112-2 Operations Research Final Project
GoStation Site Selection Problem

Team O

Hong-Kai Yang B11611047
Yi-Ting Chen B11705051
Chi-Wei Ho B11702044
Yu-Ting Chou B11702080

June 6, 2024

1 Introduction

Gogoro is a company renowned for its own products of electric scooters, scooter-sharing platform as well as the largest battery-swapping refueling system for electric scooters in Taiwan. For individuals without scooters, the GoShare app provides a convenient solution for locating nearby available cars and accessing relevant information for rental purposes. No matter people who use GoShare for public-sharing scooters or their own Gogoro electric scooter products, they can utilize GoStation for battery charging.

Currently, no matter riders use GoShare or Gogoro electric scooter, when they notice their electric scooters are running low on battery, they can click on the bottom right corner to access the electronic exchange station. Through the mobile app, riders can view available charging locations and relevant battery information, providing a convenient service for customers.

Despite seemingly ideal conditions, there are numerous issues concerning charging and battery swapping stations. Our data analysis revealed that some GoStations have low utilization rates while their operational costs remain high. By applying Operations Research techniques, we aim to maximize the dismantling of underperforming GoStations and minimize the shipping costs. This approach will reduce waste and ensure that GoStations operate efficiently.

2 Problem description

Based on existing data from Gogoro, we observed that there are some stations whose utilization are low, and the operation cost is rather high. Therefore, we aim to address issues of resource relocation and minimizing the shipping cost in a phased approach, and out target city is Hsinchu City.

2.1 Part 1: GoStation Optimization Strategy

In the first stage, we aim to reduce the number of GoStation sites as much as possible.

From the current map of Hsinchu City, we can see that there are no GoStation sites near the Air Force base, so we will distribute 10 points every 4 kilometers square on average, avoiding areas near the Air Force

base. One point represents one community. We assume that communities prioritize charging at the nearest GoStation, and thus the existing GoStations can meet the demand.

However, according to our observations, some areas do not have as much demand, and the current operation leads to high operational costs. We hope to reduce the number of existing GoStations as much as possible. Therefore, we will inspect each GoStation individually to decide whether it can be removed or not. The necessary condition to remove a GoStation is: When this GoStation is removed, the community originally relying on it must be able to find the second nearest GoStation, and this second nearest GoStation must be able to supply enough power to the community. In other words, if any community originally relied on this GoStation cannot find a second nearest GoStation that can supply enough batteries, that GoStation cannot be removed.

2.2 Part 2: Minimizing Battery Shipping Cost

After removing excessive GoStations in stage 1, in the second stage, we wanted to know how batteries would be dispatched to the remaining GoStations so that the battery shipping costs could be minimized while meeting each community's demand for fully charged batteries.

Here we split one day into 48 periods, and each lasts for 30 minutes to simplify our model.

The detailed operation method is as follows: Staff will dispatch batteries and transport the fully charged battery inventory from some of the GoStations to the others whose inventory cannot meet the demand in their service coverage. The freight cost is charged linearly according to the shipping distance. The farther the freight is, the higher it charges, and vice versa. The maximum cost is $1 per battery. However, sometimes there could be no excessive inventory in all of the GoStations. This way, staff have to transfer batteries from the warehouse, which leads to an extra battery cost of $1,000 per battery.

Additionally, regarding assigned cost, a certain amount of costs will be incurred each time batteries are transported from the warehouse to GoStations. It is calculated by the number of periods. Therefore, if there is no battery dispatching in a period, there is no assigned cost then.

3 Data Information and Processing

Initially, we planned to use Selenium to scrape one-week data from "Gogoro ". However, we discovered that the data for full battery levels was not directly stored in the website's HTML; it required hovering the mouse over the relevant sections to load the exact numbers. Consequently, we decided to use the coordinates of the line chart (Figure 1) points along with the recorded extrema of the x and y axes. By applying the internal division formula in our Python script, we were able to calculate the data points' timestamps and the number of full batteries for each station.

3.1 Part 1: GoStation Optimization Strategy

To reduce the number of removal Gostations in part 1, the supply of this part should be underestimated and the demand should be overestimated as much as possible.

For the data on supply, we can directly calculate the supply of each data point and the maximum number of batteries at the gostation as the immediate demand. Then, we took their daily peak time usage and added them together to divide the total number of periods adopted to obtain the average demand.

Figure 1: Gostation immediate full battery amount line chart

For the data on demand, the first quantity of fully charged batteries in each period is regarded as the beginning inventory of that period, while the additional number of fully charged batteries within that period is the additional batteries that can be provided during that period. The sum of the two numbers is the total supply. We took the average supply during the peak hours as the data.

3.2 Part 2: Minimizing battery shipping cost

To estimate the immediate demand more accurately, we found that the raw material may have the situation that the newly charged battery will be consumed immediately. Through observation and searching information on major forums, the average number of batteries that can be charged at a site in half an hour is about two to three, but it still varies by regions. In the end, we chose to use 3 batteries as the number that can be fully charged in half an hour as an additional rule for this section. And we also found that in many periods, the maximum number of fully charged batteries will not reach the maximum, which is designed to protect the batteries, so they will control the maximum number of fully charged batteries to 80% of the total. Therefore, in the end, the demand and supply of places where the number of fully charged batteries is the same and is less than 80% of the maximum batteries are added by three batteries.

For the data on demand, we assign the demand from each community to the remaining stations from part 1, and calculate the demand for each remaining station at every time point. Since the removed stations will not be assigned any demand, their demand will be zero at every time point. The main method for calculating the demand at each time point is the same as in part 1, which involves observing the decrease in the number of full batteries over a time period. However, additional rules will be applied to handle this part. Finally, average the original 7-day, 336-period data into a single day with 48 periods.

The supply for this part will only consider the number of batteries produced in each period compared to the previous period. Similarly, the supply file will only include the remaining stations after the phase 1 removals. The original 7-day, 336-period data will then be averaged into a single day with 48 periods.

4 Mathematical model

4.1 Part 1: GoStation Optimization Strategy

There are n communities in Hsinchu City. Let $I = \{1, ..., n\}$ be the set of communities.
There are m GoStations in Hsinchu City. Let $J = \{1, ..., m\}$ be the set of GoStations.
Let D_i be the demand of batteries in community $i \in I$.
Let K_j be the capacity of GoStation $j \in J$.
Let d_{ij} be the distance between community $i \in I$ and GoStation $j \in J$.
Let $G = \sum_{i \in I} d_{ij} \quad \forall j \in J$.
Let $x_j = 1$ if GoStation $j \in J$ will keep running, or 0 otherwise.
Let $y_{ij} = 1$ if the demand in community $i \in I$ is satisfied by GoStaion $j \in J$, or 0 otherwise.
Let w_i be the distance between community $i \in I$ and its cloest GoStation.

$$
\begin{aligned}
\min \quad & \sum_{j \in J} x_j \\
\text{s.t.} \quad & \sum_{i \in I} D_i y_{ij} \leq K_j x_j \quad \forall j \in J && (1) \\
& y_{ij} \leq x_j \quad \forall i \in I \quad \forall j \in J && (2) \\
& \sum_{j \in J} y_{ij} = 1 \quad \forall i \in I \quad \forall j \in J && (3) \\
& w_i \leq d_{ij} x_j + G(1 - x_j) \quad \forall i \in I \quad \forall j \in J && (4) \\
& w_i \leq \sum_{j \in J} d_{ij} y_{ij} \quad \forall i \in I && (5) \\
& x_j \in \{0, 1\} \quad \forall j \in J && (6) \\
& y_{ij} \in \{0, 1\} \quad \forall i \in I \quad \forall j \in J && (7)
\end{aligned}
$$

Constraint (1) specifies that a GoStation can supply enough battery capacity to meet the demand of communities. Constraint (2) regulates the relationship between the availability of sufficient battery supply to community j at a GoStation i and GoStaion i's existence; some stations may not supply enough power to the community even if they keep running. Constraint (3) ensures that each community is served by exactly one GoStation.

Constraint (4) and (5) are used to define the minimum distance between GoStation and community. Constraint (4) specifies the minimum distance between a community and the currently existing stations, and constraint (5) specifies the minimum distance for a community to find a currently existing station that can supply enough power

4.2 Part 2: Minimizing battery shipping cost

Let $M = \{1, ..., 27\}$ be the set of GoStations and $T = \{1, ..., 48\}$ be the set of periods.
Let S_{mt} be the additional supply at GoStation $m \in M$ in the beginning of period $t \in T$,
D_{mt} be the demand of GoStation $m \in M$ at period $t \in T$,
$C_{mm'}$ be the cost shipping a battery from GoStation $m \in M$ to GoStation $m' \in M$.

Let $z_{m,t,m',t+1}$ be the number of batteries transferred from GoStaiton $m \in M$ in the beginning of period $t \in T$ to GoStation $m' \in M$ in the beginning of period $t+1 \in T$,
$f_{m,m',t,t+1} = 1$ if in period $t \in T$ staff transfer batteries from GoStation $m \in M$ to GoStation $m' \in M$, or 0 otherwise,
E_m be the number of batteries added to GoStation $m \in M$ before period 1.

$$\min \sum_{m=1}^{42} \sum_{m'=1/\{m\}}^{42} \sum_{t=1}^{47} C_{mm'} z_{m,t,m',t+1} + \sum_{m=1}^{42} \sum_{m'=1/\{m\}}^{42} C_{mm'} z_{m,48,m',1} + \tag{8}$$

$$\sum_{m=1}^{42} \sum_{m'=1/\{m\}}^{42} \sum_{t=1}^{47} f_{m,t,m',t+1} + \sum_{m=1}^{42} \sum_{m'=1/\{m\}}^{42} f_{m,48,m',1} + 1000 \sum_{m=1}^{42} E_m \tag{9}$$

$$\text{s.t.} \sum_{m'=1}^{42} z_{m',t,m,t+1} + S_{m,t+1} \geq D_{m,t+1} \quad \forall m \in M, \quad \forall t = 1, ..., 47 \tag{10}$$

$$\sum_{m'=1}^{42} z_{m',t,m,t+1} + S_{m,t+1} - D_{m,t+1} = \sum_{m'=1}^{42} z_{m,t+1,m',t+2} \quad \forall m \in M, \quad \forall t = 1, ..., 46 \tag{11}$$

$$\sum_{m'=1}^{42} z_{m',48,m,1} + S_{m,1} + E_m \geq D_{m,1} \quad \forall m \in M \tag{12}$$

$$\sum_{m'=1}^{42} z_{m',48,m,1} + S_{m,1} + E_m - D_{m,1} = \sum_{m'=1}^{42} z_{m,1,m',2} \quad \forall m \in M \tag{13}$$

$$\sum_{m'=1}^{42} z_{m',47,m,48} + S_{m,48} - D_{m,48} = \sum_{m'=1}^{42} z_{m,48,m',1} \quad \forall m \in M \tag{14}$$

$$z_{m',t,m,t+1} \leq 549 f_{m,m',t,t+1} \quad \forall m \in M, \quad \forall m' \in M, \quad \forall t \in T \tag{15}$$

$$z_{m,t,m',t+1} \in \mathbb{N} \quad \forall m \in M, \quad \forall m' \in M, \quad \forall t \in T \tag{16}$$

$$f_{m,m',t,t+1} \in \{0,1\} \quad \forall m \in M, \quad \forall m' \in M, \quad \forall t \in T. \tag{17}$$

Considering minimizing battery shipment cost, we can take a look at (8) and (9). In our objective value, we sum up all types of costs together.

For constraint (10), we make the beginning inventory of period t $\sum_{m'=1}^{42} z_{m',t,m,t+1}$ plus the number of batteries become fully charged during period t $S_{m,t+1}$ be greater than the demand of period t, to ensure we can satisfies all of the demand in each period.

Constraint (11) regulates the ending inventory of one period to be the beginning inventory of the next period at all of the GoStations.

Constraints (12) and (13) indicate that before the beginning of the first period, we have to check if there are no excessive batteries in all of the GoStations. If so, then extra batteries E_m are needed to ensure no shortage in the next periods.

In constraint (15), we identify whether there is an assigned cost incurred in period t or not.

5 Heuristic Algorithm

Before removing any station, we will first calculate the ratio of the average full battery capacity to the maximum battery capacity and sort it from smallest to largest. This ratio represents the load of each station. A higher load means more people are likely to visit this station for swapping batteries, and thus we should consider removing this station last.

Next, based on the sorting we mentioned above, we will use that order to input each station into the following function to determine whether the GoStation should be removed or not.

5.1 Part 1: GoStation Optimization Strategy

```
function plan(abandoned, station):
    assumed_abandoned = abandoned + [station_name]

    if station not in assigned_station:
        RemoveStation(station)
        abandoned = assumed_abandoned
        return True

    floating_communities = CommunitiesUsesStation(station)
    each_community_demand =
    demand[station] / (size of floating_communities)

    for each community in floating_communities:
        new_station = find_the_next_nearest_station(community)
        new_stations.add(new_station)
        //test if the demand of the community can be accepted
        //by the nearest station left. if not return false

    for station in new_stations:
        UpdateNewDemand(station, each_community_demand)

    RemoveStation(station)
    abandoned = assumed_abandoned
    return True
```

When the above function starts, we will assume that this station will also be removed.

We will create a new array called $assumed_abandoned$ with both removed stations and stations that are about to be removed. Then, the function will check if the second closest station to each community can meet each community's demand. If all the second closest station to each community can meet the community's demand, the station will be removed. Otherwise, the station should keep operating.

Regarding the above algorithm, we only need to have each GoStation execute the `plan` function, and each `plan` function will execute at most m times, where m represents the number of communities. Therefore, its time complexity is $O(n \times m)$.

5.2 Part 2: Minimizing battery shipping cost

```
while (not elements in extra are not all zero):
    reset extra for all stations to 0
    set left_battery of stations to their need of extra battery

    total_send_cost = 0

    for period in periods:
        total_battery = 0
        for each station in stations:
            station.left_battery +=
            supply at current period - demand at current period

        for each station in stations:
            if station.left_battery < 0:
                //check from the closest to the farthest station
                //if any one can meet this station need. If there
                //is not any station that can meet the need, then
                //get the batteries from others in the distance order.
                //Store the number of taken stations to workers,
                //the total distance of taken stations
                //to send_distance,
                //and the amount of not available from
                //other stations to  extra_needed.

                if extra_needed > extra[station]
                   extra[station] = extra_needed
                        total_send_cost +=
                        workers +
                        (send_distance / farthest_two_stations)


    if elements in extra is not all zero:
        for each station in stations:
            if extra[station] is in max of extra:
                station.etra += max(extra)
```

Initial Inventory Calculation

- **Start-of-Period Inventory:** Calculate the initial inventory by summing the determined Extra Battery and the starting inventory.

Demand Check

- **Inventory Monitoring:** At each time period, check each user's battery inventory. If a user's inventory is less than zero (indicating demand exceeds supply), initiate the replenishment process.

One-to-One Replenishment

- **Proximity-Based Check:** Begin with the nearest GoStation and check if it can fully meet the user's demand.
- **Full Replenishment:** If the nearest GoStation can fully satisfy the user's need, replenish the batteries accordingly.
- **Partial Replenishment:** If the nearest GoStation cannot fully meet the demand, proceed to the next nearest GoStation, continuing this process until you find a station that can fully satisfy the demand or exhaust all options.

Many-to-One Replenishment

- **Collaborative Replenishment:** If no single GoStation can meet the demand, initiate a many-to-one replenishment strategy.
- **Iterative Fulfillment:** Start again from the nearest GoStation. As long as the station has remaining batteries, provide a portion to the user. Continue this process, distributing batteries from multiple GoStations, until the user's demand is completely satisfied.
- **Incremental Reduction:** Given that no single station can fully meet the demand, each replenishment step will incrementally reduce the outstanding demand until it is entirely covered.

Update Replenishment Records

- **Inventory Adjustment:** Based on the outcomes of the one-to-one and many-to-one replenishments, update the battery stock levels of the GoStations involved in the replenishment process.

Special Situations Handling

- **Extra Battery Utilization:** If all GoStations combined still fall short of meeting the demand, utilize the Extra Batteries to cover the shortfall.
- **Inventory Adjustment:** Adjust the user's inventory to zero in such cases.
- **Extra Battery Usage Tracking:** Record the number of Extra Batteries used, ensuring this is updated in the respective GoStations' records.

This process ensures that at any given time, regardless of fluctuations in battery demand, replenishments are timely and effective, preventing battery shortages.

Repeating the Process

- **Iteration Continuation:** Repeat steps 2-6 until all time periods within the current period are processed.
- **Extra Battery Restocking:** For GoStations that utilized Extra Batteries, increment their start-of-period Extra Battery count by one.
- **Inventory Recalculation:** Recalculate their start-of-period inventory and continue repeating steps 2-6.
- **Process Termination:** Continue this iterative process until no GoStation uses Extra Batteries within the period.

By following these steps, the algorithm ensures continuous, efficient battery inventory management, adapting dynamically to meet demand and prevent shortages.

Our algorithm's main structure involves calculating the battery flow within a certain period. To obtain the remaining battery levels after the flow, we need to iterate over each station. Thus, we'll spend a time complexity of $O(\text{period} \times n)$, where period = 48. Furthermore, the described algorithm needs to be executed until all conditions are met, i.e., the extra supplement is completed. We've observed in advance that there will likely be a maximum of k units of batteries to be replenished, where k is the total extra needed if no station performs exchanges. Hence, the time complexity is $O(k \times \text{period} \times n)$.

6 Results and Comparison

6.1 Part 1: GoStation Optimization Strategy

In the first stage, we see how many GoStations are being torn down and its location. The results are presented in the following table and Figures. Table (1) shows the amount of GoStation being removed. Figure (2) (3) (4) (5) shows the remaining GoStations' locations.

Algorithms	Demolished	Remained
Gurobi	15	27
Heuristics	15	27
Simple Heuristics	14	28

Table 1: GoStation being removed and remained

We can see that our heuristic perform nearly the same as Gurobi, and simple heuristics perform slightly different from them.

6.2 Part 2: Minimizing battery shipping cost

In the second stage, we try to minimize better shipment cost. After obtaining the solution from Gurobi, the heuristic algorithm, and the simple heuristic algorithm, we evaluate the performance of the heuristic we developed by calculating their optimality gaps. The results are presented in the following table. We evaluate

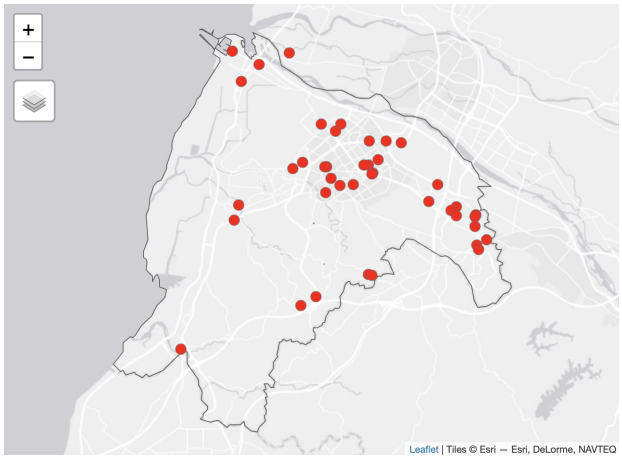

Figure 2: Hsinchu City Original GoStation

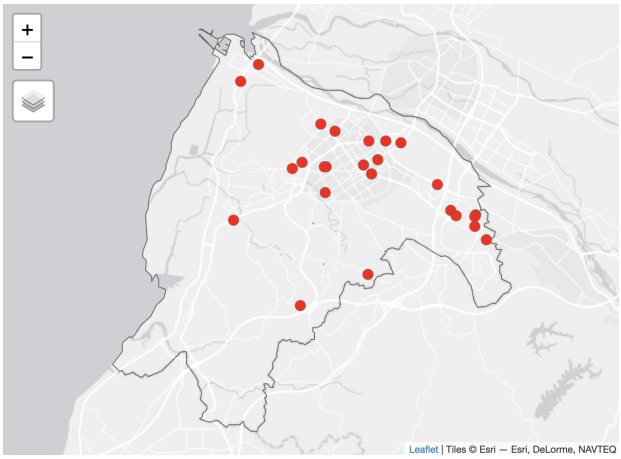

Figure 3: Gurobi Result

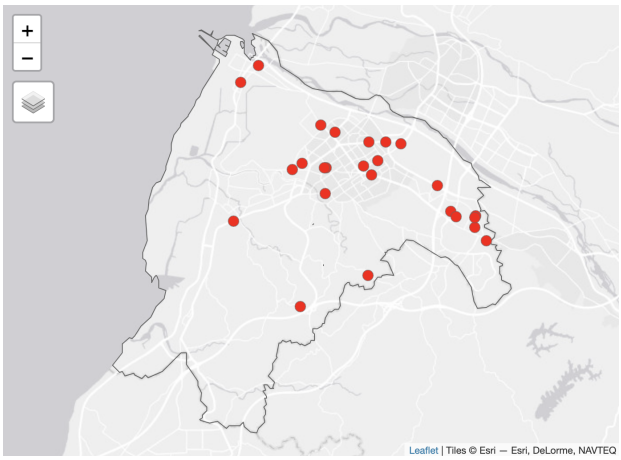

Figure 4: Heuristic Result

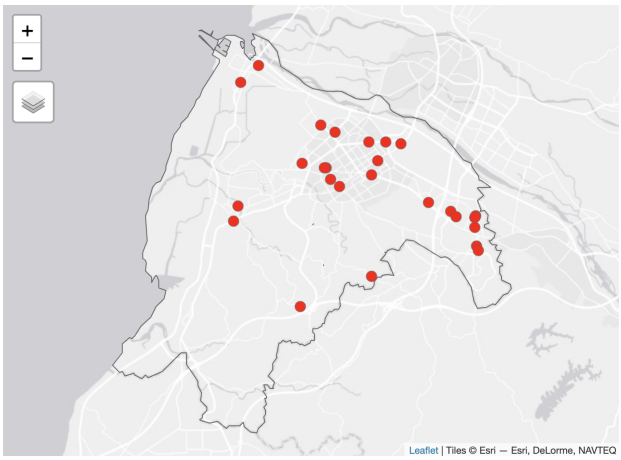

Figure 5: Simple Heuristic Result

Algorithms	shipping times	assigned times
Heuristics	78.7%	381.4%
Simple Heuristics	239.6%	894.3%

Table 2: Optimality gaps of shipping times and assigned times

the performance with two indicators: shipping times and assigned times.

Table 2 shows that our heuristic algorithm performs better than the simple one on both shipping times and assigned times. The following is the reason why: for a simple heuristic, extra batteries are intensively located in particular GoStations. This way, although the total amount of extra batteries of the simple heuristic and heuristic algorithm is identical, it takes the simple heuristic algorithm more shipping times and assigned times to operate.

7 Future Improvement

7.1 Data Acquisition Challenges

The current method of estimating demand using the number of full batteries at each station has limitations. Further research revealed that the charging patterns at each station can vary due to factors such as time, existing circuits, and battery protection measures, making it difficult to accurately predict supply. Additionally, if a particular GoStation runs out of batteries, we cannot accurately determine how many users still need charging at that moment. Therefore, at certain time points, supply cannot be accurately estimated.

7.2 Difficulty in Estimating User Experience

Removing stations could force users to travel further to find a battery swap station, which could negatively impact user experience and, consequently, Gogoro's profitability. Therefore, successfully quantifying the decline in user experience due to increased distance is essential for developing a better mathematical model.

7.3 Implementing in Wider Areas

As our techniques continue to improve, we aim to implement our solution across a broader range of areas. By continually revising and optimizing our algorithm, we can adapt to various scenarios, ensuring that each city benefits from an enhanced Gogoro station network. This tailored approach will help us address the unique needs and challenges of different urban environments, leading to more efficient and effective GoStation locations.