Optimizing the placement of EV charging stations

Chen Zhang Brianna Gautama Ben Rutkowski

Dec 11 2023

Problem Statement

With sustainability gaining increasing attention, policymakers and scholars are exploring various resources to build a more sustainable world. The transportation sector, which accounts for 29% of U.S. emissions and represents the largest source of total emissions in the country, has attracted particular attention (Center for Climate and Energy Solutions, 2022). Additionally, transportation spending ranks as the second-largest expenditure for American households (U.S. Bureau of Labor Statistics, 2022). As a result, governments and the public are keen on understanding the transportation sector, particularly the potential of Electric Vehicles (EVs) for low or no tailpipe emissions (U.S. Department of Energy Alternative Fuels Data Center, 2021). Although tailpipe emissions are only one factor in considering a vehicle's life cycle emissions, EVs are more environmentally friendly than Internal Combustion Engine (ICE) vehicles if we consider the gasoline and electricity fuel paths, including the combination of feedstock, production process, and fuel type. EVs that use relatively low-polluting energy sources for electricity generation have an especially large life cycle emissions advantage over similar ICE vehicles (U.S. Department of Energy Alternative Fuels Data Center (2021), MIT Climate Portal (2022), MIT Energy Initiative (2019)).

In addition, the fast-growing EV market in the United States shows it can be a promising solution. EV sales in the United States, increased nearly 200 percent between the second quarter 2020 and the second quarter 2021, contributing to a domestic penetration rate of 3.6 percent during the pandemic (Fischer et al., 2021). Bloomberg NEF projects global passenger EV sales to rise sharply in the coming years, with plug-in vehicle sales expected to increase from 6.6 million in 2021 to 20.6 million in 2025, representing 23% of new passenger vehicle sales by 2025 (BloombergNEF, 2022). Simulating EV sales is not only based on their growing market share, but also on their potential as a promising solution to reduce air pollution from on-road vehicles.

Two primary challenges facing electric vehicle (EV) sales are the availability of EV charging stations and the limited mileage range of EVs. Our project aims to address the optimization of EV charging station placement, as proposed by Lam et al. (2014), emphasizing the critical importance of strategically locating charging stations. These stations must not only be widespread to ensure acces-

sibility for EVs within their driving range, but also strategically positioned to enable comprehensive city coverage, allowing EVs to navigate the entire urban area after recharging.

1 The Station-Node Model

Our model aims to find the position and scale of EV charging stations that minimize the cost of build such stations and satisfy the local charging demand of the EV owners. We generalize the local demand with N sample points, positioned in $\mathbb{R}^2$. We call these sample points *nodes*. Each node i has a *demand factor* $F_i > 0$ for $i = 1, \ldots, N$. These nodes and demand factors may represent counties, cities, or in the case of our paper, regions of a city designated by ZIP code. The demand factor, may correlate to the amount of EV owners, or the amount of EV traffic within the area represented by the node.

We want to provide charging stations to meet such demands. In our model we define a *station* as a point in $\mathbb{R}^2$ paired with a *supply factor* $z_k > 0$ (for the k-th station). A station k is able to provide supply to node i proportional to it's supply factor z_k and its the distance between the node i and station k. When station k is position exactly at node i, it is able to provide its full supply z_k. But as station k is moved further from node i, its ability to supply node i approaches to 0. In our model we define the supply of station k to node i as

$$S_{ik} = z_k e^{-r d_{ik}}$$

where d_{ik} is the Euclidean distance between node i and station k and r is the decay rate. In this paper we use $r = 0.5$. An arrangement of M stations is feasible if the demand F_i of each node i is being provided an equal or greater amount of supply from every station total. That is, node i is satisfied when

$$\sum_{k=1}^{M} z_k e^{-r d_{ik}} \geq F_i. \tag{1}$$

The cost to build a station is proportional to it's scale factor. In our model the cost of station k is a result of the sum of the initial building cost c and the scaling cost f proportional to it's scaling factor z_k, i.e

$$C_k = c + f z_k. \tag{2}$$

We want to minimize the total cost to build all stations while satisfying the demand of all nodes. The optimal arrangement of stations is the solution to the

following problem.

$$\begin{aligned}
\text{minimize} \quad & \sum_{k=1}^{M} c + f z_k && && \text{(3a)} \\
\text{subject to} \quad & \sum_{k=1}^{M} z_k e^{-r d_{ik}} \geq F_i && \text{for } i = 1, \ldots, N && \text{(3b)} \\
& z_k \geq 0 && \text{for } k = 1, \ldots, M && \text{(3c)} \\
& && && \text{(3d)}
\end{aligned}$$

where $d_{ik} = \sqrt{(u_i - x_k)^2 + (v_i - y_k)^2}$, and (u_i, v_i) and (x_k, y_k) are the positions of node i and station k respectively.

Each station has three degrees of freedom: x_k, y_k, z_k. Thus finding the optimal arrangement of M stations is at least a $3M$ dimensional optimization problem. Furthermore, it is a greater optimization problem to find an optimal M that has the best optimal solution. Note the inclusion of an initial building cost c in our model. This is necessary to add a penalty to building many stations. If this were not the case, it may be optimal to build a single station directly on top of each node, scaled perfectly to satisfy that node. But given a large set of N nodes, this optimal solution may be to build N stations, which is trivial.

1.1 Optimization Method for the Station-Node Model

Due to the high-dimensional nature of the station-node model, we believe that stochastic methods such as Simulated Annealing will be a desirable method to find a global minimum. Also, since the station-node model is defined as a constrained optimization problem, simulated annealing with penalty method is chosen as the optimization algorithm.

1.2 Implementation of Simulated Annealing with Penalty method

The code for the implementation of the Simulated Annealing with penalty method can be found in `simulated_annealing.ipynb`. The code is adapted from Kochenderfer and Wheeler (2019), with slight modifications.

Our design variable is defined to be

$$\mathbf{x} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ b_1 \\ \vdots \\ x_M \\ y_M \\ z_M \\ b_M \end{pmatrix}$$

where M is the maximum number of stations considered by the model, x_i and y_i are the x, y coordinates of the ith station, z_i represents the capacity of the ith station, and b_i represents whether station i will be built.

Here, the objective function becomes

$$\text{minimize} \quad \sum_{k=1}^{M} (c + f z_k) b_k \tag{4}$$

such that the cost of station k is only summed if station k is built ($b_k = 1$). Thus, we add another constraint

$$b_k \in \{0, 1\} \qquad \text{for } k = 1, \ldots, M \tag{5}$$

where station k will be built when $b_k = 1$, and station k will not be built when $b_k = 0$.

In order to satisfy the constraints set out in (3b), let

$$g_i = F_i - \sum_{k=1}^{M} z_k e^{-r d_{ik}}.$$

Then constraint (3b) is equivalent to $g_i \leq 0$

Here, we employ a combination of the simple penalty and quadratic penalty, yielding a penalty function:

$$p = \rho_1 \sum_{i=1}^{N} (\max(g_i, 0))^2 + \rho_2 \sum_{i} (g_i(x) > 0) \tag{6}$$

where ρ_1 and ρ_2 represents the magnitude of contribution of the simple penalty and quadratic penalty to the overall penalty function, and ρ_1 and ρ_2 will be increased by a factor of γ in every iteration, where $\gamma > 1$. When $p = 0$, constraint (3b) is satisfied.

Additionally, in order to satisfy (3c) and (5), we implement these constraints within the simulated annealing algorithm, restricting the values of $z_k \geq 0$,

and $b_k \in \{0, 1\}$. Thus, we satisfy (3b-c) and (5) by implementing the penalty function and above restrictions.

Here, the objective function becomes

$$\text{minimize} \quad \sum_{k=1}^{M} (c + fz_k)b_k + p \tag{7}$$

such that the cost of station k is only summed if station k is built ($b_k = 1$).

Moreover, we constrain the location of the stations. We first find the smallest rectangular region that contains all the nodes with demand, defined by the 4 vertices $(min(x_{node}), min(y_{node}))$, $(min(x_{node}), max(y_{node}))$, $(max(x_{node}), min(y_{node}))$, $(max(x_{node}), max(y_{node}))$. Thus, the values of x_M and y_M will be valid only when they lie within the region, and this is enforced within the simulated annealing algorithm.

1.3 Testing the Simulated Annealing with Penalty Method with Generated Data

For the first part of testing, we apply generated data to the model, which consists of 8 nodes, and we take the initial cost for building a station $c = \$15$ and cost for scaling the capacity of the station $f = \$10$.

We initialized the design variable x, and set the variable $M = 3$, replicated the test 10 times, and yielded the following results:

	Mean	Standard Deviation
Cost	127	4.4721
Constraint g_i	-3.7362	0.7618

From 10 iterations, we obtained the best result (where the best result had the most minimized cost, with the node constraint being satisfied):

$$\sum_{k=1}^{M} (c + fz_k)b_k = \$125 \tag{8}$$

$$\sum_{k=1}^{M} g_k = -2.5286 \tag{9}$$

$$\sum_{k=1}^{M} fb_k = 11 \tag{10}$$

$$\sum_{k=1}^{M} b_k = 1 \tag{11}$$

$$\tag{12}$$

From the above results, the minimized cost to supply adequate charging infrastructure to these set of nodes, given their charging demand, is calculated

to be \$125. The constraint is well satisfied, yielding $\sum_{k=1}^{M} g_k \leq 0$. Also, the results suggest that the optimized charging infrastructure would have 1 station and a total charging capacity of 11.

Furthermore, we quantitatively evaluated our results, by computing number of function evaluations, computing time, and convergence, and the results are tabulated below:

Metric	Mean	Standard Deviation
Convergence	3.9996	0.9783
Computation Time	0.20461	0.10347
Function Evaluations	2004	0

1.4 Testing the Simulated Annealing with Penalty Method with Real-life Data

For the second part of testing, we apply the real-life dataset (EV Charging Demand in Queens County) to the model, where the processed dataset consist of x,y coordinates of 72 different locations, and its corresponding local demand. The code used to process the dataset can be found in `data_processing.py`. We take $c = \$3500$ and $f = \$2000$, which are the initial cost of building a station and the cost for scaling the capacity of the station in Queens county respectively.

We initialized the design variable x, and set the variable $M = 30$, replicated the test 10 times, and yielded the following results:

	Mean	Standard Deviation
Cost	1698900	229856.8576
Constraint g_i	-131.2177	113.9397

From 10 iterations, we obtain the best result (where the best result had the most minimized cost, with the node constraint being satisfied):

$$\sum_{k=1}^{M}(c + fz_k)b_k = \$1475000 \tag{13}$$

$$\sum_{k=1}^{M} g_k = -221.5633 \tag{14}$$

$$\sum_{k=1}^{M} fb_k = 713 \tag{15}$$

$$\sum_{k=1}^{M} b_k = 14 \tag{16}$$

$$\tag{17}$$

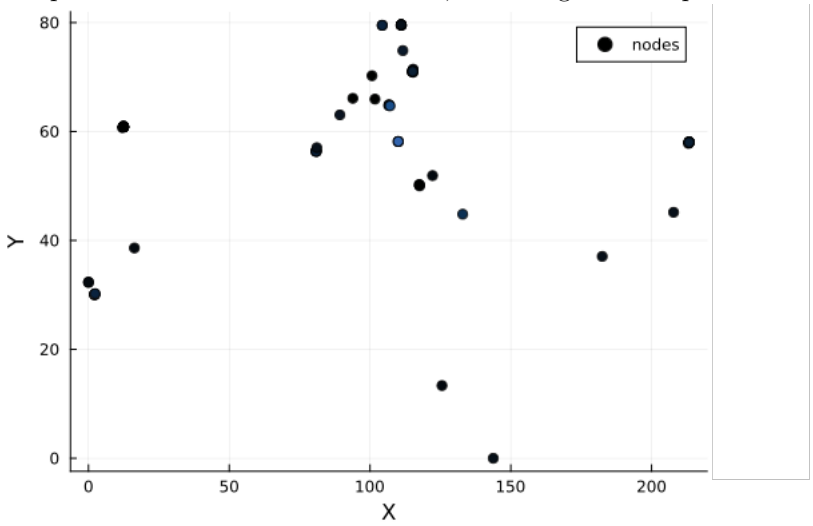
From the above results, the minimized cost to supply adequate charging infrastructure to Queens county, given their charging demand, is calculated to

be \$1475000. The constraint is well satisfied, yielding $\sum_{k=1}^{M} g_k \leq 0$. Also, the results suggest that the optimized charging infrastructure would have 14 stations and a total charging capacity of 713.

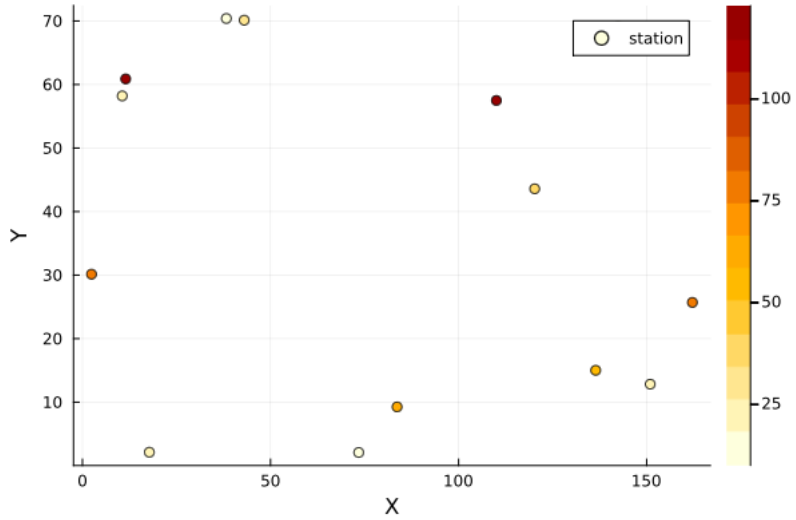
Furthermore, we quantitatively evaluated our results, by computing number of function evaluations, computing time, and convergence, and the results are tabulated below:

Metric	Mean	Standard Deviation
Convergence	4.1012	3.7528
Computation Time	315.6481	5.1568
Function Evaluations	702000	0

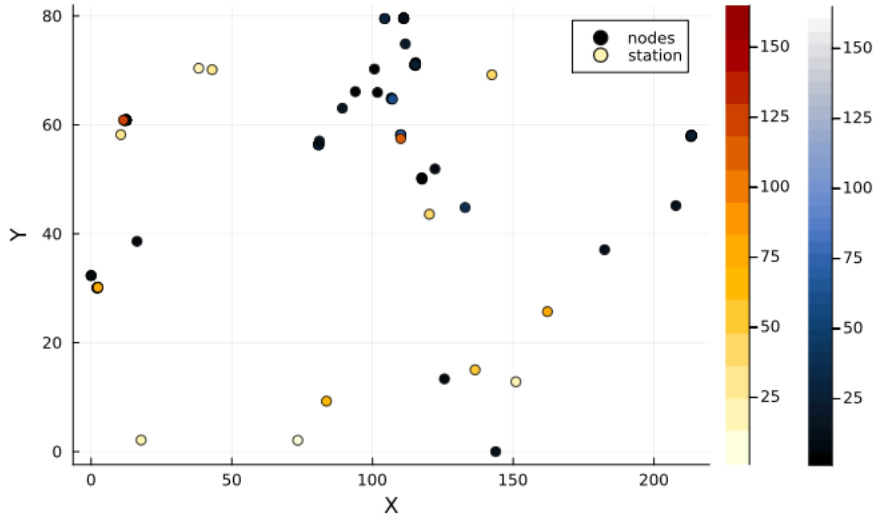
We plot the locations of the nodes, as shown below, where the color of each point represents the demand of each node, according to our input data.



Here, we plot the optimized location of stations generated by our algorithm, and the color of each point represents the capacity of each station.



Finally, we combine the above 2 plots together, where the blue colormap represents the colormap for the demand of nodes, and the yellow colormap represents the capacity of the charging stations.



From this final plot, we can see that although there is a cluster of dark blue points (low demand) in the central upper region, there are only few stations with higher capacity near the cluster to satisfy its demand. This suggests that it may be more cost-effective to build stations with high capacity if there is a cluster of many nodes with low demand. Moreover, there are some low-capacity stations built in the lower left region despite not having nodes in the area, which suggests that these are not well optimized yet, and this issue can be improved by modifying the parameter r to be higher, which implies that each station will provide even less supply to nodes located further away. Finally, for the nodes

that are isolated, we see that mostly, there are higher capacity stations built near these nodes, to maximally supply charging capacity to these isolated nodes.

2 An Alternative Solution

In order to gain more accuracy on solving the original problem (3), we must perform some analysis on its specific form. If we fix the position of every station in our arrangement and only vary the supply factor z_k of each station, the coefficient $e^{-rd_{ik}}$ in inequality (3b) remains constant. In this case, each sum in (3b) is simply a linear combination of the variables z_k. This makes the entirety of the original problem into a linear program (LP). Fortunately, the global optimum of an LP can be calculated directly in polynomial time (if such an optimum exists). In this section we propose a potential alternative solution method to (3) and investigate its potential success by implementing it in julia and performing some preliminary tests.

2.1 The Linear Sub-problem

Fixing x_k, y_k for all $k = 1, \ldots, M$ we get the following LP from (3).

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & \mathbf{f}^T\mathbf{z} && \text{(18a)} \\ \text{subject to} \quad & -D\mathbf{z} \leq -\mathbf{F} && \text{(18b)} \\ & \mathbf{z} \geq \mathbf{0} && \text{(18c)} \end{aligned}$$

where $\mathbf{f} = (f \cdots f)^T$, $\mathbf{F} = (F_1 \cdots F_N)^T$ and $D_{ik} = e^{-rd_{ik}}$. Note that in (18a), we do not include the initial cost constants c in our cost function. If we were to include it, the vector-form formula corresponding to (3a) would simply be

$$cM + \mathbf{f}^T\mathbf{z}. \tag{19}$$

But cM is a constant, thus, any and every solution that minimizes (19) also minimizes (3a).

2.2 Solution Methods Using the LP

We propose a method to solve (3) using a combination of a the Cross Entropy method and the Simplex method. Because the original problem is non-linear and high dimensional, we believe that using a stochastic method will be favorable in order to find a global minimum. However, with the presence of the LP sub-problem (18), we have the opportunity to use our stochastic method in conjunction with a deterministic method.

In this section, we chose our stochastic method to be the Cross Entropy method. From a high level, this method involves taking a distribution over the design space and iteratively decreasing its variance until it (hopefully) converges to the optimal point. Our naive Cross Entropy method is as follows.

Algorithm 1: Naive Cross Entropy

```
compute initial distribution over design space;
for k iterations do
    sample large amount of design points from distribution;
    select m elite sample points that best optimize objective function;
    recalculate distribution from the m elite samples;
end
return distribution
```

The naive algorithm will surely not be successful at all on solving (3) because we will mostly be sampling outside of the feasible set. We may incorporate a penalty term to our objective function to encourage samples to remain feasible. However, with experimentation, this was not nearly enough to coax the samples to both optimize the objective function and remain feasible. We must tune the algorithm to better fit the optimization problem.

Recall that the amount of supply a station provides to a node is related to its distance from such node. If a station is outside of the bounds of the nodes (i.e. the smallest rectangle in $\mathbb{R}^2$ that contains every node position), its influence of supply will mostly matter to the few nodes near the edge of the rectangle. If we move that station within the boundary of the nodes, it will be able to provide a greater supply with the same supply factor. Thus we can infer that the optimal arrangement of stations will only feature stations within the bounds of the nodes.

We use our boundary inference in conjunction with the LP sub-problem observation to achieve our modified Cross Entropy method. Define a design variable to be *legal* if the each station's coordinates (x_k, y_k) for each $k = 1, \ldots, M$ are within the bounds of the node positions.

Algorithm 2: Modified Cross Entropy

```
compute initial distribution over design space;
for k iterations do
    sample large amount of design points;
    select m elite sample points that best optimize objective function,
      favoring legal points first;
    for each elite station do
        fix each station's position, solve the LP (18) and change each
          station's supply factor to the result, ensuring feasibility;
    end
    recalculate distribution from the m elite samples;
end
return distribution
```

2.3 Implementing the Modified Cross Entropy Method

The code of our implementation of the Modified Cross Entropy method is featured in `cross_entropy.jl`, `simplex_algorithm.jl` and `new_objective_function.jl`.

In order to solve the LP, we utilize the Simplex Method which finds the optimal value by traversing the edges of the polytope boundary of the feasible set. The Simplex method assumes that the LP is in equality form, so we first must convert (18) into equality form. The functions `testVertex` and `stepVertex` are modified from Algorithm 11.2 and Algorithm 11.3 in (Kochenderfer & Wheeler 2019) respectively.

We an arrangement of stations as a single vector. In our implementation, we define our design variable to be

$$\mathbf{x} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ \vdots \\ x_M \\ y_M \\ z_M \\ b \end{pmatrix}.$$

Here M is the maximum number of stations to be considered by our model. The component $b \in \{1, \ldots, M\}$ is an integer that represents how many stations are built in the station arrangement represented by $\mathbf{x}$. Note that a single arrangement of stations (position and supply factor) may be represented by multiple distinct $\mathbf{x}$ variables; simply permute the elements x_k, y_k, z_k and x_j, y_j, z_j within $\mathbf{x}$. We want a single station arrangement to be unambiguously represented by a distinct $\mathbf{x}$. We implement `correct` to take a given $\mathbf{x}$, and permute the stations x_k, y_k, z_k within the variable so all stations $k = 1, \ldots, M$ are sorted by x_k respectively. We also correct all $z_k \leq 0$ to be positive (in this case we set $z_k = 1$).

We choose to utilize a penalty function to push design points towards the feasible set. Let

$$g_i = F_i - \sum_{k=1}^{b} z_k e^{-r d_{ik}}.$$

Then constraint (3b) is equivalent to $g_i \leq 0$. We define our penalty to be

$$\rho = \sum_{i=1}^{N} (\max(g_i, 0))^2 + \sum_{k=1}^{b} (\max(-z_k, 0))^2. \tag{20}$$

When $\rho = 0$, our constraints (3a-b) are satisfied.

The objective function is implemented as the `costFunc` function. In `costFunc`, only the cost of the first b stations in $\mathbf{x}$ are summed together. Thus, $\mathbf{x}$ can represent station arrangements of a variable amount of stations (up to M).

2.4 Testing the Modified Cross Entropy Method

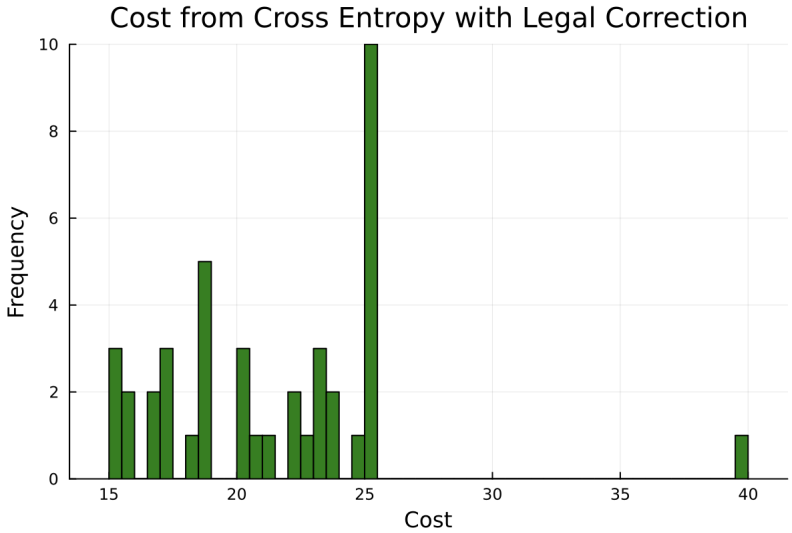
We test the accuracy and performance of The Modified Cross Entropy Method (Algorithm 2) and variants on an a generated data set. The data presented in this section were computed in `cross_entropy.ipynb`. The node system will consist of $N = 8$ nodes whose positions (u_i, v_i) are uniformly sampled from $(0, 10) \times (0, 10)$, and whose demand factor F_i is uniformly sampled from $(0, 5)$. We set the initial cost and scaling cost to be $c = 15$ and $f = 10$ respectively. We implement Algorithm 2 as `crossEntropyLegalCorrection` in our code. The function `crossEntropyLegalCorrection` first calculates an initial distribution over our design space. This initial distribution is uniform in x_k, y_k, z_k, where x_k, y_k is bounded between the minimum and maximum x, y-value respectively of the N node positions. z_k is uniform in $(0, F_l)$ where F_l is the maximum demand factor of all nodes. Lastly b is uniform in $\{1, \ldots, M\}$. The objective function we use is

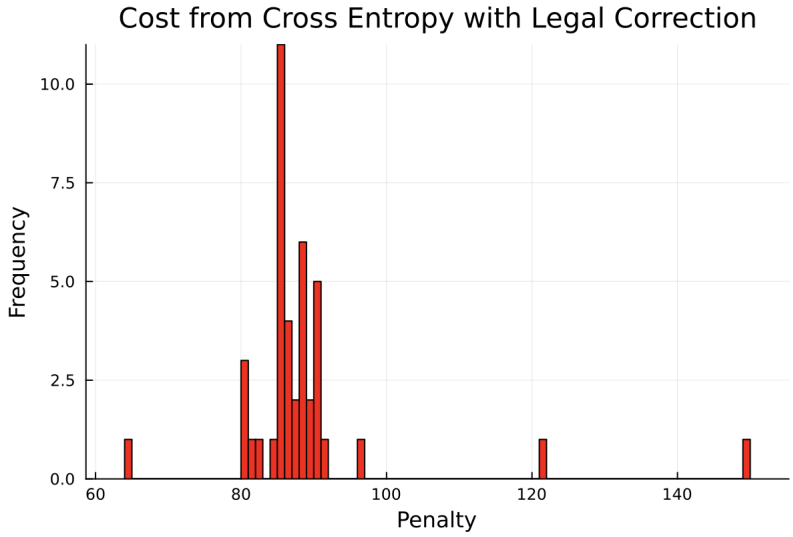
$$f_{\text{objective}}(\mathbf{x}) = \sum_{k=1}^{b} C_k + p\rho. \tag{21}$$

The constant p scales how much the objective function favors $\mathbf{x}$ that are feasible over optimal.

The function next samples $m = 100$ design points from this distribution, corrects them using **correct**, and finds $m_{\text{elite}} = 10$ samples that best optimize the objective function, favoring legal design points. Next `crossEntropyLegalCorrection` optimizes the LP sub-problem for the m_{elite} samples using our function `simplex` and changes the z_k of each sample. Theoretically, this should make each sample feasible. Lastly the function recalculates a new distribution for the next iteration. This distribution is a multivariate normal distribution whose mean and covariant matrix are calculated on the m_{elite} feasible samples. When we have completed k_{max} iterations, we return the best corrected, and legal sample from the last distribution. (We do not push the return sample to be feasible using `simplex`).

We set the maximum number of stations to $M = 3$ and set the penalty constant in (21) to $p = 10$. Lastly, we set the decay rate of a station's ability to provide supply over distance to $r = 0.3$. We perform 50 tests on our node system tuned with the above parameters using `crossEntropyLegalCorrection`. Below we display the histograms of the cost and penalty of the results of our tests. In order to present meaningful graphs, we clean out data points that feature a cost or a penalty greater than 1000.





We observe that a majority of the costs range around $C = 25$. However we excluded 9 data points from the graphs. And the cost of those data points are incredibly large as we can see by the calculated statistics on our test. The calculated mean and standard deviation for the cost is $4.0598 \cdot 10^{14} \pm 2.0005 \cdot 10^{15}$. Furthermore, the penalty for every data point is extremely high. The calculated mean and standard deviation of the penalty is $2.2450 \cdot 10^8 \pm 1.5875 \cdot 10^9$. We want this number to be 0. It appears that forcing samples to be legal or "more feasible" actually cause the results to stray extremely far from the feasible set.

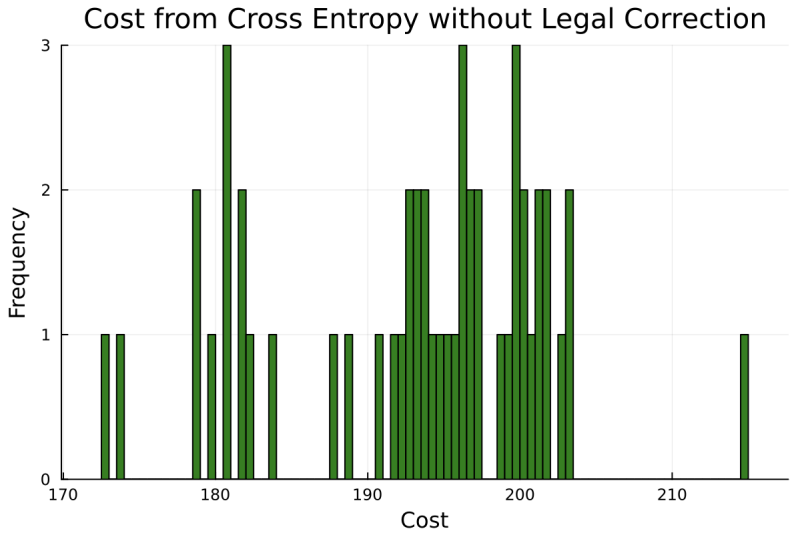
We must make some modifications to our method in order to produce accurate and consistent results. We observed that when the model is left to choose freely how many stations to build (represented by b), it will quickly converge to $b = 1$ within one or two iterations. Was this because a single station is op-

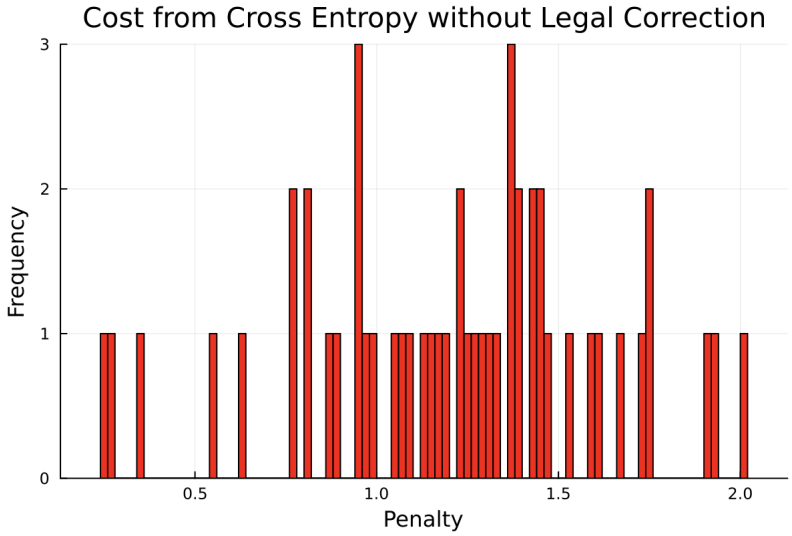
timal or is it a defect of the current algorithm? We wanted to study this with phenomenon with greater control so, implemented into our code is the ability to run every optimization while holding the value of b fixed.

In Algorithm 2, we sample from the current distribution, correct (by sorting x_k, and pushing z_k to be positive), and then find the best m_{elite} samples that optimize the function, *favoring legal samples first*. However, in practice, we found that, out of our entire $m = 100$ samples, it was very rare that we found a single sample that is legal. And in the cases where there are around five legal samples, there was a very low probability that they are within the top m_{elite} optimizing samples. So when we *favor legal samples first*, we are mostly incorporating poor preforming samples into the calculation of the new distribution. These samples can drastically skew the mean and covariance.

In the function `crossEntropy`, we preform the same steps as in `crossEntropyLegalCorrection` except when sampling the top m_{elite} samples, the function remain ignorant to whether the sample is legal or not. The function then proceeds solve the LP sub-problem, modify the z_k values, and continue to follow Algorithm 2.

Below are histograms of the cost and penalty of 50 points calculated by `crossEntropy` on (21) with $p = 10$. We also keep the number of stations built fixed to be $b = 3$.





The mean and standard deviation of the cost is 193.0282 ± 8.8827 and the mean and standard deviation of the penalty is 1.2024 ± 0.4088. We see that this method has a much higher degree of convergence. But most importantly, our results are almost completely feasible. The total penalty on every constraint averages 1.2.

2.5 Further Research on the Alternative Solution

Note how in both functions `crossEntropyLegalCorrection` and `crossEntropy`, we do not solve the LP sub-problem on the final return value. Wouldn't it be more optimal to solve the LP to ensure that any final result is at least feasible? In theory yes, however more this step must be handled with more scrutiny. The current method we have implemented to solve the LP uses the Simplex method. This method my be preferable because it solves a given LP in polynomial time. However, this method operates by solving many linear systems in the form of $A\mathbf{x} = \mathbf{b}$. Our implementation uses the julia package `LinearAlgebra` and operator `A\b` to perform this task. According to the `LinearAlgebra` documentation, this operator is primarily computed using LU-factorization when handling non-square matrices. If a matrix is ill-conditioned, this solution may be inaccurate. In our model, each node introduces a new constraint of the form (3b). Large, ill-conditioned matrices will further decrease computational accuracy. And if we have a large number of nodes N, then corresponding linear systems will have a matrix of size $O(N^2)$, risking greater floating-point error.

We have seen this phenomenon in practice when experimenting with our code. When solving the LP on samples that are close to feasible, the resulting design point will either be equal to the original design point or violate the penalty greater than the original design point.

If one is able to accurately and reliably solve the LP sub-problem, we hypothesize that Algorithm 2 will provide optimal and consistent results. We believe

that this is the crux to addressing the failure of our first method `crossEntropyLegalCorrection`. For, if we can accurately calculate m_{elite} completely feasible design points from a sample set of nearly all legal samples, we hypothesize that the distribution on the next iteration will produce nearly all legal samples. In this case *favoring legal samples first* may only discard one or two non-legal samples from the elite set preserving the optimality of the distribution.

3 Work Decomposition

Our team holds regular weekly meetings to assess progress. Cassie is responsible for crafting the problem statement, which involves providing an overarching perspective on the issue. This encompasses conducting literature reviews, citing relevant sources, brainstorm problem models, and attempting to locate a suitable dataset for the project. It also involves the analysis of results. Cassie accessed a dataset from the New York State public dataset, containing information on Conventional Vehicle Registrations, Electric Vehicle Registrations, and Charging Stations. The dataset provides detailed characteristics of nearly all charging locations, sourced from the U.S. Department of Energy Alternative Fuel Data Center. Data points include Station Name, Street Address, Zip Code, EV level number, Open Date, and other relevant characteristics. To tailor the dataset to our specific program, Cassie filtered it to only consider Registration Valid Year 2022 in Queens County, New York. Queens was chosen because it ranks among the top three counties with the highest EV registrations in the state. This makes Queens a representative location for assessing EV demand in New York State without introducing high dimensionality problems. The total EV registrations in Queens amount to 1572. After filtering the dataset, Cassie pinpointed latitude and longitude for each specific Zip code to determine the EV demand 'nest' of the problem. Additionally, Cassie collected information on the cost of charging stations in New York State using publicly available government data. In the New York City metro area, a typical installation costs between \$2,000 and \$10,000 per port, while in the rest of the state, a typical installation costs between \$2,000 and \$5,000 per port.

Rutkowski was responsible for defining both mathematical models: the applied model, and the ideal model. He further performed analysis on the model and discovered the linear program sub-problem and designed the alternative solution method to solving the original optimization problem using a combination of the Simplex method and the Cross Entropy method to find a minimum within the feasible set. Rutkowski implemented the code for both models, the code for the Simplex method and the code for the Cross Entropy method in julia. He performed the necessary experiments to tune this method to produce consistent results. He also demonstrated the methods prospect of success on an ideal low dimensional problem. He wrote the entirety of §2 as well as the model description in the introduction of §1. He implemented all of the code in `cross_entropy.ipynb`, `cross_entropy.jl`, `new_objective_function.jl` and `simplex_algorithm.jl`.

Brianna was responsible for processing the dataset, which involves converting latitude and longitude values into spatial coordinates and normalizing the values. She also implemented the Simulated Annealing with penalty method in Julia, and applied the optimization method on a set of generated data, as well as the real-life dataset, to generate locations and capacity of charging stations that minimizes the cost function while obeying the constraints set in the problem. Brianna also analyzed the results and tested out the set of parameters that would yield the best optimization results. She also evaluated the optimization algorithm using quantitative evaluation metrics. Brianna also generated the scatter plots which display the nodes (according to the dataset) and optimized stations (optimized by the algorithm), with charging demand and capacity represented by the color of the points respectively. She implemented all of the code in `data_processing.py` and in `simulated_annealing.ipynb` (except for the block of code defining the station-node model which was implemented by Rutkowski).

3.1 Coding Experience

Ben Rutkowski

It took myself a couple attempts to converge at working model. We originally were going to investigate optimization methods on a model from the literature. However, everything we read seemed to be beyond the scope of the methods we learned in class or part of a different field of study altogether (statistics).

After our attempts I sat down to design our own model. The original model that I had proposed used the reciprocal of distance (instead of an exponential decay) to calculate the roll-off of supply provided by a station with respect to its distance from a node. The rationale was that, as distance increased, the supply would decrease. What completely skipped my mind with this model is was happens on the other side of the spectrum: a station will provide infinite supply when it is infinitesimally close to a node. It wasn't until my preliminary tests with my code that I caught this bug and changed it to the realistic exponential decay.

Before implementing the Simplex method as our function to solve an LP, I attempted to implement a algorithm due to Seidel for solving linear programs. The algorithm is random and incremental and has an expected time complexity of $O(d!n)$, where d is the dimension of the linear program and n is the number of constraints. However, I was not able finish my implementation given my time constraints. I had to switch methods completely and discard all of my work and code. After discovering the pitfalls of our current implementation of the Simplex method, I believe there is a chance that I was correct to attempt to implement Seidel to avoid ill-conditioned matrix inverses and the discrepancies that follow. But it was not realistic to for me complete that method by myself given my deadline.

Brianna Gautama

I implemented the Simulated Annealing with penalty method. Initially, I implemented only the simple penalty function, but the method was inclined to output the total number of stations as 0. As a result, I implemented the mixed penalty function, composed of both the simple and quadratic penalty function. In addition, I also increased the value of the parameter γ to increase the punishment of the penalty, thereby making it more feasible to converge to a reasonable value. Another issue I encountered was that the locations of stations tend to become similar, and I addressed the issue by doing random initialization of the coordinates in the design variable, and modified the simulated annealing algorithm, such that at each iteration, we can add or subtract the random values, as opposed to having the option to only add the random value to the current design variable. As a result, it is more likely for the locations of the stations to be more different from each other, and it will be more feasible for the algorithm to find an optimized set of stations to satisfy the demand. Moreover, the algorithm was likely to output points that were unreasonable, such that the points are located relatively far out of the region containing all the nodes. As a result, I implemented an additional constraint within the simulated annealing method, such that the stations are restricted to be built within the smallest rectangular region containing all the nodes. After implementing all these changes and fine-tuning of the parameters, the model could converge, satisfy the constraints, and produce reasonable results. Nonetheless, further improvement could be done to add more constraints to the model, especially with respect to how stations provide charging supply to nodes with demand, in order to make the model more realistic and obtain better results.

4 Conclusion

The Simulated Annealing with Penalty method is effective in optimizing the station-node model, and can be scaled to optimize data with at least 72 nodes, as shown in the above section (Section 1.4). Nonetheless, further modifications to the station-node model is required to allow the model to become more applicable to real-life circumstances and thus obtain more reasonable and feasible optimization results.

On further analysis of the given model, we find that optimizing cost in an arrangement of stations with fixed positions is a linear program problem. We motivated an alternative solution method using Cross Entropy method and Simplex method and demonstrated its potential success on optimization problems of this form. We detail aspects that can be improved upon in our implementation and hypothesize that it will produce accurate and consistent results.

References

BloombergNEF, "Electric Vehicle Outlook 2022," 2022.

Center for Climate and Energy Solutions, "U.S. Emissions," 2022.

Fischer, Maximilian, Nicolaas Kramer, Inga Maurer, and Rachel Mickelson, "A turning point for US auto dealers: The unstoppable electric car," 2021.

Kochenderfer, Mykel J. and Tim A. Wheeler, "Algorithms for Optimization," *The MIT Press*, 2019.

Lam, Albert Y.S., Yiu-Wing Leung, and Xiaowen Chu, "Electric Vehicle Charging Station Placement: Formulation, Complexity, and Solutions," *IEEE Transactions on Smart Grid, 2014*, 2014, *5* (6), 1949–3053.

MIT Climate Portal, "Are electric vehicles definityly better for the climate than gas-powered cars?," 2022.

MIT Energy Initiative, "Insights Into Future Mobility," 2019.

U.S. Bureau of Labor Statistics, "Consumer Expenditures - 2021," 2022.

U.S. Department of Energy Alternative Fuels Data Center, "Emissions from Electric Vehicle," 2021.