685
5	60639	85,248
6	60804	84,189
7	60617	83,188
8	60608	82,749
9	60625	80,296
10	60623	76,611

Figure 6: Illinois Zip Codes by Population

With this list, we will introduce another method from minimum spanning tree, the **Steiner Tree**.

[2] https://www.illinois-demographics.com/zip_codes_by_population

4.4 Steiner Tree

A Steiner tree, named after the Swiss mathematician Jakob Steiner, is a concept in graph theory that extends the idea of a minimum spanning tree. In a Steiner tree problem, given a graph and a subset of vertices (called Steiner points or terminals), the goal is to find the minimum length tree that connects all the terminals. Unlike a minimum spanning tree, which connects all vertices in the graph, a Steiner tree only needs to connect the specified subset of vertices, and it can include other vertices not in this subset if doing so results in a tree of lesser total length. Key aspects of a Steiner tree include:

- **Steiner Points:** In addition to the given set of terminals, a Steiner tree can include additional vertices from the graph. These additional vertices are called Steiner points.
- **Minimum Total Length:** The goal is to minimize the total length of the edges in the tree.

The Steiner tree problem is known to be NP-hard, which means that there is no known algorithm that can solve all instances of this problem efficiently (in polynomial time). In this project, we will use a Heuristic algorithm due to Kou et al. (1981):[3]

1. Generate a complete graph $G_1(N_1, A_1)$ from G where $N_1 = S$ and A_1 are arcs connecting every pair of nodes in S. The cost of every arc is equal to the shortest path distance between any two nodes in S in G.
2. Find a minimum spanning tree T of G_1.
3. Replace every tree arc (i, j) in T with the actual shortest path between i and j in G. Call this new graph G_T.
4. Find a minimum spanning tree T_1 of G_T.
5. Construct a Steiner tree T_S by dropping unnecessary arcs from T_1.

[3] L. Kou, G. Markowsky, and L. Berman, A fast algorithm for Steiner trees, Acta Inform., 15 (1981), 141–145.

Based on this algorithm, we successfully generated the Steiner tree with the $k = 10$ nodes. Here is a view of the Steiner tree graph with label as zip code.

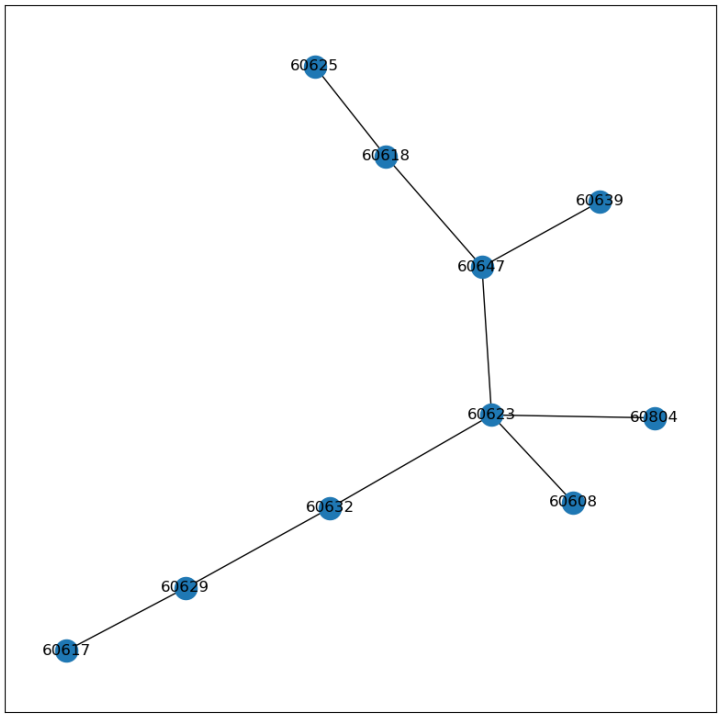


Figure 7: Illinois Steiner Tree Graph

4.5 Solve Problem with Steiner Tree

Now we can solve this problem based on our Steiner tree. Our algorithm is, for any node i that is not in the Steiner tree S, we can calculate its distance to a charger as:

1. Calculate the shortest path using the full network from i to j, where j is in S.
2. Calculate the shortest path using the Steiner tree from j to the nearest charger (which is definitely in the Steiner tree, by construction).

By perform these steps with the data, we get the result as:

- The top $k = 10$ vertices with highest population for electric charger hubs in the Steiner tree are **60629, 60618, 60632, 60647, 60639, 60804, 60617, 60608, 60625, 60623**.
- The sum of the shortest distances from each node to the Steiner tree is approximately **186,616.84** miles.
- The sum of the shortest distances from each node inside the Steiner tree to the selected electric charger hubs is **0**.
- The average of these shortest distances, calculated across all nodes in the minimum spanning tree, is approximately **134.64** miles.

Interesting finding is that the result we get for realistic life focus on population is even **worse** than the random selection. This might because the top $k = 10$ vertices we select is too close. As we can see in the Steiner tree graph, the 10 vertices with the edge between them is the exact Steiner tree we get, which is not a general case. With more research of these selections, we find that nine of these ten zip codes are in Chicago, the most important city in Illinois. This results that our plan to place electric charger hubs based on population looks bad. There should at least some extra hubs outside Chicago. Future works might need to consider increase the amount of k to involve zip codes outside the gather location set. Also we could try solve the problem under the assumption that these top $k = 10$ electric charger hubs are selected ahead and do optimization problems.

5 Conclusion

This project explored various methodologies for optimizing the placement of electric vehicle (EV) charger hubs in Illinois. Through the application of network analysis, spanning tree algorithms, and Steiner tree methodologies, the study revealed intricate dynamics in optimizing charger hub locations. The initial approach focused on minimizing distances within a network, but the incorporation of real-world variables like population density and city layouts indicated the need for a more nuanced strategy. However, our study also uncovered the necessity of incorporating more comprehensive data into future work. Future studies should integrate a broader range of factors, including detailed geographic features, evolving demographic trends, economic considerations, and emerging technological advancements in EV infrastructure. By doing so, future research can provide even more precise and sustainable solutions for the development of EV charging networks, not only in Illinois but in other regions facing similar challenges. This will ensure the deployment of EV charger hubs is not only efficient but also equitable and adaptable to the rapidly changing landscape of electric mobility.