Impact of Electric Vehicle Charging Station Placement on Traffic Flows

Brendon Gu, Emilia French, Alden Pritchard
Carnegie Mellon University
Pittsburgh, PA 15213
bkgu@andrew.cmu.edu, efrench@andrew.cmu.edu,
atpritch@andrew.cmu.edu

December 19, 2020

1 Introduction

One necessity that accompanies the increasing popularity of electric vehicles is the need for adequate charging infrastructure. The problem of optimally placing charging stations is an interesting and challenging one. Compared to traditional vehicles, electric vehicles have a much shorter range, making well-placed charging stations much more important. A strategic allocation of the charging infrastructure may not only reduce range anxiety for electric vehicle owners, but also minimize the initial cost of installment of new charging stations, as well as relieve the load of the electrical power system. [1]

In this paper, we first consider representative sets of routes in the Sioux Falls transportation network, both by generating weighted random routes and by calculating shortest paths between all nodes. Given a set of routes, we formulate the optimal placement of charging stations as a set cover problem to ensure that there is a charging station along any sufficiently long route.

We also extend our analysis to consider the effects of charging station placements. In particular, we consider how the placement of stations affects traffic flows, and how we may be able to place stations to influence these flows in a way that we want. In this paper, we mainly concern ourselves with trying to place stations 'optimally' - that is, to place them such that in the long run, traffic flows roughly evenly through each node.

To represent traffic flows, we use transition probabilities between nodes to calculate stationary distributions that model long-run driving behavior in the network. We account for potential changes in the transition probabilities after adding stations and then recalculate the stationary distribution to estimate the new traffic flow.

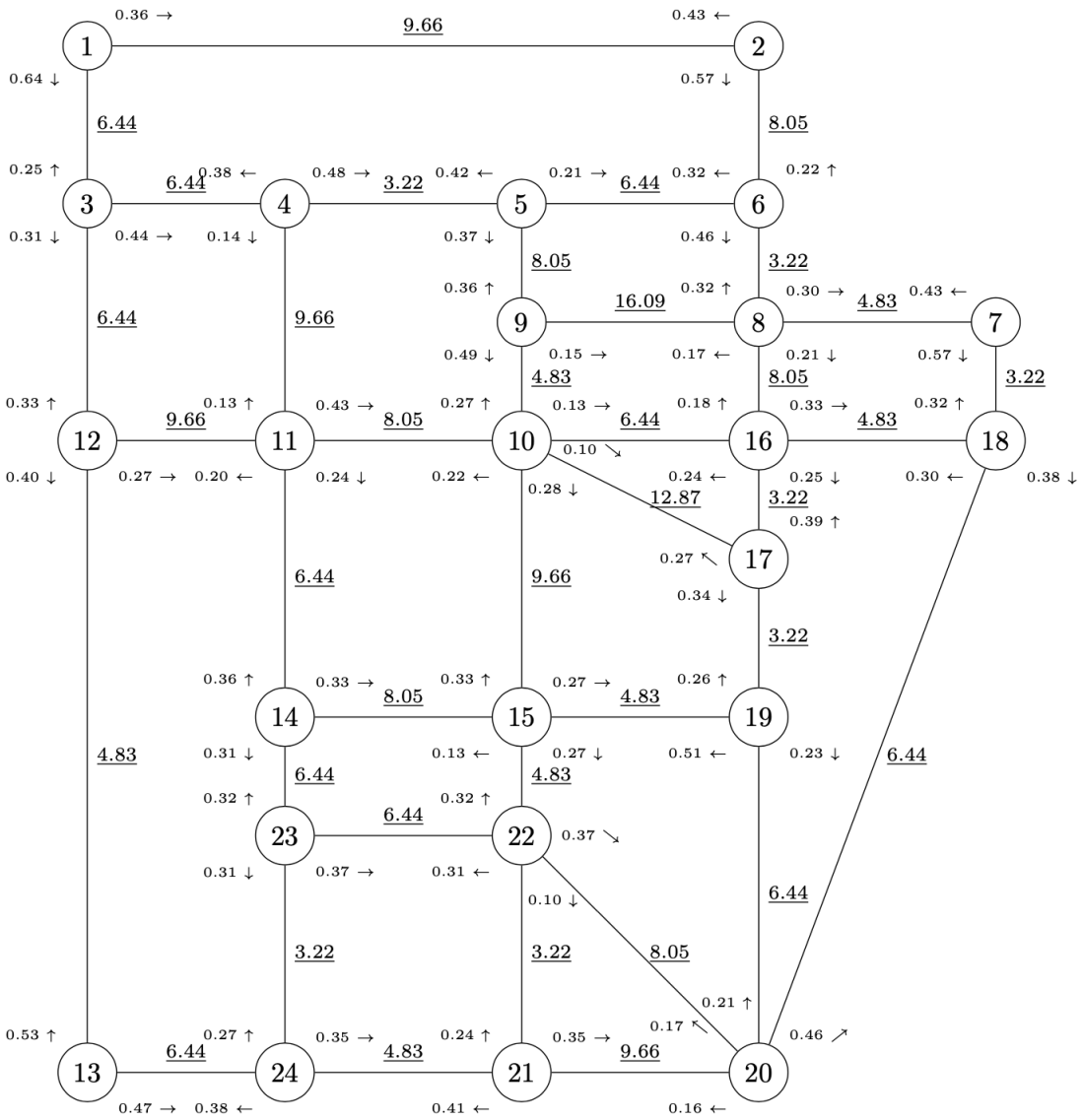
We discuss greedy and randomized methods, as well as heuristics, for choosing the placement of charging stations and compare their performance on a variety of criteria. Results are described in the Results section.

Finally, we mention areas we would have liked to pursue further, adjustments we made through the process of the project, and additional remarks in the Discussion section.

2 Methods & Formulation

We describe a transportation network by a set of nodes N, transition probabilities P, and path weights W. We consider an assignment of charging stations to be a binary assignment to variables x_i, where $x_i = 1$ if a charging station is placed at node $i \in N$, and $x_i = 0$ otherwise.

In our testing, we focused on the Sioux Falls transportation network [2], a network used for network design problems. The network has 24 nodes and 76 links.



Set Cover Formulation

Given a set of routes, we formulated the optimal placement of charging stations as a set cover problem, in which there must be a charging station along any route longer than a certain length.

The Gurobi code for the optimization returns a dictionary mapping each node to either 0 or 1, corresponding to whether there is a station at that node in the optimal assignment. It can be found in `optimization.py`.

Adding a Charging Station

Suppose we want to add a charging station at a vertex v which has neighbors $u_1, ..., u_n$. We assume that adding a station increases the probability of visiting v from one of its neighbors as this simulates a vehicle making a slight detour from its original course to go recharge its battery at a nearby station. In our case, we simulate this by scaling the transition probability from u_i to v by a factor of α (in our experiments we used $\alpha = 1.25$ and $\alpha = 1.5$).

Let P_{u_iv} be the probability of transitioning from u_i to v, and $P_{u_iN(u_i)}$ the probability of transitioning from u_i to one of its neighbors that is not v.

We know we must maintain $P_{u_iN(u_i)} + P_{u_iv} = 1$ to preserve the validity of our transition matrix.

So we find that after scaling we must have $\alpha \cdot P_{u_iv} + \beta \cdot P_{u_iN(u_i)} = 1$.

But we know that $P_{u_iN(u_i)} = 1 - P_{u_iv}$.

So we can rewrite the above as $\alpha P_{u_iv} + \beta(1 - P_{u_iv}) = 1$
$\Rightarrow \beta = \frac{1-\alpha P_{u_iv}}{1-P_{u_iv}}$, which we use in our code to observe how the transition matrix changes when we add a charging station.

Experiments

We used a combination of graph-theoretical and probabilistic tools to analyze the impacts of different charging station placements. The goal of these methods is to allow us to see how the placements affect both (a) traffic flows for people traveling a smaller set of routes, and (b) traffic flows for a broader population of travelers.

We used five different pairs of sinks and sources, and then for each pairing computed our graph-theoretical and probabilistic impact measures for nine different charging station placements. Three of our placements were driven by greedy algorithms, another three came from heuristic ideas, and the last three were derived randomly. We also decided to place four charging stations in the network as we felt this would provide sufficient coverage without over-saturating the network and was justified by our solutions to the prior set cover formulation. These are also defined in `experiments.py`. Note that in the code all vertices are specified as vertex - 1 since Python indexes from 0.

Source - Sink Pairings:
(2, 7, 18) - (13)
(20, 21) - (1, 3)
(6, 7) - (12, 13)

(17, 19) - (3, 4)
(12, 13) - (7, 18)

Charging Station Placements:
Random1: a random selection of four vertices in the graph
Random2: a random selection of four vertices in the graph
Random3: a random selection of four vertices in the graph
Greedy1: pick four vertices from the outermost edges of the graph, here we chose 1, 8, 20, and 13.
Greedy2: imagine an $s-t$ cut on the pair of vertices farthest apart in the graph (in this case 2 & 24), pick a spot in the graph where we can draw a straight line through the graph and make this cut using exactly four edges. Then the four of these eight endpoints closest to the middle are chosen to become charging stations. In this case this algorithm chooses 1, 5, 8, and 18 as stations.
Greedy3: choose any set of four vertices where each vertex is at least 3 edges away from another chosen vertex. Here we chose 4, 8, 19, and 24.
Heuristic1: chose four vertices each 1 edge away from a vertex on the outer layer of the graph. We chose 4, 23, 8, and 20.
Heuristic2: imagine an $s-t$ cut on the pair of vertices farthest apart in the graph (in this case 2 & 24), then pick four that are roughly equidistant from both s and t. In this case we choose 9, 11, 15, and 16.
Heuristic3: any set of four vertices where each vertex is at least two edges away from another chosen vertex while avoiding the outer layer of vertices. In this graph we chose 5, 11, 17, and 22 as stations.

Note that some of the 'magic numbers' here are chosen arbitrarily to fit this graph, and in the case of a different sized or connected network or different number of charging stations, these numbers would need to be adapted.

Probabilistic Measurements
Modeling the network as a first-order time-homogeneous Markov Chain, we release a large number of simulated travelers into the graph at one of our sources and let them run for some random number of steps. We then filter to only include paths ending at a sink, and then compute the resulting percentage of visits each edge receives to measure how frequently each edge was visited by travelers heading from a source to a sink. Using these percentages, we computed the L_2 norm distance between vectors containing percentages for each edge both before and after adding all of the charging stations. This gave us a measure of how much adding the stations disrupted traffic flows. We also computed the maximum visitation percentage as to record whether adding charging stations increased or decreased congestion at the busiest routes in the network. Our hope was to establish whether we could significantly alter traffic flows to ease congestion at busier edges in the network, hopefully smoothing traffic distributions and reducing overall congestion in the network. The code for these procedures is found in `starter_code.py`.

Graph Theoretical Measurements
We used a modified version of Dijkstra's Algorithm [3] to compute the shortest $s-t$ path weight and total length under the condition that the shortest path goes through a charging station. The code for this procedure is found in `all_paths.py`. This allows us to compare

how charging station placement affects the cost of traveling from a source to a sink, given that each path must go through at least one charging station.

3 Results

We first present results for our set cover code. We can find optimal placements when considering all shortest paths in the network with at least some given length, or at least some given weight.

Optimal placements in the Sioux Falls network:

	Nodes	Number of Stations
Length > 2	Nodes 3, 6, 8, 10, 16, 18, 19, 22, 24	9
Length > 3	Nodes 3, 6, 10, 14, 18, 19, 24	7
Length > 4	Nodes 3, 8, 11, 16, 22, 24	6
Length > 5	Nodes 4, 8, 15, 24	4
Length > 6	Nodes 12, 17, 18	3
Length > 7	Node 19	1

	Nodes	Number of Stations
Weight > 5	Nodes 2, 3, 5, 8, 11, 15, 16, 20, 23, 24	10
Weight > 10	Nodes 3, 6, 9, 11, 16, 19, 20, 22, 24	9
Weight > 15	Nodes 3, 6, 8, 11, 15, 20, 24	7
Weight > 20	Nodes 3, 6, 11, 15, 20	5
Weight > 25	Nodes 8, 10, 11, 13, 19	5
Weight > 30	Nodes 1, 6, 21	3
Weight > 35	Nodes 1, 2	2

These optimal placements pass the sanity check - the number of stations required as the length or weight thresholds increase are nonincreasing, which makes sense because every valid placement is still valid when we increase the thresholds.

Also, we can consider some extreme cases in the network and see how they're dealt with. For the length threshold, we can consider some shortest paths that contain several edges such as the route 7-18-16-17-19-15, which is a shortest path from node 7 to node 15. A charging station should be placed at some node within this route when the length threshold is 5 or below, and we can verify that there is.

Similarly, we can look at edges with large weights for the weight threshold. The edge between 8 and 9 has weight 16.09, so one of 8 or 9 must have a charging station when the weight threshold is 15 or below, and this is also the case.

This table shows the results from our experiments using $\alpha = 1.25$ when adding charging stations:

Placement	Path Length	Path Weight	L2_Diff	Max Flow Base	Max Flow New
Greedy1	6.6	26.726	0.00022	10.38	10.408
Greedy2	7.4	29.946	0.00016	10.334	10.29
Greedy3	6.8	27.692	0.00044	10.156	10.158
Heuristic1	7	30.59	0.00014	10.376	10.378
Heuristic2	7	32.522	0.00036	10.186	10.126
Heuristic3	7	29.302	0.00062	10.182	10.16
Random1	7	32.522	0.00028	10.326	10.292
Random2	7	30.59	0.00058	10.546	10.542
Random3	6.8	26.404	0.00024	10.142	10.138

The next table shows the results from our experiments using $\alpha = 1.5$ when adding charging stations:

Placement	Path Length	Path Weight	L2_Diff	Max Flow Base	Max Flow New
Greedy1	6.6	26.726	0.00048	10.626	10.66
Greedy2	7.4	29.946	0.00022	10.368	10.35
Greedy3	6.8	27.692	0.00066	10.276	10.26
Heuristic1	7	30.59	0.0002	10.648	10.646
Heuristic2	7	32.522	0.00022	10.054	10.064
Heuristic3	7	29.302	0.00034	10.2	10.196
Random1	7.4	28.658	0.00022	10.558	10.566
Random2	6.6	29.302	0.00032	10.112	10.132
Random3	6.6	29.302	0.00026	10.188	10.154

Our experiments as recorded above produced surprising results. Most surprisingly, there was far less variation across placement algorithms than we anticipated there being. This was true for both the path length and weight measurements, as well as the probabilistic traffic flow measurements.

Regarding the path weights and lengths, we saw that the greedy algorithms produced shorter paths than the heuristic paths did. In this regard the greedy algorithms seemed to outperform the heuristic algorithms.

However, when we looked at the probabilistic measures, we saw that the heuristic algorithms tended to slightly outperform the greedy algorithms. Especially in the case when we used $\alpha = 1.5$, we saw that the heuristic algorithms produced less disturbance to traffic flows than the greedy algorithms did. Under $\alpha = 1.5$ we also observed that the heuristic placements resulted in smaller maximum traffic flows compared to the greedy algorithms, which suggests that the heuristic algorithms fared marginally better than the greedy algorithms in not clogging up traffic routes.

The randomized placements seemed to perform better than the heuristic placements yet worse than the greedy placements regarding the path weight and length placements. For the probabilistic measures, there was little to no change when we added the stations. This suggests that randomized placement may be practical in the case of extremely large networks as the benefits of greedy and heuristic station placements may not justify the process of computationally defining and approximating traffic flows through sufficiently large networks.

4 Discussion

Optimally placing charging stations in a transportation network is hard. A portion of the difficulty comes from defining the problem and objectives specifically, but also in a way that allows for quantitative calculation of an optimum.

To that end, one thing that we would have liked to work on further is a more in-depth application of OR methods. In formulating our problem as a set cover problem, we were forced to design our optimization program to take in a predefined set of routes, and this only worked due to the simplicity of the network. We dealt with a highly idealized and unrealistic network and so we weren't able to evaluate performance as we scaled the size of our network or added additional constraints to our network model.

In the end, we transitioned our project to evaluate the effect of station placement on traffic flow, and we didn't have time to fully flesh out a way to model this effect. Thus the majority of our results involve comparing the effects for different placements instead of finding a particular placement that was optimal. Another problem we encountered was that our results showed little to no variation between placements. This made it hard to establish how effective or ineffective our algorithms worked at achieving their goals, and perhaps highlights the need for a different way to evaluate performance.

Another area in which we would have liked to research was different definitions of optimality. We chose to set our target as even traffic flow at all nodes as it made intuitive sense - this would theoretically minimize congestion. However, there were many other ideas we discussed in the meetings that never came to fruition, such as choosing charging stations to direct traffic towards a node or divert traffic away from one.

One problem we had with our project because the stationary distribution requires prior knowledge of transition probabilities, which are generally difficult to calculate in practice. When we were dealing with the Sioux Falls network, this wasn't a problem as turning probabilities were given, but we wanted to test our methods and conclusions on larger or real-life networks (such as the Pittsburgh bus system) and this prevented us from doing so. However, we thought our approach of modifying the transition probability to a station and rescaling the other probabilities was a creative workaround and it also afforded us an opportunity to test different ways to scale the probabilities and added another dimension to our analysis.

Finally, all of our code and csv files with raw results can be found here in our GitHub repository for this project.

Bibliography

[1] Henrik Fredriksson, Mattias Dahl, and Johan Holmgren. "Optimal placement of Charging Stations for Electric Vehicles in large-scale Transportation Networks". In: *Procedia Computer Science* 160 (2019), pp. 77–84. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2019.09.446. URL: http://www.sciencedirect.com/science/article/pii/S1877050919316618 (page 1).

[2] Ben Stabler. *Transportation Networks*. URL: https://github.com/bstabler/TransportationNetworks/tree/master/SiouxFalls (page 2).

[3] *Dijkstra's shortest path algorithm*. URL: https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/ (page 4).