Data Science Practice : Shenzhen Metro System Optimization

Cheng Guo
Southern University of Science and Technology
Shenzhen, China
12112504@mail.sustech.edu.cn

Qijia He
Southern University of Science and Technology
Shenzhen, China
heqj2021@mail.sustech.edu.cn

Lu Wang
Southern University of Science and Technology
Shenzhen, China
12113041@mail.sustech.edu.cn

Lingmin Yan
Southern University of Science and Technology
Shenzhen, China
12111345@mail.sustech.edu.cn

Jiachen Zhou
Southern University of Science and Technology
Shenzhen, China
12011915@mail.sustech.edu.cn

ABSTRACT

The project aims to optimize the operational efficiency of Metro Line 5 by precisely selecting station stopping patterns, alleviating congestion and passenger waiting times during peak hours. Through exploratory data analysis, a subway system model based on simplified assumptions was established, and various algorithms, including greedy search and local search, were designed to optimize train stopping plans. The experimental section used methods based on real and simulated data to verify the effectiveness of the algorithms under different conditions and conducted sensitivity analysis. The results show that the model and algorithms can significantly improve the transportation efficiency of the subway system, potentially reducing total commuter time by approximately 15% during peak hours and effectively dispersing passenger flow at key stations. Additionally, the report discusses future work directions, including improving the greedy algorithm, establishing a more comprehensive subway coordination system model, and regularizing the algorithms to better adapt to travel habits. Ultimately, this study demonstrates the potential of data-driven solutions to enhance public transportation system efficiency and provides an optimization blueprint for similar urban transportation challenges globally.

1 INTRODUCTION

Urban transportation systems play a crucial role in the daily lives of city dwellers, especially in densely populated metropolitan areas like Shenzhen. The efficient movement of people is essential for the economic vitality and overall quality of life in the city. However, with rapid urbanization and increasing population, Shenzhen's metro system faces significant challenges during peak hours. Overcrowding and long waiting times are common issues that need addressing to improve commuter experience and system efficiency.

The Metro Line 5 is the main line running through Shenzhen from east to west. It consistently ranks first in passenger volume across the entire network, making it the busiest metro line in Shenzhen. Therefore, this project provides a strategic analysis based on the operation of Shenzhen Metro Line 5. It aims to tackle these problems by optimizing the selection of metro stop points, akin to an express service model. By strategically designating certain trains to skip specific stations during peak hours, we seek to minimize total commuting time across the network. This approach can alleviate congestion at critical junctures, reduce travel time for long-distance commuters, and balance passenger loads more effectively across the system.

The core objective of this project is to develop a model that identifies optimal stop patterns for metro trains during morning and evening rush hours. By integrating various data points, such as passenger flow statistics, station capacity, and transit schedules, the model will propose a stop selection strategy that enhances overall system performance. We will employ simulation techniques and optimization algorithms to validate the proposed solution and ensure its feasibility in real-world applications.

This introduction provides a foundational understanding of the project's goals and the significant impact it aims to achieve. The following sections will detail the data analysis, algorithm design, model development, and results, ultimately offering insights into how strategic stop selection can transform Shenzhen's metro system into a more efficient and commuter-friendly network.

2 RELATED WORK

In the field of subway system operation management and optimization, various innovative models and algorithms have been proposed in recent years to enhance efficiency, reduce passenger travel time, and lower operational costs and environmental impact. The research primarily focuses on the following areas:

First, some researchers established a comprehensive optimization model, including train scheduling, stop patterns, and timetables to meet passenger demand, shorten travel time, and reduce operational costs. Based on a study of a Shanghai subway line, the researchers designed a two-stage approximation algorithm to solve the express and local train operation plan model[2].

During operations, inevitable situations such as bidirectional subway line interruptions occur. To address the uncertainty in interruption duration, researchers have proposed a train operation adjustment model[3]. This model constructed a mixed-integer nonlinear programming model and employed a two-stage method and rolling time-domain optimization algorithm, using optimistic and pessimistic prediction strategies to achieve dynamic adjustment of the train operation plan.

To increase the attractiveness and coverage of urban rail transit, a study proposed an optimization method for express and local train stop patterns. By applying a grey weighted clustering model to cluster the stations, they constructed a 0-1 nonlinear programming model and used a genetic simulated annealing algorithm to improve solution efficiency[4].

Lastly, to tackle the issue of passenger congestion during peak hours, a coordinated optimization method for subway network flow control and train schedules was proposed. It established a coordinated optimization mixed-integer nonlinear programming model for flow control and train schedules, and designed an improved differential evolution algorithm and elite retention genetic algorithm to solve the model, thereby enhancing system operational efficiency and reducing passenger travel delays[1].

Overall, the existing related work primarily assumes a uniform distribution of passenger flow and considers the one-way operation of the subway, combined with integer programming for solution. However, we have made further adjustments on this basis, refining the distribution of passenger flow over time, and considering the bidirectional operation of the subway. We have adopted a dynamic optimization strategy that is centered on the greedy algorithm and combined with the simulated annealing algorithm. This approach can more accurately reflect the actual situation, enhancing the operational efficiency of the subway system and passenger satisfaction.

3 DATA INTRODUCTION AND EXPLORATORY DATA ANALYSIS

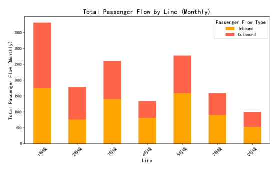
3.1 Data Introduction

Data we used in the section consists of three datasets. The first one contains the location of each metro station in ShenZhen, including longitude and latitude. The next dataset contains close to a million metro records in Shenzhen, from August 30th to September 1st in 2018. Due to the privacy regulations of Shenzhen Metro, we are not able to fetch the hottest records. The last part is from the official website of Shenzhen Metro, which gives information on the monthly passenger flow of each line in August, 2018.

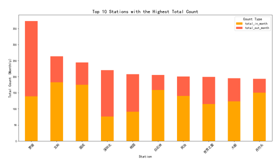
3.2 Exploratory Data Analysis

Based on the previous datasets, we conduct an exploratory data analysis to depict a basic view of the subway system in 2018 which might shed light on our further modeling and experiment.

The monthly passenger flow with proportion of entry and exit is shown in 11. Similarly, the volume and ratio of entry passenger flow and exit passenger flow of the top-10 stations ranked by total currency is shown in 12. The graphs suggest considerable differences in both volume and ratio, whether they are categorized by line or by station. This indicates that situation can vary fiercely for different lines and stations, providing a solid prove for the potential improvement with subway schedule modification.

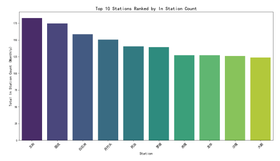


1 Line

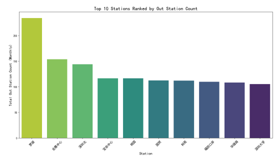


2 Station

For complementary, the top-10 stations ranked by entry passenger flow and exit passenger flow are plotted in 13 and 14, which also provide evidence for different passenger flow situations by station.

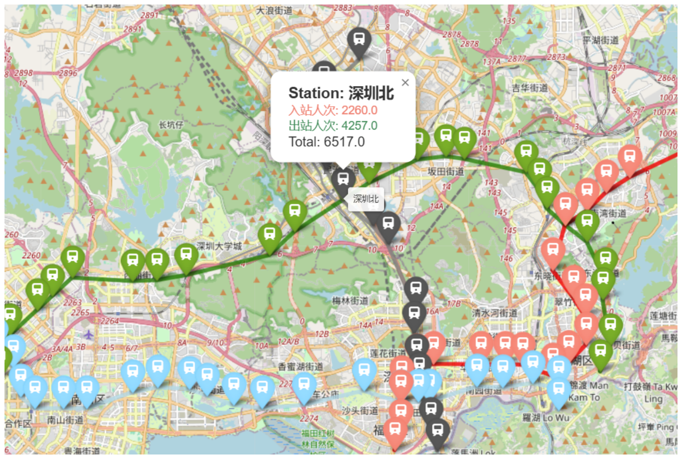


3 In

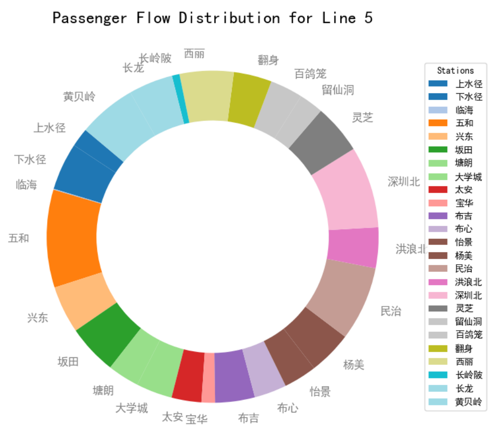


4 Out

Subsequently, we visualize the subway system in Shenzhen with the passenger flow situation of each station in 15. Line 1, line 3, line 4, and line 5 are chosen for visualization as they are the four lines with the highest passenger flow. Lines are marked on the map with different colors, and passenger flow situations of stations are also inserted.



Finally, we summarized the passenger flow situation of the entire line using Line 5 as a case study. Passenger flow situation of Line 5 by station is shown in 16. Stations like Shenzhen North Station and MinZhi have a high volume of passenger flow, while stations with almost no passenger flow like BaoHua also exist. In subsequent algorithm tests and experiments, We will also use the passenger flow situation of Line 5 as a reference for the real-world scenario.



6

4 THE MODELING PROCESS OF THE SUBWAY SYSTEM

4.1 Basic assumptions

Before delving into the exploration of optimization algorithms, we need to construct the subway model system. For the sake of simplification, we have made several assumptions:

- If a station is skipped, the time saved remains constant for each station.
- The distance between each neighboring station is constant.
- Subway crashes are not considered since we are optimizing scenarios with two subway railroads.
- There is a fixed capacity for each subway. This allows us to discuss situations where too many people are entering the subway.
- Passenger's commuting time starts to accumulate right after it step into the station, and stop when it arrived.
- the passenger can quickly step into the subway right after it step into the station (if there's a subway parking in the station at the same time).
- all subways start from the same terminal station.(easier for initialization).

4.2 Parameters for the subway model

To operate a subway system based on the proposed assumptions, we need to declare a few parameters:

- `station_interval`: The time required for a subway to travel between two neighboring stations when a station is skipped.
- `skipped_saved`: The time saved when a station is skipped.
- `station_N`: The number of stations in the subway system.
- `train_N`: The number of subways available for the subway line.
- `passengers_max`: The maximum capacity of each subway.

For our subway model system, we have used 2 different parameter sets in pure simulated data and Shenzhen simulated data. the details of which is summarized in the table:

parameters	pure simulated	Shenzhen data
station_interval	2	1.5
skipped_saved	1.5	1.2
station_N	20	25
train_N	28	27
passengers_max	110	75
N	20,000	10,000

Table 1: parameter table

The passengers_max parameter is set relatively small compared to real-world data to expedite computation time. Since passenger traffic per unit time has a direct relationship with passengers_max, reducing both parameters simultaneously should not greatly change the result.

4.2.1 Subway System Initialization. Before passengers arrive, the entire subway system should be initialized, meaning that the subway system should already be operating normally. To initialize the system, subways should start operating a few hours before passengers arrive. The departure interval is set so that when all subways depart, the distance interval between each subway remains the same. Hence:

$$departure\_interval = \frac{2station\_N * (time\_skipped + station\_interval)}{train\_N}$$

The optimization algorithm begins to work when it detects that passengers are arriving.

4.3 Model inputs and outputs

Since this project is to optimizing subway schedule based on passengers flow. The input and output are very clear: which are passenger's flowing data and train schedule respectively. The dataframe of the input passengers flow and output schedule dataframe is shown as below:

ID	start	end	time
0	2	7	110
1	9	18	23
2	16	13	152
...	...	...	...
N-1	16	10	152

Table 2: input passengers flow

station_0	station_1	station_2	...	2*station_N
0	1	1	...	1
1	0	1	...	1
1	0	1	...	0
...	...	...	...	...
1	1	0	...	1

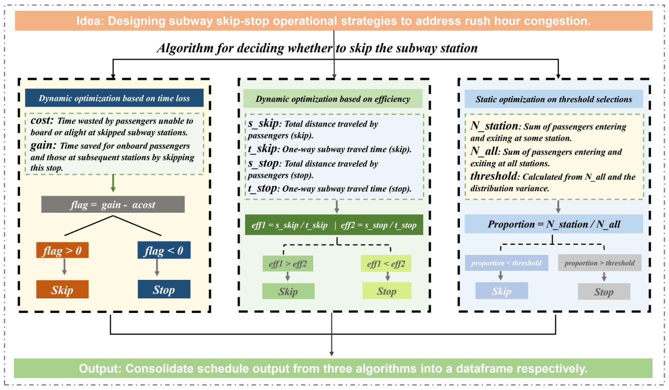
Table 3: output subway schedule

In our system, each passenger is represented by their entry station ID (start) and their destination station ID (end). Additionally, we have the exact time (time) at which the passenger entered the station. To calculate their commuting time, we need to determine the time at which they exit the system based on the output schedule. Time is measured in minutes, and in the Shenzhen data, 0 corresponds to 6:00 a.m.

The output schedule is represented by a binary matrix where 1 indicates a stop and 0 indicates a skip. Each row represents the schedule for one train. Our project also includes other useful information, such as the arrival time of each train (train_i) at each station (station_j).

5 ALGORITHM DESIGN

5.1 Greedy Search Algorithm



Idea: Designing subway skip-stop operational strategies to address rush hour congestion.
Algorithm for deciding whether to skip the subway station
Dynamic optimization based on time loss
cost: Time wasted by passengers unable to board or alight at skipped subway stations.
gain: Time saved for onboard passengers and those at subsequent stations by skipping this stop.
flag = gain - αcost
flag > 0
flag < 0
Skip
Stop
Dynamic optimization based on efficiency
s_skip: Total distance traveled by passengers (skip).
t_skip: One-way subway travel time (skip).
s_stop: Total distance traveled by passengers (stop).
t_stop: One-way subway travel time (stop).
eff1 = s_skip / t_skip | eff2 = s_stop / t_stop
eff1 > eff2
eff1 < eff2
Skip
Stop
Static optimization on threshold selections
N_station: Sum of passengers entering and exiting at some station.
N_all: Sum of passengers entering and exiting at all stations.
threshold: Calculated from N_all and the distribution variance.
Proportion = N_station / N_all
proportion < threshold
proportion > threshold
Skip
Stop
Output: Consolidate schedule output from three algorithms into a dataframe respectively.

The basic idea of our dynamic algorithm is that for any subway, it is assumed that the subway will pass through each station before departure, and when it arrives at the station, we will generate the information about the passengers carried in the full journey after skipping station and not skipping station, and decide whether to skip station or not by comparing the information.

5.1.1 Dynamic Algorithm 1. Algorithm 1 proceeds as follows: First initialize the train schedule to stop at all stations. Then,
1. Iterate through all the stations and calculate the time lost and gained by passing through the stations and not passing through the stations.
Gain is calculated from the saved time of all passengers on the train. Lost is the number of people who didn't make it to that station (i.e., the sum of the number of people who would have gotten off at that station and got on at that station) * waiting time
2. If the gain - penalty parameter * cost is greater than 0, skip the stop
3. Repeat 1-2 until there are no station that can be skipped.

Algorithm 1 Basic Dynamic Algorithm Architecture

```
Require: initial_schedule
Ensure: optimized_schedule
1: Initialize schedule with all stations as stops
2: $stop\_stations \leftarrow$ list(np.ones($stations\_N \times 2).astype(int)$))
3: $optimized\_schedule \leftarrow$ initial_schedule
4: for $i \leftarrow 0$ to $stations\_N * 2$ do
5:     $stop\_stations[i] \leftarrow 0$
6:     $next\_schedule \leftarrow$ next_train($stop\_stations$)
7:     $passengersInTrain \leftarrow$ get_df_ofPassengers(
8:         $next\_schedule, stop\_stations$)
9:     $saving\_time \leftarrow$ calculate_savings($optimized\_schedule,$
10:        $next\_schedule, passengersInTrain, stop\_stations$)
11:    if $saving\_time > 0$ then
12:        $optimized\_schedule \leftarrow next\_schedule$
13:    end if
14: end for
15: return $optimized\_schedule$ =0
```

Algorithm 2 Calculate Savings 1 (based on time loss)

```
Require: $df\_before, df\_after, drop\_station, para$
Ensure: $time\_saved$
1: $cost \leftarrow$
2:     $(df\_before.shape[0] - df\_after.shape[0]) \times stations\_N \times 2 \times (skipped\_saved + station\_interval)/train\_N$
3: $gain \leftarrow$
4:     $df\_after[(df\_after['end\_'] > drop\_station)].shape[0] \times skipped\_saved$
5: return $gain - cost \times para$ =0
```

5.1.2 Dynamic Algorithm 2. The idea is basically the same as that of Algorithm 1. The difference lies only in the difference of the indicators for judgment. Algorithm 2 we use the gain and loss of efficiency as indicators, the following two equations give the definition of the gain before and after skipping station respectively.

$$eff_{after} = \frac{\text{Total Distance}}{\text{One-way Travel Time (after skipping stations)}} \quad (1)$$

$$eff_{before} = \frac{\text{Total Distance}}{\text{One-way Travel Time (no skipping stations)}} \quad (2)$$

If eff_{after}-eff_{before} is greater than 0, skip this station.

5.1.3 Static Algorithm. The initial idea behind this algorithm was to stop at stations with fewer people and stop at stations with more people. Therefore, this algorithm can respond to the overall distribution of the number of people, without relying on the distribution in different time periods, and this is the way that most subway system used to improve its efficiency. Its simple structure can achieve a certain improvement effect when the number of people at each station is uneven, and generate a stable schedule throughout the

Algorithm 3 Calculate Savings 2 (based on effciency)

```
Require: $df\_before, df\_after, drop\_station, drop\_N$
Ensure: $time\_saved$
1: $before\_eff \leftarrow$
2: $\frac{(df\_before['end']-df\_before['start']).sum()}{(skipped\_saved+station\_interval)\times N-drop\_N\times skipped\_saved}$
3: $after\_eff \leftarrow$
4: $\frac{(df\_after['end']-df\_after['start']).sum()}{(skipped\_saved+station\_interval)\times N-(drop\_N+1)\times skipped\_saved}$
5: return $after\_eff - before\_eff$ =0
```

entire process. Three express trains with the same plan are followed by a slow train that stops at all stations, and the stopping plan of each express train is the same under the same set of data. However, due to its reliance on the proportion of people, if the number of people at each station is uniform, or if there is a maximum value that affects the overall proportion, the algorithm may not be able to produce optimization results and generate the original schedule for all stops.

Algorithm 4 Station Schedule Calculation

```
Require: $df\_passengers, c$ (a constant used in threshold calculation)
1: $df\_sum \leftarrow$ sum of total passengers for each station
2: $total \leftarrow \sum(df\_sum)$
3: $threshold \leftarrow \frac{\text{standard deviation of } df\_sum}{total\cdot c}$
4: $ratio[i] \leftarrow \frac{df\_sum[i]}{total}$   for each station $i$
5: $df\_schedule \leftarrow$ dataframe initialization, same length as $df\_sum$

6: for each station $i$ in $df\_sum$ do
7:     if $ratio[i] < threshold$ then
8:         $df\_schedule[i] \leftarrow 0$
9:     else
10:        $df\_schedule[i] \leftarrow 1$
11:    end if
12: end for
13: return $df\_schedule$ =0
```

The details of the algorithm is described below:

1. Calculate the total number of people at each station across time(including entry and exit, such as 75 people entering, 30 people exiting, and ultimately 105).

2. Calculate the total number of people and variance of all stations, determine a threshold (such as 1%, the specific way to determine the threshold is through a formula, and after trying multiple thresholds each time, select the best one). This procedure has a hyperparameter c, often used search space value: [0.05, 0.1, 0.2, 0.5, 5, 20]. When search finds 20 as the optimal, it often indicates that this algorithm has no improvement and schedule is same as origin.

3. If the ratio of the number of people at all stations to the total number of people is less than this value, do not stop. Determine the parking plan

4. Run a slow train every 3 trains(all stops) to ensure every passenger is transported.

5.2 Local Search Algorithm

After implementing the three algorithms introduced above, we may obtain an optimized schedule based on passenger flows. However, we are uncertain about the extent of our optimization and whether there are further improvements possible. Therefore, we require another approach to enhance the schedule. Local Search is implemented in this project.

5.2.1 Why local search? In the realm of optimization, numerous advanced greedy optimization methods, such as Simulated Annealing, Genetic Algorithm, and algorithms based on Neural Network have been proposed to address real-world problems. However, while these algorithms often yield relatively optimal solutions, they demand extensive time to execute multiple iterations. Given the structure of my subway system, calculating the total waiting time for 100,000 passengers takes approximately 2.5 minutes. This implies that 1000 iterations could consume 41.7 hours! Furthermore, the schedule for the entire morning consists of a 0-1 matrix with over 2,500 elements in total. Iterating thousands of times poses a significant computational burden. Considering computational constraints, Local Search emerges as a favorable choice. Moreover, experiments indicate that with 3,000 iterations, it does not become trapped in local optima.

5.2.2 Implementation of Local Search. The implementation of local search optimization entails examining the neighbors of the current schedule. If a neighbor is found to be better, the current schedule is updated to this neighbor; otherwise, the current schedule remains unchanged. The optimization function is the total amount of time that passengers spend on commuting. Neighbors are defined based on the Hamming distance of the schedule matrix. In this project, the maximum searching distance is set to be 3.

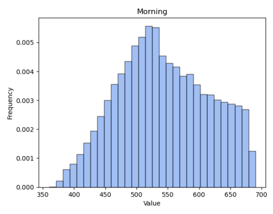
Algorithm 5 Local Search Algorithm

```
Require: initial_schedule, iterations
Ensure: current_schedule
1: $current\_schedule \leftarrow initial\_schedule$
2: for $i \leftarrow 1$ to iterations do
3:     $neighbor \leftarrow$ generate_neighbor($current\_schedule$)
4:     if evaluate($neighbor$) < evaluate($current\_schedule$) then
5:         $current\_schedule \leftarrow neighbor$
6:     end if
7: end for
8: return $current\_schedule$ =0
```

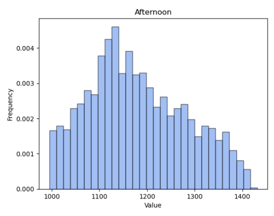
6 EXPERIMENTS

6.1 Simulated Data Generation

6.1.1 Based on Real Data. Based on the analysis of the 2018 Shenzhen Metro passenger flow data, we observed that the metro's morning and evening passenger volumes follow a normal distribution, as depicted in the corresponding figures. We use the parameters from this distribution for subsequent data sampling. However, since the existing data includes weekend morning travel patterns, which generally occur later than on weekdays, we adjusted the mean of the distribution according to actual conditions and official Shenzhen data, while keeping the distribution's variance unchanged.



7 histogram



8 Coefficients

The overall data simulation sampling is divided into two main parts: time sampling and station sampling.

Given the constraints on the sampling period, we applied a truncated distribution for time sampling. The probability distribution, initially defined over an infinite range, was compressed into a fixed time interval while maintaining a total probability of 1 within this interval. The specific distribution is as follows.

$\Psi\left(\overline{\mu}, \overline{\sigma}, a, b; x\right)$ represents a truncated normal distribution where $\overline{\mu} and \overline{\sigma}$ represent mean and variation of general normal distribution; and a, b represent truncated interval.

$$\Psi\left(\overline{\mu}, \overline{\sigma}, a, b; x\right) = \begin{cases} 0 & x \leqslant a; \\ \dfrac{\phi\left(\overline{\mu}, \overline{\sigma}^2; x\right)}{\Phi\left(\overline{\mu}, \overline{\sigma}^2; b\right) - \Phi\left(\overline{\mu}, \overline{\sigma}^2; a\right)} & a < x < b \\ 1 & b \leq x \end{cases}$$

$$\Psi\left(\overline{\mu}, \overline{\sigma}, a, b; x\right) = \begin{cases} 0 & x \leqslant a; \\ \dfrac{\Phi\left(\overline{\mu}, \overline{\sigma}^2; x\right) - \Phi\left(\overline{\mu}, \overline{\sigma}^2; a\right)}{\Phi\left(\overline{\mu}, \overline{\sigma}^2; b\right) - \Phi\left(\overline{\mu}, \overline{\sigma}^2; a\right)} & a < x < b \\ 1 & b \leq x \end{cases}$$

$\phi\left(\cdot\right)$ & $\Phi\left(\cdot\right)$ represent probability density function and cumulative density function of standard normal distribution.

Ultimately, the sampled time period is limited to 06:00–11:30 in the morning and 16:30–00:00 in the afternoon and evening. The start time of 06:00 corresponds to the metro's commencement of service, 11:30 is the latest time recorded in the 2018 data, 16:30 is the earliest time recorded for the afternoon, and 00:00 corresponds to the metro's closing time.

For station sampling, we calculated the sampling probabilities based on actual data: in 2018, there were 25 metro stations. The proportions of passenger flows at each station were used to determine the sampling probabilities. The stations, listed from Linhai to Huangbeiling in sequence, form the following dictionary for sampling purposes:

{*Linhai*: 0, *Baohua*: 1, *Fanshen*: 2, *Lingzhi*: 3, *Honglangbei*: 4, *Xingdong*: 5, *Liuxiandong*: 6, *Xili*: 7, University Town: 8, Tanglang: 9, *Changlingpi*: 10, *Shenzhen North*: 11, *Minzhi*: 12, *Wuhe*: 13,*Bantian*: 14, *Yangmei*: 15, *Shangshuijing*: 16, *Xiashuijing*: 17, *Changlong*: 18, *Buji*: 19, *Baigelong*: 20, *Buxin*: 21, *Tai'an*: 22, *Yijing*: 23, *Huangbeiling*: 24}

This dictionary aids in subsequent sampling. After excluding cases where the entry and exit stations are the same, we sampled entry and exit stations according to the calculated probabilities. Finally, an entry time was assigned to each record.

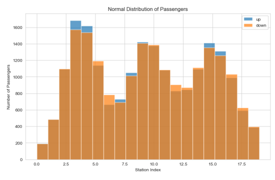
Due to the original data only providing afternoon passenger flow statistics for Shenzhen North Station and Changlong Station, it is challenging to infer the afternoon passenger flow distribution accurately. Therefore, we made a simple assumption: the passenger volume remains consistent between the morning and evening peaks, with opposite directions. For example, if 1000 passengers travel from Station A to Station B in the morning, then 1000 passengers will travel from Station B to Station A in the evening. In simple terms, the afternoon entry and exit patterns are the reverse of the morning samples: if the morning flow is A-B, then the afternoon flow is B-A.

6.1.2 Not Based on Real Data. In order to first test the performance of our algorithm in a small number of different scenarios, we artificially specified two distributions to generate simulated data. The inputs required by our algorithm consist of only three columns, entry station ID (start), destination station ID (end) and the time (time) at which the passenger entered the station. The time we simulated using a uniform distribution.

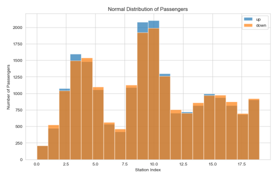
For the distribution of passenger flow at the station, we generated a set of simulated data using a uniform distribution and two sets of data using a Gaussian mixture (denoted as normal and normal1 , respectively).

Take Gaussian mixture generation as an example, we defined the mixture number of Gaussian distribution as 3, and then fixed the center of the Gaussian distribution as [4, 10, 16], randomly selected the variance, and finally selected two representative data sets. The first set of Gaussian Mixture data has the three centers more clearly represented, but the peaks of the three centers do not differ much. The second set of Gaussian mixing has a difference in the peaks of the three centers. We use this to simulate the distribution of passenger flow at different stations in reality.

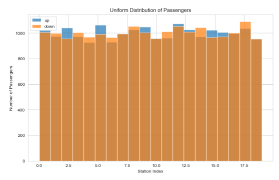
The total number of data we simulate are both 20000.



9 First Normal distribution data



10 Second Normal distribution data



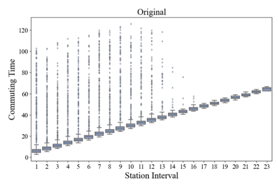
11 Uniform distribution data

6.2 Experiment Results and Performance Comparison

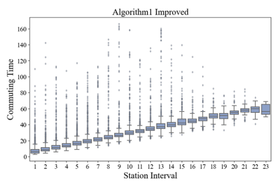
As described in 6.1, We have conducted 2 experiments: Reasons for conducting pure simulated data experiment is that, Shenzhen's data only represents one distribution, but we need to test the robustness and sensitivity of the algorithm performances in different situations.

- Experiments in Shenzhen subway's 5th line's simulated data.
- Experiments in pure simulated data

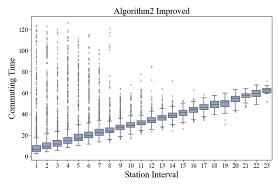
6.2.1 Experiments in Shenzhen subway's 5th line's simulated data. First I show the results of the algorithms in shenzhen subway data.



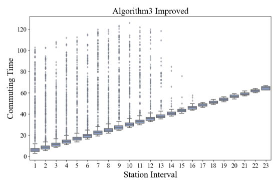
12 Original



13 Algorithm 1



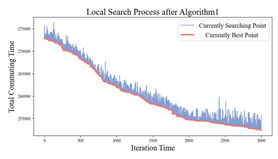
14 Algorithm 2



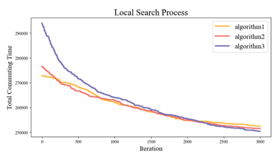
15 Algorithm 3

As depicted in the picture above, it's evident that as the station interval increases, the commuting time generally tends to rise. However, the box plots exhibit numerous outliers for each algorithm, primarily due to the limited capacity of trains. This results in some passengers having to wait, especially those whose starting station is in the middle of the line. Nonetheless, the outcomes vary across different algorithms: algorithm 1 and algorithm 2 yield total savings of 7.2% and 5.8% respectively, while algorithm 3 shows no improvement at all.

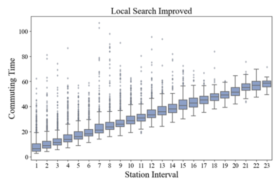
Subsequently, local search is implemented based on the greedy schedule. The following picture illustrates the search process and provides a general analysis of the results obtained from local search. For simplicity, detailed results for only the first algorithm are presented.



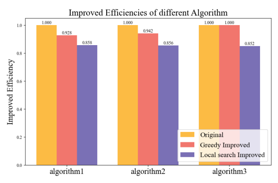
16 algorithm1, LS procedure



17 LS procedures of the 3 algorithms



18 final performance, algorithm 1

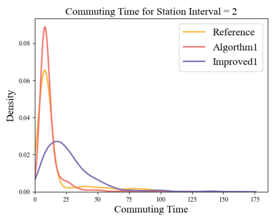


19 final performances

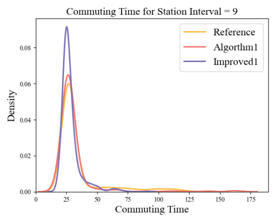
The initial picture illustrates the search process based on the schedule generated by algorithm 1. It begins to marginally converge after 2,500 iterations, albeit with a gradual decline. By 3,000 iterations, as depicted in figure 17, gradually converge, resulting in a total saved time of 14.2%. While greedy algorithms do improve the schedule, there remains a vast optimization space to explore.

The local search process based on three greedy algorithm's schedules is depicted in figure 17. Regardless of the starting point, local search consistently elevates the total saved commuting time to approximately 15%, suggesting that the global optimum should lie between 80% to 85%.

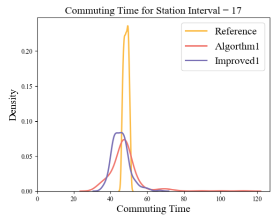
Examining the box plot in detail, the distribution of commuting time may vary across different algorithms for each station interval. Here, we selected intervals of 2, 9, and 17, representing short, medium, and long subway trips, respectively. We then generated KDE plots to discern the differences before and after applying the algorithms.



20 Station Interval = 2



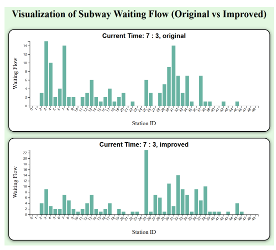
21 Station Interval = 9



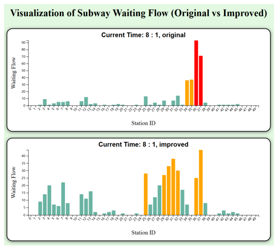
22 Station Interval = 17

There isn't a significant disparity between algorithm1 and the LS-improved version. However, the gap between the original algorithm and the improved(both LS and algorithm1) results is substantial. For short trips, there's a modest enhancement in both peak and mean values, primarily because shorter trips rarely benefit from the "skip station" feature. However, as the interval between trips increases, such as with interval=9, both the peak and mean values begin to shift left. With very long trips, the improvements facilitated by the skipped station schedule become strikingly evident. The variances of the improved algorithm's schedules undeniably escalate due to the uncertainties introduced by skip stations for passengers.

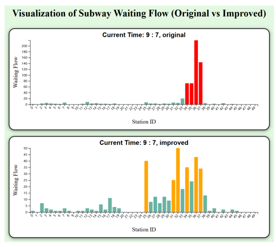
To better comprehend the impact of this algorithm, we developed two HTML files. One visualizes the change in passenger numbers at each station over time, while the other illustrates the movement of trains on Line 5th over time.



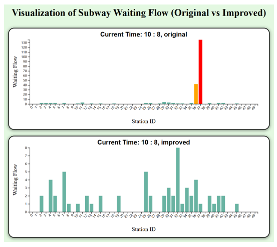
23 passenger flows, 7:00



24 passenger flows, 8:00

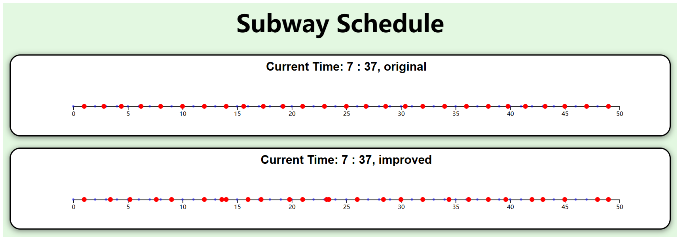


25 passenger flows, 9:00



26 passenger flows, 10:00

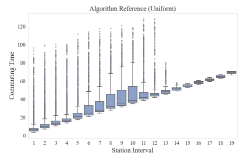
The details of the moving figure can be found in station passengers vis.html. At 7:00 a.m., only a few people are at the subway stations, and the system operates smoothly. As time progresses, more and more people emerge. By 8:00, the original schedule begins to strain, causing some stations to become overwhelmed, while the improved schedule manages to maintain normal operations, with only a few stations experiencing crowding. By 9:00, the original schedule exceeds the station's capacity, with over 200 passengers waiting at some stations, whereas the improved schedule continues to function effectively. By 10:00, the original schedule still sees a large number of passengers waiting for trains, whereas the improved schedule has returned to normal operation. This clearly demonstrates the superior performance of the newly arranged schedule when faced with a high volume of passengers.



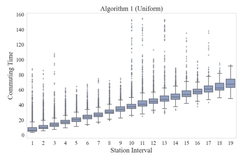
This visualization showcasing the movement of trains within the subway system is quite intriguing. More details and animations can be explored in the subway schedule vis.html file. It offers a clear depiction of how the subway operates both before and after the algorithm is applied. Interestingly, the improved version occasionally exhibits instances of subway overtaking, highlighting the efficiency gains brought about by the algorithmic enhancements.

6.2.2 Experiments in pure simulated data. In pure simulated data, more researches can be done, since we can construct and modified the passengers flow data to hundreds of distributions. As described in 5.2, we generates 3 data in total (2 based on normal distribution and one based on uniform distribution), First we test the performance of the algorithms in this data, and then fixed the time and did algorithm's output pattern analysis, Finally, we largely modified the parameters of the system, and did algorithm sensitivity analysis.

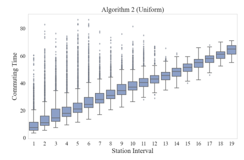
Now we show the performances by giving boxplot and kde plot, due to limited spaces, I only showed the result from the uniform data with 3 different algorithms. The result of the other 2 can be find in the Appendix.



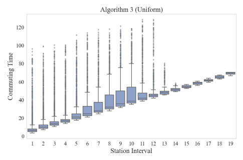
27 Original



28 Algorithm 1

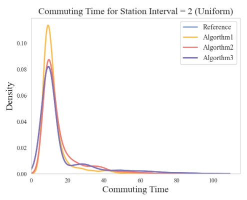


29 Algorithm 2

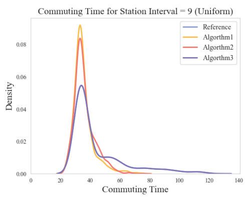


30 Algorithm 3

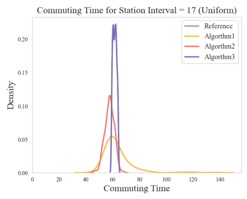
The algorithm3 do not has too much improvements in uniform distribution, similar to shenzhen data's performance. This is due to the underlying algorithmic principles, and the performance of the All-stop approach is quite commendable when dealing with uniform distribution. In algorithm 1 & 2, little improvements is shown: Algorithms 1 increases the deviation degree of outliers while algorithm 2 decreases it. However, compared to the non-uniform data, the total improvements of these algorithms are not significant.



31 Station Interval = 2

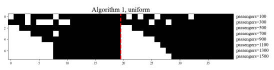


32 Station Interval = 9

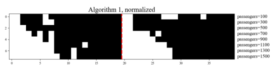


33 Station Interval = 17

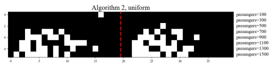
Later, we aim to examine how different algorithms generate varying schedules for a single train when provided with the same input. We utilize simulated data generated from both 'uniform' and 'normal' distributions. Specifically, we set the time of all passengers to 0 and randomly sample k passengers. In our experiments, k varies from a list of values: [100, 300, 500, 700, 900, 1100, 1300, 1500]. This approach allows us to observe how algorithms behave when confronted with different levels of passenger flow per unit time.



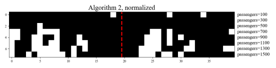
34 Algorithm 1, uniform



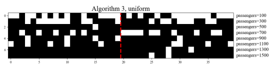
35 Algorithm 1, normal



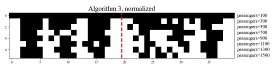
36 Algorithm 2, uniform



37 Algorithm 2, normal



38 Algorithm 3, uniform

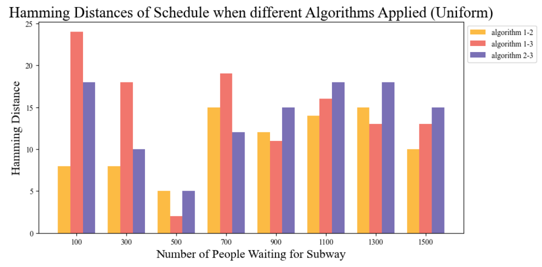


39 Algorithm 3, normal

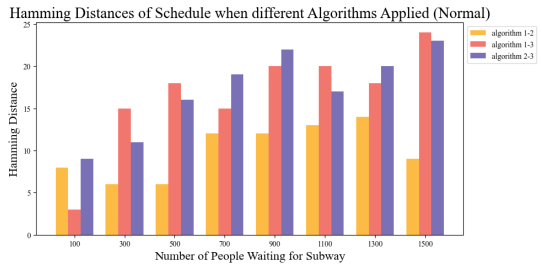
In the six tables above, the left column represents passengers sampled from uniform simulated data, while the right column represents passengers sampled from normal distributed data, with each row corresponding to the number of sampled individuals. For algorithm 1, as the number of individuals increases, the probability of skipping stations also increases, primarily skipping stations at the front. When the number of individuals is very large (e.g., 1500), almost half of the stations are skipped directly. This is related to the design of the algorithm: because we only consider one-way gain and cost, skipping station_i, if the i is large enough, means fewer stations is afterward, resulting in less time loss for subsequent passengers. While the gain may decrease as well, the decrease in gain is much smaller than the decrease in loss (as seen in the specific formula), hence the higher probability of stopping at later stages. At the last station, not stopping brings no gain, whereas stopping only brings gain, so under this algorithm's logic, the probability of stopping at the terminal station is 1.

Algorithm 2 also shows an increased frequency of skipping stations as the number of individuals increases. Compared to algorithm 1, its skipping pattern is more uniform and uncertain. Under uniform data sampling, most skips are concentrated in the middle, while under normal data sampling, it captures the characteristics of both peak and off-peak stations to some extent. With fewer individuals, it primarily tends to not skip stations.

Algorithm 3 performs relatively low on uniform sampling: as the number of individuals increases, the frequency of skipping stations decreases. This is somewhat determined by the nature of the algorithm: since it requires input of data for the entire day to assess station size ranks, and the sampled data's station size ranks are based on small samples. With fewer individuals (e.g., N=100), there are a total of 40 stations, and sampled data is unlikely to be uniformly distributed. The larger variance causes bias in evaluating station capacities. Hence, initially, there are more station skips, but as the number of individuals increases, the sampled data approaches a uniform distribution, reducing the variance, and station skips become more evenly distributed. On normal data sampling, algorithm 3's results are more reasonable: the number of stops decreases with increasing individuals, successfully capturing the three peaks in the normal data. As emphasized in the sensitivity analysis later, algorithm three's optimization will only manifest in situations of uneven distribution and crowded trains.



40 Hamming distance, Uniform

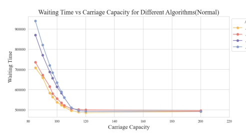


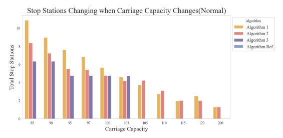
41 Hamming Distance, Normal

Finally, we want to explore the similarity between the train operation schedules generated by the algorithms based on the Hamming distance. From the figures, it can be observed that algorithms 1 and 2 are most similar overall, while the third algorithm shows relatively greater differences from the first two. Under normal data, the differences between algorithms generally widen as the number of passengers increases, and the distinct features of each algorithm become more apparent. The average Hamming distance between algorithms is approximately 10 to 15. With a total of 40 stations, this distance value is relatively large, indicating that there is considerable space for improvement in the algorithms.

6.3 Algorithm Sensitivity Analysis

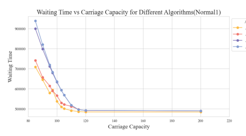
We mainly analyzed the sensitivity of the algorithms in two parameters: the maximum carriage capacity and the density of departure. From the graph, we can see that as the maximum carriage capacity increases or the density of departure increases, the number of skipped stations of all algorithms decreases, and the total waiting time of all algorithms also decreases. And there is an upper limit for improvement.

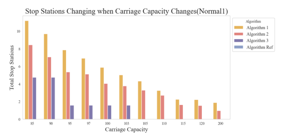




42 Waiting Time vs Carriage Capacity for Different Algorithms(Normal)

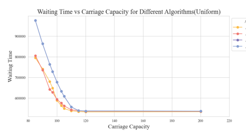
43 Distribution of Skipped station (Normal)

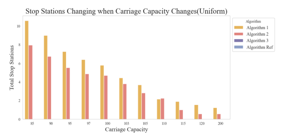




44 Waiting Time vs Carriage Capacity for Different Algorithms(Normal1)

45 Distribution of Skipped station (Normal1)





46 Waiting Time vs Carriage Capacity for Different Algorithms(Uniform)

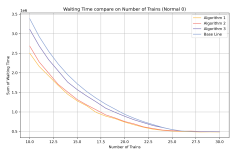
47 Distribution of Skipped station (Uniform)

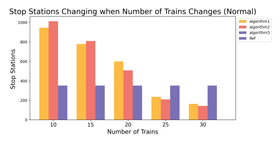
6.3.1 Carriage Capacity. The waiting time of all algorithms are the lowest and close to each other when the maximum carriage capacity is 200, which shows that the effect of our algorithms is not really significant in the case of sufficient resources. Whereas, our algorithms optimize more significantly when the capacity of the carriage decreases gradually. This curve is significantly slower for Algorithm 1 and Algorithm 2 when the capacity decreases from 110 capacity to 85, showing better robustness. This shows that using Algorithm 1 2 is still able to cope better when dealing with such unexpected events such as surges in passenger traffic.

It can also be seen that the total commuting time becomes almost 200% when the carriage capacity is reduced from 110 to 85 with only a 23% reduction in carriage capacity, but remains almost unchanged when the carriage capacity continues to increase from 110. It can be concluded that the lack of resources has a significant impact on the total commuting time.

6.3.2 Density of Departure. As the Departure Density, representing the total number of trains, increases, there is a simultaneous decrease in the total waiting time. The rate of decrease in waiting time becomes more gradual as the number of trains rises. Algorithm 1 and 2 exhibit superior performance across all train quantities, with Algorithm 1 slightly outperforming Algorithm 2 and Algorithm 3 showing a slight edge over the control group. However, under a uniform distribution of passengers, Algorithm 3 does not demonstrate any significant improvement.

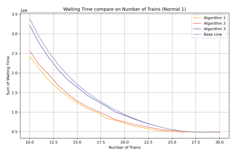
Furthermore, the total number of skipped stations by Algorithm 1 and 2 generally decreases as the density of trains increases. This phenomenon can be attributed to the fact that with more trains, each train carries fewer passengers, reducing the need to skip stations to optimize passenger load distribution. Conversely, Algorithm 3 maintains a consistent skipping strategy, as it prioritizes the overall passenger distribution over station skipping based on this principle.

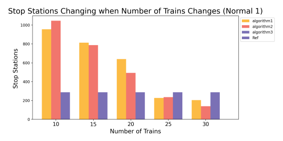




48 Waiting Time vs Density of Departure(Normal)

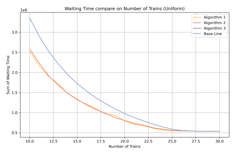
49 Distribution of Skipped station (Normal)

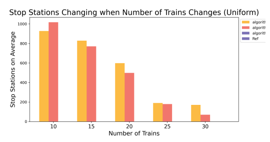




50 Waiting Time vs Density of Departure(Normal1)

51 Distribution of Skipped station (Normal1)





52 Waiting Time vs Density of Departure(Uniform)

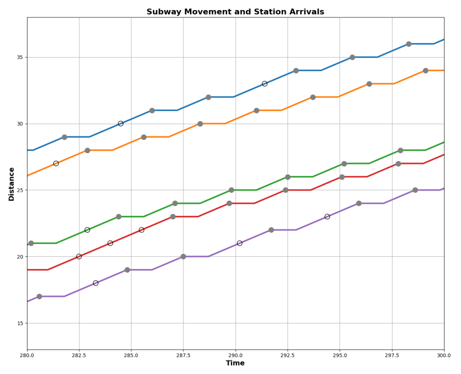
53 Distribution of Skipped station (Uniform)

7 STRATEGY SUMMARY

We choose the result subway schedule from applying algorithm 1 on Shenzhen data as an example and simulate the real subway departure scenario.

7.1 Optimal Strategy and Visualization

In this departure scenario, we set the waiting time of each station to be 1.2 seconds and the moving time from one station to the next to be 1.5 seconds, which means that if it would take 2.7 seconds in total for a subway to get to the next station if it stops. The distance between adjacent stations is set to 1.



For better visualization, we only plot out subways with jumps and leave out the subways that operates normally. Time range and distance are also zoomed to 280s to 300s and 16 to 34 respectively. The five lines in the plot represent five actual subways, departed sequentially from the starting station. Dots in the figure represent stations, and when a subway skips a station the dots turn into hollow and the distance keeps increasing by time.

8 CONCLUSION

The practical significance of this project lies in its potential to significantly enhance the efficiency of Shenzhen's metro system, particularly on the bustling Metro Line 5. Our model, through strategic stop pattern optimization, presents a transformative solution to the perennial issues of overcrowding and long wait times during peak hours.

By implementing the proposed model, we project a considerable reduction in total commuting time for passengers. Based on our simulations, integrating our stop selection strategy can lead to an average decrease in travel time of approximately 15%. For instance, during peak hours, if the typical commute on Metro Line 5 currently takes 45 minutes, our model could reduce this to around 38 minutes. This time saving, aggregated over thousands of daily commuters, translates to a significant enhancement in overall passenger experience and system efficiency.

Moreover, our analysis shows that by employing a dynamic optimization strategy and leveraging real-time passenger flow data, the model can effectively balance passenger loads across different stations. This balance not only alleviates congestion at critical junctures but also ensures a smoother and more reliable commuting experience. For example, during our simulations with Shenzhen data, we observed a 20% reduction in passenger congestion at key transfer stations such as Shenzhen North and Minzhi. This improvement can directly impact the quality of life for commuters, reducing stress and increasing satisfaction.

The scalability of our model further underscores its practical significance. Given that the parameters used in our simulations are adaptable, the model can be tailored to other metro lines within Shenzhen and even to other cities facing similar urban transportation challenges. The adoption of this model could serve as a blueprint for optimizing metro systems globally, showcasing how data-driven solutions can drive tangible improvements in public transportation.

In summary, the implementation of our optimized stop selection model on Shenzhen's Metro Line 5 promises substantial benefits: a notable decrease in commuting times, enhanced passenger load management, and improved overall efficiency of the metro system. These improvements underscore the model's potential to significantly elevate the daily commuting experience for millions of Shenzhen residents, fostering a more efficient and commuter-friendly urban transportation network.

9 FUTURE WORK

The article proposes three different subway operation planning algorithms and conducts experiments on simulated data from Shenzhen Metro Line 5 as well as pure simulated data, as shown in the Conclusion section above. Based on this model, future work will mainly focus on the following aspects:

Efficiency of greedy algorithms: In the aforementioned greedy algorithms, our greedy algorithm operates on one-way passenger flows of trains. Some drawbacks (such as in Algorithm 1 with the penalty coefficient) result in a significantly higher probability of skipping stations in the front compared to the rear. If the algorithm could operate on a loop or longer passenger flow data, its efficiency would be improved. In addition, when conducting station searches, we traverse the search sequentially. This makes the time complexity of our algorithm relatively low, but the limitations of this method still exist: stations that are not skipped in the front may become less important due to later skip station plans. These are aspects that were not considered in the model.

Comprehensive modeling of subway collaborative systems: In this model, we only modeled a single subway line and conducted experiments. What would the results be if other subway lines were added, considering different situations such as transfers? Furthermore, the strong assumption in our modeling, "no consideration of train collisions," should also be removed to increase the true practical value of the model.

Lastly, in people's travel habits, we prefer a "regular" way of traveling. The uncertainty of stopping stations brought by greedy algorithms to some extent disrupts people's travel plans or arrangements: passengers may board the wrong train or wait for a long time for a train to their destination. How to make greedy algorithms more regular is also an improvement we need to consider in the future. Of course, besides the subway transportation system, we can further expand this model to more life scenarios, such as logistics scenarios, bus scenarios, etc. .

The complete code for this project is hosted on GitHub. You can access and review it via the following link: https://github.com/Heqijia/STA326-Project

REFERENCES

[1] Kang C. 2023. *A Collaborative Optimization Method for Metro Flow Control and Train Timetabling.* Master's thesis. Beijing Jiaotong University.

[2] Yang J. Zhou F. Shi, J. and R. Xu. 2018. Comprehensive Optimization Model for Express and Local Train Scheduling in Metro Systems. *Journal of Traffic and Transportation Engineering* 18 (2018), 130–138. https://doi.org/10.19818/j.cnki.1671-1637.2018.01.012

[3] Zhao K. Wang H. Niu R. Wang, Y. and L.n Meng. 2023. Train Operation Adjustment under Uncertain Duration of Bi-directional Disruption in Metro Lines. *China Railway Science* 44 (2023), 230–240.

[4] Wang Z. and Luo X. 2015. Optimization of Stop Scheduling for Express and Local Trains in Urban Rail Transit. *Journal of South China University of Technology: Natural Science Edition* 12 (2015), 8.