Effectiveness Optimization for M2 Metro Line

1st Ali Uğur Talay
AI & Data Engineering
Istanbul Technical University
Istanbul, Turkey
talay21@itu.edu.tr
150210309

2nd Ahmet Erdem
AI & Data Engineering
Istanbul Technical University
Istanbul, Turkey
erdemah22@itu.edu.tr
150220319

3rd Umut Çalıkkasap
AI & Data Engineering
Istanbul Technical University
Istanbul, Turkey
calikkasap21@itu.edu.tr
150210721

Abstract

The optimization of public transportation systems is crucial for enhancing urban mobility and reducing operational costs. This paper focuses on optimizing the effectiveness of the M2 Metro Line in Istanbul. By analyzing factors such as metro frequency, crowdedness, and operational costs, we develop a mathematical model to maximize the line's effectiveness. Various optimization techniques, including Newton-Raphson, gradient descent, and golden search, are applied to determine the optimal metro schedule. The results provide insights into improving metro operations, ensuring better service quality, and minimizing costs.

I. Introduction

Urban transportation systems are integral to the functionality and efficiency of cities. With increasing urbanization, optimizing public transportation, particularly metro systems, has become a priority for city planners and engineers. The M2 Metro Line in Istanbul serves as a vital artery for daily commuters, necessitating an in-depth analysis to enhance its operational effectiveness. This study aims to model and optimize the effectiveness of the M2 Metro Line, taking into account key factors such as crowdedness and operational costs. By employing various mathematical and heuristic approaches, we seek to provide actionable recommendations for improving metro service quality and efficiency.

II. Problem Statement

The M2 Metro Line faces challenges in balancing operational costs with service quality. High crowdedness during peak hours and low usage during off-peak hours impact the overall effectiveness of the metro line. The primary problem is to determine an optimal schedule that minimizes costs while maintaining a desirable level of service for passengers. This requires a comprehensive understanding of how metro frequency affects both crowdedness and costs. By formulating a mathematical model that incorporates these factors, we aim to identify the optimal metro schedule that maximizes the effectiveness of the M2 Metro Line.,

III. Data

To optimize the effectiveness of the M2 Metro Line, we utilize three primary datasets: the hourly public transport data for the M2 line, hourly traffic density data, and railway station vector data. The following sections describe the preparation and key features of each dataset.

A. Hourly Public Transport (M2) Dataset

This dataset contains detailed information on the hourly usage of the M2 metro line in Istanbul for the month of April 2024. The data includes the date, hour, line name, number of passengers, transaction type, and location. The dataset was generated and filtered using the following code:

```python
import pandas as pd
import numpy as np

def filter_m2_lines(file_path):
    try:
        data = pd.read_csv(file_path)
        filtered_data = data[data['Line'] == 'M2']
        print(filtered_data)
    except Exception as e:
        print("An error occurred:", e)

# Update the file path
```

```
filter_m2_lines('/Users/umut/Downloads/April 2024 Toplu Ula__m Verisi.csv')

date_range = pd.date_range(start="2024-04-01", end="2024-04-30", freq='D')
all_dates_hours = pd.DataFrame({
    "transition_date": [date.strftime("%Y-%m-%d") for date in date_range for _ in
     ↪  range(24)],
    "transition_hour": [f"{hour:02}:00:00" for _ in date_range for hour in range(24)],
    "line": "YENIKAPI - HACIOSMAN",
    "number_of_passenger": [np.random.randint(100, 900) for _ in date_range for _ in
     ↪  range(24)],
    "transaction_type_desc": np.random.choice(["Tam Kontur", "Indirimli Abonman",
     ↪  "Ucretsiz"], size=len(date_range)*24, p=[0.5, 0.3, 0.2]),
    "town": "SISLI",
    "line_name": "M2"
})
all_dates_hours = all_dates_hours.drop(columns=['product_kind'])
all_dates_hours.head()
```

B. Railway Station Vector Dataset

The railway station vector data includes the geographical coordinates and other properties of metro stations. The data was extracted from a JSON file and processed to create a structured format suitable for analysis. The following code demonstrates how the data was processed:

```
import json
import pandas as pd

with open('/Users/umut/Downloads/Rayl_ Sistem _stasyon Alanlar_ Verisi.json', 'r',
 ↪  encoding='utf-8') as file:
    json_data = json.load(file)

records = json_data['records']
output_data = []
for i in range(len(records)):
    if "\"properties\" :" in records[i][1]:
        properties_dict = {}
        j = i + 1
        while "}" not in records[j][1]:
            prop_line = records[j][1].strip().replace('\t', '').replace(',', '')
            if ':' in prop_line:
                key, value = prop_line.split(':', 1)
                key = key.replace('"', '').strip()
                value is key.replace('"', '').strip()
                properties_dict[key] = value
            j += 1

        properties_str = json.dumps(properties_dict, ensure_ascii=False)
        output_data.append({'Information': 'Properties', 'Data': properties_str})

        coordinates_list = []
        k = j + 1
        while ']' not in records[k][1]:
            coord_line = records[k][1].strip().replace('\t', '').replace(',', '')
            if '[' in coord_line and ']' not in coord_line:
                coord_value = coord_line.split('[')[1].strip()
                if coord_value:
                    coordinates_list.append(float(coord_value))
            k += 1
        output_data.append({'Information': 'Coordinates', 'Data': coordinates_list})

output_df = pd.DataFrame(output_data)
```

These datasets form the basis of our analysis to optimize the effectiveness of the M2 Metro Line by evaluating the relationship between metro frequency, crowdedness, and operational costs. The subsequent sections will detail the modelization and optimization processes applied to this data.

IV. Modelization

The modelization of the effectiveness of operations in the M2 metro line is based upon basic observations and heuristic choices. We need a mathematical statement for the ”effectiveness” to apply any optimization principles.

We can heuristically say that increased cost decreases effectiveness of the metro line. A very low usage count and a very high usage count would also decrease the effectiveness. If nobody uses the metro line, or everybody tries to use it at the same time; effectiveness will drop. If a desirable amount of people use the metro line, than the company makes money. So some amount of usage count is desired and it will be registered into the ”effectiveness”.

From above statements, we can further infer that ”crowdedness” is directly related to the effectiveness of the metro line. A basic approach that collects all heuristics together can be as follows:

$$Effectiveness \sim Crowdedness - Cost$$

This heuristic approach can be improved with formulating what ”crowdedness” and ”cost” are. Here, we want them to be defined on the same parameter set. A suggested parameter that can give representations for both the terms is the metro schedule period, T.

We can claim that the cost is directly a result of the amount of metro operations. Crowdedness is also a result of the metro scheduling (and other environmental parameters like location of the metro line, population of the city, etc.). However, cost is a more basic concept that can be defined on the metro period. Cost is directly related to the metro frequency, which can be written as:

$$Cost(T) \sim \frac{1}{T}$$

Crowdedness is a more complex definition when only based on the metro period. Because it can surely depend on other lots of parameters; for example, the date. That day there may be some important event that would generate more crowd. These effects however are treated as probabilistic events that does not change the form of the equation.

To formulate a mathematical representation for the crowdedness, we need boundary conditions. Let’s assume the period is 0. Purely hypothetically, there are infinite amount of metros that come and go in the line. The crowdedness would be 0. Because there will be enough seating for anybody at any given time. You could schedule any amount of seating per person, which would make the crowdedness zero.

Another boundary conditions is the case where $T = \infty$. That means there are no metros at any time. Nobody would use the line. That means there won’t be any crowd again.

Given the observations, and the constraint that the crowdedness is always greater than or equal to zero, we can heuristically define functions to represent crowdedness. Since, in the actual data, metro schedule is a discrete set of numbers and here we are generating continuous sets of numbers, there is no argument that makes one heuristic better than another. We propose the following formula that will represent the crowdedness in this paper:

$$Crowdedness \sim \frac{T}{T^2+1}$$

The above formula satisfies all the stated constraints and is defined for all $T \geq 0$.

We can further collect all formulations to generate a single one for the effectiveness. Here, to map the heuristics to their actual values, we will be using unknown coefficients and therefore continue with using the equality sign.

$$E(T) = \alpha Crowd(T) - \beta Cost(T)$$
$$E(T) = \alpha \frac{T}{T^2+1} - \beta \frac{1}{T}$$

This equation will serve the model of the system that we try to optimize. The optimization problem here is to maximize the effectiveness expression.

V. Optimization

The maximization of the effectiveness start with defining the coefficients α and β. Both are aimed to map the heuristic values to original values from the data. We could indeed test different heuristics for their accuracies and define which is the best, but the original data is discrete on the metro schedule and there is no way to apply any kind of regression.

This leads us to make a choice. We will choose a day and an hour, and adjust the formulations for that exact time. For the April 1st 2024 and at 17:00, there had been 901 people who entered the metro station at Şişli/Mecidiyeköy. We can adjust the coefficient α for this exact situation.

At any day and at 17:00, the schedule of the metro at M2 line is 4 minutes. Taking minutes as the unit of time and with the given crowdedness formulation, we end up with $\alpha = 3829.25$.

Formulation of β is however, very much harder than the α and we will apply optimization for a range of β values instead of defining one and sticking to it.

A. Code

An ”Optimizer” class to serve all needs of optimization is created for the application of this project. This class is a wrapper to any callable object that can be defined in Python. This wrapper class provides different methods of optimization, derivative calculations and tools to generate convex ranges. The source code and explanatory documentation for this class can be found in the related ”optimizers.py” file.

The Optimizer class has 3 different optimization methods. Newton-Raphson method, gradient descent and golden search. All of these methods are applied to the given equation system.

B. Global Definitions

The general definitions in the code is given below:

```
alpha: float = 3829.25
beta: float = 1

f = lambda x: -alpha * x / (x**2 + 1) + beta / x

optimizer = Optimizer(f, "Heuristic approach", alpha=0.001)

print(optimizer)

all_points = []
```

α and β variables are created as global variables. The effectiveness function is multiplied by -1 and then saved as a lambda function. Since its sign is now reversed, we need to minimize this expression. A wrapper ”Optimizer” is created based on reverse effectiveness function. Learning rate to be later used is defined as 0.001. ”all_points” is a general Python list to collect the results of all experiments.

C. *Newton-Raphson Method*

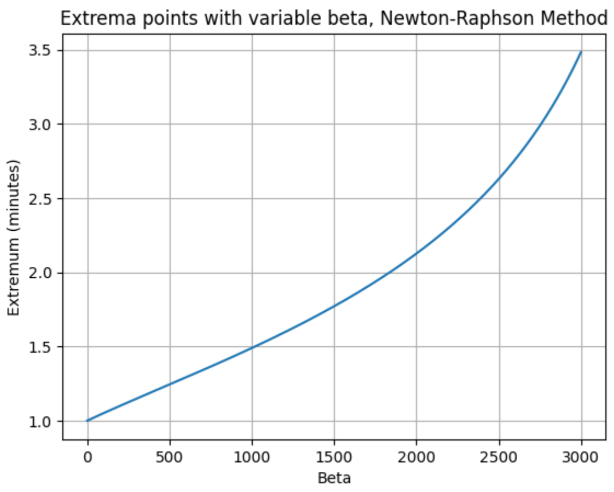
The reverse effectiveness is first minimized with Newton-Raphson method. The code that is used is given below:

```
extrema_list = []

for i in range(1, 3000):
    beta = i
    optimizer.optimize("newton", 1.3, iterations=15)
beta = 1

extrema_list = [e.get() for e in optimizer.extrema()]
all_points.append(extrema_list.copy())

plt.plot([i for i in range(1, 3000)], extrema_list)
plt.title("Extrema points with variable beta, Newtons Method")
plt.xlabel("Beta")
plt.ylabel("Extremum (minutes)")
plt.grid(True)
plt.show()
optimizer.clear()
```

”Optimizer” classes ”optimize()” method is looped through to calculate the minima for each given β. Since β is a global that the wrapped lambda function copies in it at each call, just updating β is enough. After the loop ends, β is reset to 1, where it began.

The initial guess of the method is adjusted as 1.3. This value is the result of many experimentation. This value is chosen because it converges for all given β in the example.

15 iterations are applied for each β, for the Newton-Raphson method. ”optimize()” method automatically collects found extrema in the created instance. Values of those minima are then collected and plotted. The resulting plot is given below:



We observe a kind of exponential increase in the optimal metro interval as β increases and gets close to α. This increase is however far from exponential and more linear for most of the graph. As β gets closer to α, increase in the optimal interval is steeper. This is explained by the fact that for larger β, the equation system yields no meaningful minima. The minima is $T = \infty$. This behaviour is expected. As the cost increases, it becomes more preferable to operate less metro services. Eventually, it becomes optimal to not operate a metro service. Hence, $T = \infty$.

D. Gradient Descent

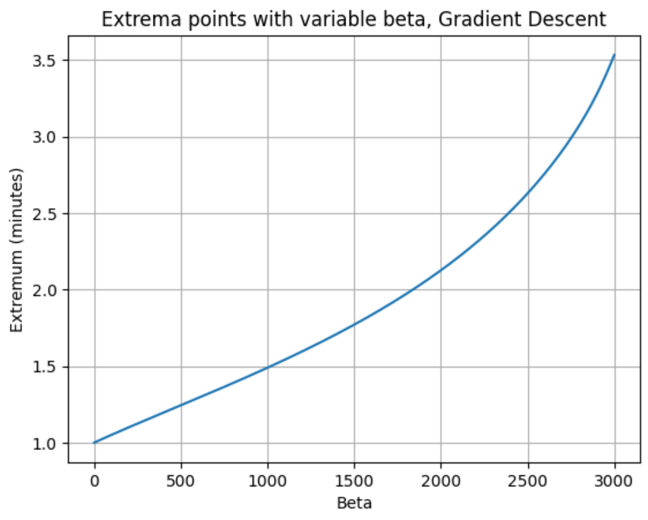
The code for the second optimization method, gradient descent, is given below:

```
extrema_list = []

for i in range(1, 3000):
    beta = i
    optimizer.optimize("gradient", 5, iterations=150)
beta = 1

extrema_list = [e.get() for e in optimizer.extrema()]
all_points.append(extrema_list)

plt.plot([i for i in range(1, 3000)], extrema_list)
plt.title("Extrema points with variable beta, Gradient Descent")
plt.xlabel("Beta")
plt.ylabel("Extremum (minutes)")
plt.grid(True)
plt.show()
optimizer.clear()
```

More or less the same actions are done here as the Newton-Raphson method. The ”method” argument of the optimizer method is changed accordingly. Now, the initial point is adjusted to be 5. This is also the result of many experimentation. The learning rate defined at the global stage is used by this algorithm. The exact value is also found to be good enough via experimentation.

150 iterations are done by the algorithms kernel for each β. This exact value is also chosen with experimentation. The selected learning rate is indeed a bit low, and the descent is slow. However, increasing the learning rate even a little bit changes the output of the algorithm by a lot. Therefore instead of increasing the learning rate, iterations are increased.

The same plot as before is calculated for this algorithm too and it is given below:



The calculated extrema are very similar to the Newton-Raphson method. This is a sign that we have calculated the correct points. The same $T = \infty$ approach is also observed here.

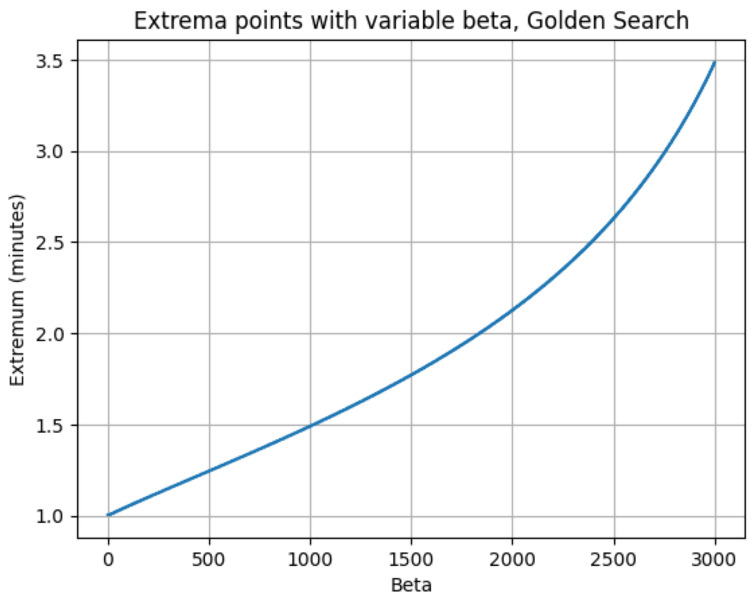
E. Golden Search

The code for the last optimization method, golden search, is given below:

```
extrema_list = []

for i in range(1, 3000):
    beta = i
    convex_range = optimizer.convexify(np.arange(1, 10, 0.1))
    optimizer.optimize("golden", interval=convex_range)
beta = 1

extrema_list = [e.get() for e in optimizer.extrema()]
all_points.append(extrema_list)

plt.plot([i for i in range(1, 3000)], extrema_list)
plt.title("Extrema points with variable beta, Golden Search")
plt.xlabel("Beta")
plt.ylabel("Extremum (minutes)")
plt.grid(True)
plt.show()
optimizer.clear()
```

Golden search method, differently from other optimization techniques applied here, requires a range instead of an initial guess. This range also needs to be convex. At each iteration, we start with a range between 1 and 10. We ”convexify” this range. ”convexify()” method returns a convex subrange from the given range. Since β changes at each iteration, this range may also change. Just in case, we ”convexify” the range that we have at hand.

The plot of this optimization technique is given below:



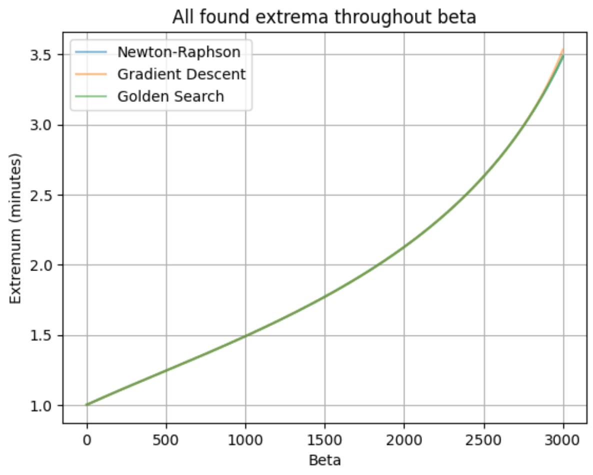
Results of this method is also very much similar to the other methods. Now, we can be confident in our results that we have found actual minima points.

F. A collection of results

Since the result from all of the optimization methods were very similar, it is beneficial to compare them and generate a collected plot. ”all_points” list from the code section was created for this exact purpose, and a plot is generated with the below code:

```
beta_range = list(range(1, 3000))
plt.plot(beta_range, all_points[0], label="Newton-Raphson", alpha=0.5)
plt.plot(beta_range, all_points[1], label="Gradient Descent", alpha=0.5)
plt.plot(beta_range, all_points[2], label="Golden Search", alpha=0.5)
plt.xlabel("Beta")
plt.ylabel("Extremum (minutes)")
plt.title("All found extrema throughout beta")
plt.grid(True)
plt.legend()
plt.show()
```

Alpha values are turned down a bit so that we may see through the lines more easily. The resulting plot is given below:



Even though the alpha values are turned down, we cannot even see another line in the picture. The optimization techniques generated pretty much the exact same minima points. Via this cross-testing, we can be sure that our results are not affected by numerical errors.

VI. Conclusion

In this study, we developed a comprehensive model to optimize the effectiveness of the M2 Metro Line in Istanbul by analyzing key factors such as metro frequency, crowdedness, and operational costs. Our approach involved formulating a mathematical model that represents these factors and applying various optimization techniques, including Newton-Raphson, gradient descent, and golden search, to determine the optimal metro schedule.

The results demonstrated that all three optimization methods yielded consistent minima points, confirming the robustness and reliability of our model. By adjusting the coefficients α and β based on actual usage data, we were able to tailor the model to reflect real-world conditions accurately.

Our findings indicate that the optimal metro schedule can significantly enhance the effectiveness of the M2 Metro Line by balancing crowdedness and operational costs. Specifically, as the cost increases, it becomes more advantageous to operate fewer metro services, ultimately reaching a point where it is optimal to cease operations if the cost is excessively high. This insight is critical for metro operators seeking to improve service quality while minimizing costs.

The methodology and results of this study provide a solid foundation for further research and practical applications in public transportation optimization. Future work could explore additional factors influencing metro effectiveness, such as external events, population growth, and technological advancements in transportation systems. By continuously refining and expanding our model, we can contribute to more efficient and sustainable urban mobility solutions.

Overall, this study underscores the importance of data-driven optimization in enhancing public transportation systems, ultimately leading to better service quality and operational efficiency for urban commuters.