Operations Research, Spring 2025 (113-2)
Final Project Proposal - Group K

Huan-Hsuan Tseng B12705002, Hong-Chi Chen B12705018, Po-Yu Cheng B12705012, and Wen-Yi Tzeng B12705017

Department of Information Management, National Taiwan University

May 31, 2025

Introduction

Shared bicycle systems like Taipei' s YouBike offer convenient urban transport but face major challenges due to supply-demand imbalances, especially during peak hours. High-demand stations often run out of bikes, while return stations lack available docks. Despite over 600 million rides logged, even a small imbalance affects thousands of users, undermining YouBike' s goals of promoting public transport and sustainable travel. To mitigate this, operators use dispatch trucks to rebalance bikes, but determining needs and planning routes is a complex optimization problem.

Though real-time monitoring helps, challenges persist. In New Taipei City, up to 250 personnel and 40 trucks are deployed daily based on big data to manage this issue, yet manual or heuristic planning remains inefficient. To improve operations, we propose a mathematical programming model aimed at optimizing bike redistribution, balancing station availability within limited resources to enhance service levels and system efficiency.

Problem Description

Our project focuses on balancing bicycle supply and demand across Taipei. Each station holds about 10–30 bikes, and continuous user activity leads to frequent imbalances. Without timely redistribution, some stations run out of bikes, while others have no available docks. We use real-time open data from YouBike 2.0 (see [1]), which updates every five minutes and includes the number of available bikes and empty docks at each station. From this, we calculate the available dock ratio—a key indicator of station status. Our goal is to keep this ratio within the ideal 30%–70% range, a threshold recommended by the operator Giant with slight adjustments (see [4]). Stations falling outside this range indicate a need for dispatch: to either replenish bikes or remove excess.

To illustrate the real-world implications, consider a friend of ours living in Xizhi. His daily school commute often exceeded one hour. Although Nanggang (Taipei) and Xizhi (New Taipei City) are geographically adjacent, traveling between them is inconvenient. For instance, traveling from NTU to Nanggang may take just 30 minutes, but reaching Xizhi can take nearly an hour. Despite being close to an MRT station, the disjointed public transportation network forces many commuters to

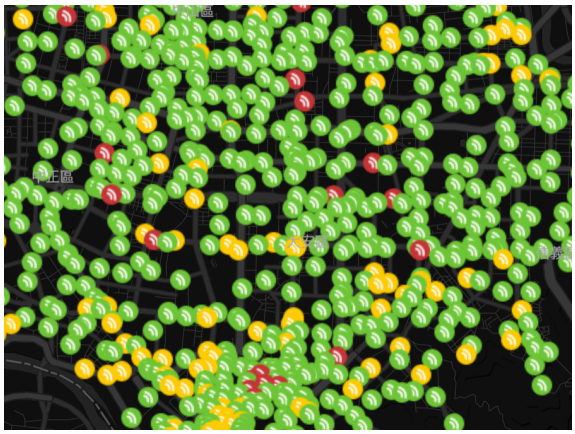


Figure 1: Distribution of some YouBike 2.0 stations (2025/03/07 13:00). Green: balanced; Yellow: no bikes; Red: no docks.

bike between the two districts. This leads to a significant directional flow of bicycles during peak hours.

Given this context and the limitations of our dataset—which excludes New Taipei City (i.e., no data from Xizhi)—we centered our analysis on Nanggang and extended it northward to Neihu, a residential zone also lacking comprehensive MRT access.

To address imbalance, the YouBike operator deploys a fleet of dispatch trucks, each with a specific capacity, loading/unloading rate, and travel speed (approximated by straight-line distance at 60 km/h). Each truck begins and ends at a designated depot and operates within a fixed dispatch time window.

Our goal is to plan dispatch routes that **maximize the number of stations** whose available dock ratios fall within the desirable 30%–70% range after dispatch. The model includes two main components and key constraints:

1. **Estimating how many bikes to move** in or out of each station to achieve the threshold.
2. **Planning optimal routes** for each truck to collect bikes from surplus stations and deliver to those in deficit.

- Every truck has a capacity and starts/ends at the depot. We set the depot at Nanggang station.
- A station can' t provide more bikes than it holds or receive more than its empty docks.
- All dispatching must be completed within the time window.

According to [4], there are typically three dispatch windows each day: 9 AM (Morning refill), 5 PM (Commuter rush preparation), and 10 PM (Night refill and next-day prep). Each dispatch scenario is defined by: Number of trucks, Time window, and Dispatch time slot. This model offers a structured, data-driven approach to mitigating YouBike distribution issues, with a special focus on high-demand, transit-inconvenient zones like Nanggang and its surrounding areas. By improving dock/bike availability at the right time and place, we aim to enhance the overall service quality and user experience.

Model Formulation

Parameters and Sets

Station set:	$i \in N$
	C_i: total capacity of station i
	B_i: current number of bikes at station i
Truck set:	$k \in K$
	Q: maximum capacity per truck
Time parameters:	t_{ij}: travel time between stations i and j
	L: loading/unloading time per bike
	T: total dispatch time window

Parameter	Way to Assign Values
N	62 (e.g., total number of stations)
C_i	The value specified in station i
B_i	The value specified in station i
K	3 cases: when there are two ($K = 2$), four ($K = 4$), or six ($K = 6$) trucks in total
Q	28
t_{ij}	The distance between stations i and j divided by 30 km/h
L	0.5 minute per bike
T	2 cases: 30 minutes or 1 hour(60 minutes)

Table 1: Ways to assign values to parameters

Decision Variables

1. **Routing Variables**

   $x_{i,j,k} \in \{0,1\} \quad i,j \in N \cup \{0\},\ i \neq j,\ k \in K$: If truck k travels directly from station i to station j, then $x_{i,j,k} = 1$; otherwise, 0. We let node 0 represent the depot.

2. **Bike Collection and Delivery**

   $a_{i,k} \in \mathbb{Z}^+ \quad i \in N,\ k \in K$: The number of bikes truck k picks up from station i.

   $b_{i,k} \in \mathbb{Z}^+ \quad i \in N,\ k \in K$: The number of bikes truck k drops off at station i.

3. **Station Balance Indicator**

   $y_i \in \{0,1\} \quad i \in N$: If station i is within 30%–70% available dock ratio after dispatch, then $y_i = 1$; otherwise, $y_i = 0$.

4. **Truck Loading State**

   $W_{i,k} \in \mathbb{Z}^+ \quad i \in N \cup \{0\},\ k \in K$: Represents the number of bikes carried by truck k upon leaving station i. For the depot (node 0), we let the model to decide the optimal $W_{0,k}$. We suppose we have infinite inventory (see [4]).

Objective Function

We aim to maximize the number of stations that achieve the balanced state after dispatch:

$$\max \sum_{i \in N} y_i.$$

Constraints

Station Constraints

(a) **Balanced Station Range:**

If station i is counted as balanced, the final number of bikes must lie between 30% and 70% of its capacity:

$$0.3\,C_i - M(1 - y_i) \;\le\; B_i + \sum_{k \in K} \big(b_{i,k} - a_{i,k}\big) \quad \forall i \in N,$$

$$B_i + \sum_{k \in K} \big(b_{i,k} - a_{i,k}\big) \;\le\; 0.7\,C_i + M\big(1 - y_i\big) \quad \forall i \in N.$$

Here, M is a sufficiently large constant. We set

$$M = \max_{i \in N} C_i.$$

(b) **Station Capacity:**

$$0 \;\le\; B_i + \sum_{k \in K} \big(b_{i,k} - a_{i,k}\big), \quad \forall i \in N.$$

$$B_i + \sum_{k \in K} \big(b_{i,k} - a_{i,k}\big) \;\le\; C_i, \quad \forall i \in N.$$

(c) **Visitation-Operation Consistency:**

$$a_{i,k} \;\le\; Q \sum_{h \in N} x_{h,i,k}, \quad b_{i,k} \;\le\; Q \sum_{h \in N} x_{h,i,k}, \quad \forall i \in N,\; k \in K.$$

If truck k does not enter station i, then no $a_{i,k}$ or $b_{i,k}$ operations occur at that station. Conversely, if truck k does visit station i, those operations are allowed. Here, Q is the maximum capacity of a single truck, treated as a sufficiently large constant.

Truck Constraints

(a) **Route Continuity**

- Start and End at Depot:

$$\sum_{i \in N} x_{0,i,k} = 1, \quad \sum_{i \in N} x_{i,0,k} = 1, \quad \forall k \in K.$$

- Visitation:

$$\sum_{h \in N \cup \{0\}} x_{h,i,k} \;\le\; 1, \quad \sum_{j \in N \cup \{0\}} x_{i,j,k} \;\le\; 1, \quad \forall i \in N,\; k \in K.$$

$$\sum_{h \in N \cup \{0\}} x_{h,i,k} \;=\; \sum_{j \in N \cup \{0\}} x_{i,j,k}, \quad \forall i \in N,\; k \in K.$$

- Preventing Subtours:

$$\sum_{i\in S,\, j\in S,\, i\neq j} x_{i,j,k} \;\le\; |S|-1, \quad \forall S\subseteq N,\ |S|\ge 2,\ \forall k\in K.$$

(b) **Total Time Window**:

$$\sum_{i\in N}\sum_{j\in N} d_{i,j}\, x_{i,j,k} \;+\; L \sum_{i\in N\cup\{0\}} \big(a_{i,k}+b_{i,k}\big) \;\le\; T, \quad \forall k\in K.$$

Truck Loading and Capacity Constraints

(a) **Load Flow**:

$$W_{j,k} \;\ge\; W_{i,k} \;+\; a_{j,k} \;-\; b_{j,k} \;-\; Q\big(1-x_{i,j,k}\big), \quad \forall i,j\in N\cup\{0\},\ i\neq j,\ k\in K,$$
$$W_{j,k} \;\le\; W_{i,k} \;+\; a_{j,k} \;-\; b_{j,k} \;+\; Q\big(1-x_{i,j,k}\big), \quad \forall i,j\in N\cup\{0\},\ i\neq j,\ k\in K.$$

If $x_{i,j,k}=1$, then $W_{j,k}$ is updated accordingly.

(b) **Truck Capacity**:

$$0 \;\le\; W_{i,k} \;\le\; Q, \quad \forall i\in N,\ \forall k\in K.$$

Flow Conservation

$$\sum_{i\in N} a_{i,k} = \sum_{j\in N} b_{j,k}, \quad \forall k\in K.$$

Variable Domains

$$x_{i,j,k}\in\{0,1\}, \quad i\neq j, \forall i,j\in N,\ \forall k\in K$$
$$y_i\in\{0,1\},$$
$$a_{i,k},\ b_{i,k}, W_{i,k}\in\mathbb{Z}^{+},$$

Data Collection and Generation

1. Collection Pipeline

We implemented a scheduled collection pipeline using GitHub Actions to obtain realistic and location-specific data of YouBike. We write a `fetch_snapshot.py` to fetch live YouBike station-level data from the Taipei City Government' s open API[1] every 30 minutes. The data include the number of available bikes, empty docks, and station metadata. Each fetch produces a CSV file containing approximately 1,590 rows, corresponding to all YouBike stations in Taipei at that moment. Sample fields from a single data entry are shown in the two tables below:

[1] `https://tcgbusfs.blob.core.windows.net/dotapp/youbike/v2/youbike_immediate.json`

Title	Value
timestamp	2025-05-22T07:04+00:00
sno	500101001
sarea	大安區
sna	YouBike2.0_ 捷運科技大樓站
latitude	25.02605
longitude	121.5436

Table 2: Station metadata

Field	Value
total docks	28
available bikes	3
available docks	25
act	1
srcUpdateTime	2025-05-22 15:03:30

Table 3: Bike availability status

2. Filter Data

The data collection was conducted over a one-week period, from May 23 to May 29, 2025. We captured data every 30 minutes, targeting weekdays only to avoid noise introduced by weekend travel behavior. Ideally, this process would yield up to 240 records (24 hours $\times$ 2 samples/hour $\times$ 5 days), though actual numbers may vary due to network and scheduling fluctuations.

We restricted our analysis to a specific area by filtering the stations based on geographic coordinates, the rectangular bounding box approximately covers the Neihu and Nangang districts with our project goal.

3. Generate Instance

For each station within this region, we computed the weekday average and standard deviation of the number of available bikes.

For model testing, we developed a Python script `generate_instance.py`. This script uses the historical averages (μ_i) and standard deviations (σ_i) for each station i to generate synthetic demand based on normal sampling:

$$b_i := \min(\max(0, X_i), C_i), \quad \text{where } X_i \sim \mathcal{N}(\mu_i, \sigma_i^2),$$

Using these data to evaluate performance across different time periods, we focused on three daily off-peak redistribution windows—9 AM, 5 PM, and 10 PM—and generated 5 instances for each scenario. In total, we produced 30 distinct instance for the whole day.

Performance Evaluation

We compare three different strategies for YouBike redistribution:

- **No Redistribution (Baseline)**: The original station status without any dispatch intervention.
- **Gurobi Optimization**: Using Gurobi to train model by the above objective function and constraints.
- **Heuristic Algorithm**: The result generated from our heuristic algorithm mentioned in previous part.

Gurobi Optimization (and some issue we meet)

Our objective is to maximize $\sum_{i \in N} y_i$, the number of stations whose available dock ratios fall within the desired range (30%–70%). We use this metric as the primary evaluation criterion.

Initially, we attempted to solve instances covering the full area mentioned in the previous section (157 stations). However, even with already a 30-minute time limit per a single instance, Gurobi frequently failed to find an optimal solution within the given time. In such cases, the model attempts to partially dispatch trucks to balance the stations as much as possible, but without completing a full allocation plan. As a result, the computed truck routes and loading decisions can become unstable. We can see an instance after finishing a non-optimal allocation as follow:

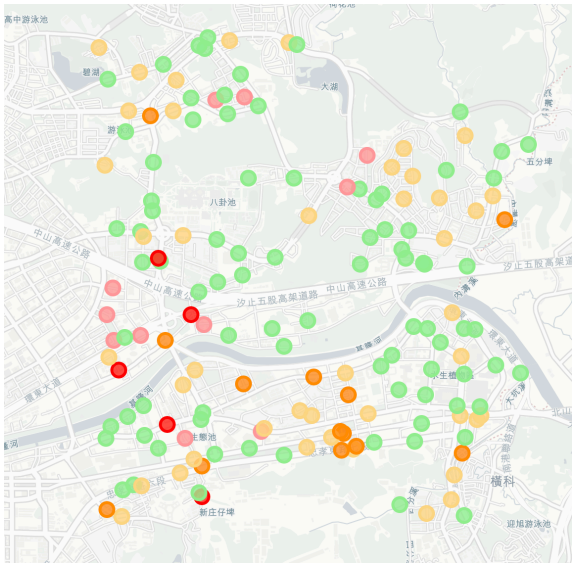


Figure 2: A result in 6-truck-60-min scenario

Moreover, we observed that the LP-relaxation gap remained relatively large for many of these instances (see Table 7), making it difficult to evaluate the quality or interpretability of the returned solutions.

Metric	**Value**
Average MIP Gap	0.5141
Standard Deviation	0.2298
Minimum MIP Gap	0.1111
Maximum MIP Gap	0.9394

Table 4: Summary statistics of MIP gaps across full-area instances

Due to such tragic gap even after running significantly large amount of time, we decided to **restrict our scope further to only covering part of Nangang district**, in order to have relatively more computation time as well as making the problem easier to solve. The corresponding geographic bounds are:

$$\text{Longitude: } 121.591256 \leq \text{lng} \leq 123.00000,$$

$$\text{Latitude: } 25.04615 \leq \text{lat} \leq 25.062016.$$

This sub-region contains approximately 62 YouBike stations, which significantly reduces the problem size. Within this smaller area, Gurobi is able to compute a better solutions within the 30-minute time limit, enabling us to benchmark our heuristic algorithm meaningfully later.

After solving the optimization model and generating route-based outputs, the results are organized under the following directory structure:

Listing 1: Folder structure of optimization results

```
163434_小南港_limit300s_時速60/
    2trucks_30min/
        morning_9am/
            instance_1/
                route.txt
                station_results.csv
        evening_5pm/
        night_10pm/
    2trucks_60min/
    4trucks_30min/
    4trucks_60min/
    6trucks_30min/
    6trucks_60min/
```

- `route.txt`: Table 5 is the example of the sequence of operations performed by each truck. The route begins and ends at the depot (station 0), and intermediate nodes indicate the station ID. A positive number indicates the number of bikes dropped off, a negative number indicates bikes picked up, and an empty pair `()` indicates no operation performed at that station.
- `station_results.csv`: Table 6 is the example of the final status of each station after redistribution, including whether the station is considered balanced and its geographic coordinates.

Truck ID	**Route Summary**
Truck 0	0 → 500111088(-4) → 500111067(-5) → …→ 0
Truck 1	0 → 500111053() → 500111113() → …→ 0
Truck 2	0 → 500111013() → 500111051() → …→ 0
Truck 3	0 → 500111037(+2) → 500108104() → …→ 0

Table 5: Truncated truck routes and operations

Station ID	**Balanced**	**Final Bikes**	**Latitude**	**Longitude**
500108041	1	7	25.06181	121.59447
500108104	1	8	25.06007	121.60138
500108169	1	14	25.06153	121.60033
500108173	1	5	25.06099	121.59662
500111003	1	20	25.05014	121.59238

Table 6: Example rows from `station_results.csv`

Insight from Gurobi Optimization result

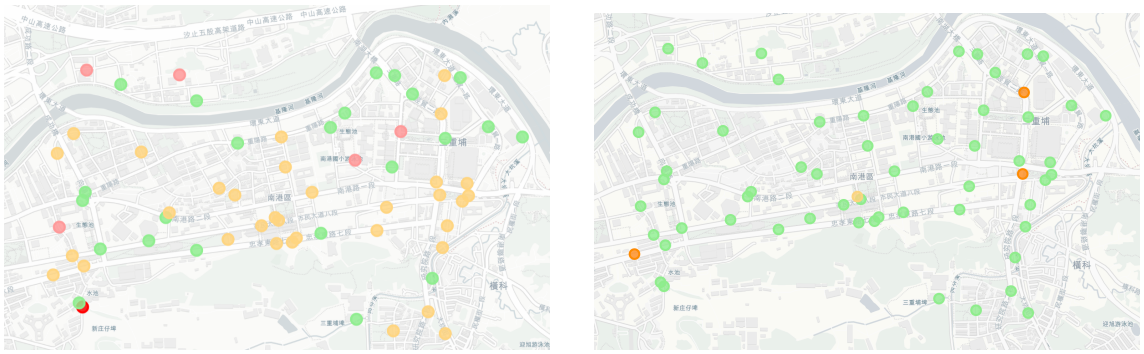
The optimality gap for the smaller area is as follows:

Metric	Value
Average MIP Gap	0.1000
Standard Deviation	0.0884
Minimum MIP Gap	0.0164
Maximum MIP Gap	0.4324

Table 7: Summary statistics of MIP gaps across smaller-area instances

As we can see, in the following chart, the objective value will be higher when having more trucks and allocation time.

Second, we visualize and compare the current station status with the original one, using colored station plots. For instance, below is a morning (9 AM) instance's result after applying to a 6-truck-60-min scenario's solution.



(a) Before redistribution (b) After redistribution

Figure 3: Station status before and after applying the heuristic

Its status is OPTIMAL, meaning the Gurobi optimizer has confirmed that this is the best possible solution. However, several bike stations remain unbalanced. Interestingly, all of these problematic stations are located exactly at Taipei MRT stations. (The only station near MRT that are successfully balanced are at Nangang Station, which makes sense since our depot is located there. On the other hand, another station at Nangang remains unbalanced because it has too many docking slots, making it too difficult to satisfy the threshold constraints.)

Heuristic Solution

Given that our selected region—Nangang—is predominantly a commercial and industrial area rather than a residential neighborhood, we adopt the following assumptions regarding daily traffic patterns to design our heuristic strategy.

Based on our experience in everyday life and some observations from the model, we designed a simple yet effective heuristic method focusing on the core public transportation region: the four

MRT stations and both Nangang TRA and THSR stations. We manually labeled the YouBike stations that are adjacent to MRT or train stations (e.g., "捷運昆陽站 2 號出口" and similar) as "stations," while the remaining ones are categorized as "outskirts.".

Then, We implemented a rule-based strategy depending on the time of day: Inbound and Outbound. The approach differentiates between morning/night inflows and evening outflows as follows:

- **Inbound Mode (9 AM and 10 PM):** In our experience, during the morning and night, there are usually few bikes available to borrow due to the commuting patterns of workers and students. Thus, we can assume a net demand for bikes at central stations. In this mode, each truck starts from the depot, travels to an outskirts location (defined as having more than 70% of its capacity filled with bikes) to load bikes, proceeds to the most under-supplied station (defined by the lowest bike-to-capacity ratio) to unload bikes, and finally returns to the depot. This sequence repeats as long as the total travel and handling time remains within the time window T.
- **Outbound Mode (5 PM):** During the evening peak, workers and students typically return home by MRT or train, often leading to an over-supply of bikes at stations. In this mode, trucks again start from the depot, identify the most overfilled station (highest bike-to-capacity ratio), load excess bikes, transport them to the most under-utilized outskirts location, and return to the depot. This cycle also repeats as long as it stays within the time constraint.

Our implementation uses a fixed truck capacity Q, a handling time L per bike, and a constant driving speed S. The heuristic works as follows:

1. Determine the operational mode based on the time slot.
2. For each truck, repeat the following steps until the available time T is exhausted:
    - **Inbound:**
        (a) Start from the depot.
        (b) Move to the nearest outskirts location that has available bikes; if no such location exists, load from the depot.
        (c) Load up to Q bikes or as many as available.
        (d) Travel to the most under-supplied station.
        (e) Drop as many bikes as possible within the station's capacity.
        (f) Return to the depot.
    - **Outbound:**
        (a) Start from the depot.
        (b) Travel to the most overfilled station (with bike-to-capacity ratio over 0.7).
        (c) Pick up excess bikes, up to a maximum of Q.
        (d) Deliver these bikes to the outskirts location with the most available space (lowest bike-to-capacity ratio).
        (e) Return to the depot.

The algorithm focuses on peak-hour demand, thereby reducing unnecessary trips by directing trucks to the most imbalanced locations and repeating the cycle until the time constraint is met.

Additionally, the output format of this heuristic is identical to that of the Gurobi optimizer described earlier.

Sad result from our Heuristic Algorithm

Our implementation for this part is all inside **run_heuristic.py**. However, the heuristic algorithm we proposed is too naive for the objective result to improve. Sadly, though having some operations by truck, but the number of balanced stations seem to be unchanged. See following snapshot for example.

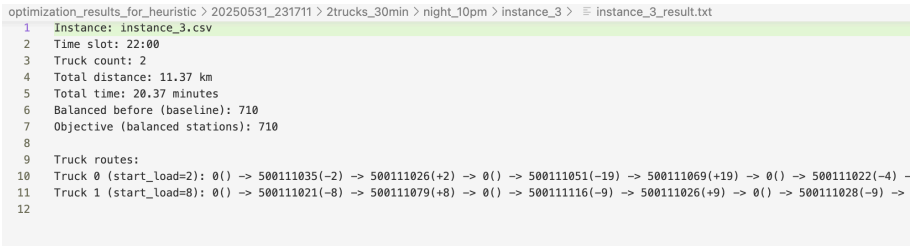


Figure 4: An example of one instance in heuristic result

Comparison & Coclusion

Finally, We also report the numerical results for each ubike stations. Let Z^{gurobi} be the number of balanced stations under the Gurobi solution, and Z^{heu} under the heuristic. We compute the relative performance gap as:

$$\text{Gap} = \frac{Z^{\text{gurobi}} - Z^{\text{heu}}}{Z^{\text{gurobi}}} \times 100\%.$$

We can see how close the heuristic comes to the optimal solution by the below table showing the head of our result. Our evaluation is repeated across 90 synthetic instances, and we summarize both average performance and worst-case deviation.

Instance ID	**Gap (%)**
2trucks_30min	39.68
2trucks_60min	48.26
4trucks_30min	42.90
4trucks_60min	53.52
6trucks_30min	46.76
6trucks_60min	55.77

Table 8: Gap between Heuristic and Gurobi solution for 6 instances. Negative values indicate worse performance by heuristic.

We also compare different scenarios—each defined by a unique number of trucks and time limits —to see how their performance differs across various time periods. The results are summarized in the following table:

Scenario	09:00 (AM)	17:00 (PM)	22:00 (Night)
2 trucks, 30 min	66.77%	79.03%	69.03%
2 trucks, 60 min	77.10%	89.35%	80.32%
4 trucks, 30 min	74.19%	84.52%	68.71%
4 trucks, 60 min	88.39%	**98.71%**	89.68%
6 trucks, 30 min	76.45%	87.74%	76.77%
6 trucks, 60 min	**95.16%**	98.06%	**97.74%**

Table 9: Average balance ratio for each scenario across time periods

We observe that:

- Longer time windows and more trucks consistently lead to higher balance ratios.
- The best performances occur under 4 or 6 trucks with 60-minute windows.
- The weakest performance appears in the most resource-constrained configuration: 2 trucks with 30 minutes.

In this project, we have finished following things:

- **Scheduled collection pipeline**: To obtain Taipei City Government's public data for long period, we use GitHub Action to load it automatically.
- **YouBike visualizer with map and threshold indicationa**: we can process all raw data, restrict our concerning area, and calculate threshold for all stations. Then, output a HTML file for visualization.
- **Gurobi**: Using simplex method to try our best to find the optimal solution.
- **Heuristic Solution**: Performing our proposed heuristic above.

All of our source code and details are available at: `https://github.com/Ocean1029/OR_final`. Feel free to check it out!

References

[1] Taipei City Government Open Data, `https://citydashboard.taipei/mapview?index=youbike`.

[2] The News Lens, `https://www.thenewslens.com/article/186604`

[3] The Reporter, `https://www.twreporter.org/a/data-reporter-taipei-youbike-free-ride-for-the-first`

[4] Taipei City Council, `https://www.tcc.gov.tw/Default.aspx`