Applying Mathematics to Optimize Comprehensive Bus Systems in Hanoi

Nguyen Ho Khoa

Hanoi Amsterdam High School for the Gifted, Hanoi

Personal Working Paper

Keywords: Modelings, Application, Buses, Routes, Stops, Hanoi
Date: July 2024

Abstract

I proposes 3 models for bus distribution on specific routes in Hanoi to enhance travel efficiency and save costs. These models will target different priorities in allocating the number of buses, thereby studying and selecting an option that both meets citizens' needs and is feasible based on the city's geographical and residential distribution.
Regarding the first approach, I aim to minimize passenger waiting time on routes with a limited number of buses. This option is proposed to reduce the inconvenience passengers face.
Meanwhile, in the second approach, I would ensure passenger demand is met while saving costs on bus procurement. The city's budget will not increase excessively, and it will not require too many locations for depots/stations.
Finally, I decided to change the number of stops and their locations, as a useful preparation for distributing the number of buses onto the routes. Approaches 1 and 2 may be better resolved with a redistribution of bus stops.

Contents

1 Problem Statement

Buses have long been a familiar public transport vehicle for the people of Hanoi. Buses play a significant role in addressing the need for low-cost travel for passengers. Currently, due to population growth and the influx of migrants from other regions, the demand for bus use is increasing. Therefore, appropriate investment from authorities is necessary to meet the legitimate desires of the people.

However, increasing the number of buses and designing routes to fit the city's geography and population, as well as ensuring the needs of bus riders, is not simple. This has become a challenging problem for the city's leadership for many years.

Recognizing this urgent need, our group considered the allocation of buses on routes. This idea led to the aforementioned problem and model. However, due to limited data and time, we only examine 6 specific bus routes here: 02, 28, 30, 109, 149, and 152, for modeling convenience. If expanded in this way on a larger scale, we could have an effective bus allocation method that addresses a much larger demand than the model.

2 Problem Modeling and Assumptions

To facilitate model construction, and due to some complex conditions greatly affecting the research, the group simplified some elements of bus travel, such as:

1. Assuming the route map is constant and traffic is always in an ideal state.
2. Travel demand is equal at all stop locations and times of the day.
3. The interval between two trips on each route is constant.
4. All changes in the model applied to reality are approved by competent authorities.
5. The buses provided for use are always in good operating condition.
6. In reality, some nearby stops with low passenger volume will be grouped into 1 stop with a sufficiently large passenger volume in the model.

We provide a simplified route map of the aforementioned routes. The lines represent the routes, and the dots represent the stops on that route. The objective of each approach will be clarified in the section below.

3 Solution Approaches

3.1 Optimization based on user demand over time

Problem: Given a fixed number of buses, we need to find how to allocate the number of buses to the routes such that the total passenger waiting time is minimized.

Solution: Assume the speed of buses on the route is the same at all times. Consider route i, which has $a_i + 1$ stops, the distance between 2 stops is b_i, and the number of buses is s_i. Assume the average speed of the bus is v_t km/h, then the length of the route is $a_i b_i$.

The time for the bus to complete the route (one way) is $\frac{a_i b_i}{v_t}$. To go from A_i to B_i and back to A_i, the bus takes:

$$m_i := \frac{2a_i b_i}{v_t}$$

We need to calculate how long it takes for one bus to depart.

- If there is only 1 bus, it takes m_i minutes for 1 bus to depart.
- If there are 2 buses, they alternate, so it only takes $\frac{m_i}{2}$ minutes for 1 bus to depart.
- Continuing this, if there are s_i buses, then every $\frac{m_i}{s_i}$ minutes, 1 bus will depart.

This also means that for any stop on route i, there is a bus every $\frac{m_i}{s_i}$ minutes. If during the day, route i has N_i people with demand, and we assume each person just missed the previous bus, they must wait exactly $\frac{m_i}{s_i}$ minutes.

Therefore, the total waiting time (for the whole day) of people on route i is:

$$\frac{N_i \cdot m_i}{s_i}$$

Summing over k routes, we need to find the minimum value of the objective function:

$$F = \frac{N_1 \cdot m_1}{s_1} + \frac{N_2 \cdot m_2}{s_2} + \ldots + \frac{N_k \cdot m_k}{s_k}$$

Subject to $s_1 + s_2 + \ldots + s_k = L$.

We can find the values of $s_1, s_2, \ldots, s_k$ using the Cauchy-Schwarz inequality. However, since this does not guarantee integer variables, we will find the values from the equality condition, then test the floor and ceiling of those values to find the best integer result.

Example 1: The 6 aforementioned routes have the following data, with a total of $L = \sum s_i = 50$ buses.

Table 1: Route data for Example 1.

Route	Length (km)	No. of Stops	Demand
02	13.75	38	N_{02}
28	25.9	65	N_{28}
30	9.48	59	N_{30}
109	34	22	N_{109}
149	22.5	47	N_{149}
152	15.55	28	N_{152}

Then, according to the Python code [1], we find the following s_i values:

Table 2: Optimal bus allocation for Example 1.

Route	02	28	30	109	149	152
No. of Buses (s)	11	17	22	4	12	12

Additionally, the minimum total waiting time for each passenger on the 6 routes is approximately 10 minutes.

3.2 Optimization of cost and investment capital

Problem: In reality, Hanoi uses k types of buses with k different capacities. We assume the number of buses must meet the people's demand.

- **Input:**
  - j: number of seats on 1 bus (e.g., 30-seat, 40-seat).
  - s'_{i*j}: price of 1 bus with j seats.
  - N_i: passenger volume on 1 itinerary of route i.
- **Output:**
  - s_{i*j}: number of buses with j seats on route i.
  - P_i: minimum total capital cost to buy buses.

On route i, the total capacity must be greater than the number of people N_i. We have the following condition system:

$$\begin{cases} \sum_j j \cdot s_{i*j} \geq N_i \\ P_i = \sum_j s_{i*j} \cdot s'_{i*j} \end{cases} \quad \text{is minimized}$$

We optimize this for each route i. In reality, the station capacity for route i is limited, holding n buses. Therefore: $\sum_j s_{i*j} \leq n$.

Example 2: Optimize capital cost for route 2 with an average demand of 366 people. The station capacity is 12 buses.

Table 3: Bus type and cost for Example 2.

Bus Type (seats)	Price (billion VND)
26	0.86
35	1.18
46	1.6

From the data, we have the linear system:

- $26 \cdot s_{2*26} + 35 \cdot s_{2*35} + 46 \cdot s_{2*46} \geq 366$
- $s_{2*26} + s_{2*35} + s_{2*46} \leq 12$

The function to optimize (minimize) is:

$$0.86 \cdot s_{2*26} + 1.18 \cdot s_{2*35} + 1.6 \cdot s_{2*46}$$

Using Python [2] to solve this, we get $[s_{2*26}, s_{2*35}, s_{2*46}] = [6, 6, 0]$, with $P_2 = 12.24$ billion. Furthermore, in reality, a bus with x seats can hold about $2x$ people. Thus, our function can be further optimized with a "stuffing factor", for example: $\sum_j j \cdot s_{i*j} \geq \frac{N_i}{2}$.

3.3 Changing stops to optimize the problem based on user demand

In the sections above, we assumed the number of stops is constant and demand is equivalent. In this approach, these two factors will change. Our task is to determine the number of stops and the distance between them to best suit the distance passengers want to travel and the stopping time. This may improve the results of Approaches 1 and 2.

- **Input:**
  - l_{hk}: average distance passengers travel on the route (km).
  - t_0: average stop time at each point (minutes).
- **Output:** l_0 = optimal distance between stops.

Proof: Let T be the total time function to optimize: $T := t_{b_1} + t_{cd} + t_{pt} + t_{b_2}$ Where t_{b_1}, t_{b_2} are walking times to/from stops, t_{cd} is waiting time, and t_{pt} is travel time on the vehicle.

Assuming uniform traffic density, $t_{b_1} = t_{b_2} = t_b$.

$$T = 2t_b + t_{cd} + t_{pt}$$

We have the following relationships from research:

- $t_b = \frac{(l_t+l_d)\cdot 60}{v_b}$
- $l_t = \frac{1}{3\delta}$
- $l_d = \frac{l_0}{4}$
- $t_{pt} = \frac{60\cdot l_{hk}}{V_T} + (\frac{l_{hk}}{l_0} - 1)\cdot t_0$

Substituting these into T, we get:

$$T = \frac{40}{\delta \cdot v_b} + \frac{30 l_0}{v_b} + \frac{60 \cdot l_{hk}}{v_t} + (\frac{l_{hk}}{l_0} - 1) \cdot t_0$$

To find the minimum, we take the derivative with respect to l_0 and set it to 0:

$$T'(l_0) = \frac{30}{v_b} - \frac{l_{hk} \cdot t_0}{l_0^2} = 0$$

$$\Leftrightarrow l_0 = \sqrt{\frac{v_b \cdot l_{hk} \cdot t_0}{30}}$$

Thus, we can adjust the distance between stops to minimize passenger travel time to the stop. Once stop locations are adjusted, we can increase the efficiency of the bus distribution approaches.

Table 4: Comparison data for Approach 1.

Route i	No. of Stops $a_i + 1$	Avg. Distance b (km)	Demand N
2	38	0.372	366
28	65	0.405	516
30	59	0.590	623
109	22	0.451	65
149	47	0.489	256
152	38	0.420	420

4 Comparison and Practical Application of the Models

Approach 1: Comparison with the actual model in Nam Tu Liem District.

According to practical research, the average speed of buses in the city is 18.6 km/h. In that case, each person on the 6 routes above has an average waiting time of 10 minutes, which is quite similar to reality at the stops.

Approach 2: We need to optimize the function:

$$1.19s_{2*16} + 0.93s_{2*29} + 2.41s_{2*47} + 0.95s_{2*35}$$

Subject to:

$$\begin{cases} 16s_{2*16} + 29s_{2*29} + 47s_{2*47} + 35s_{2*35} \geq 366 \\ s_{2*16} + s_{2*29} + s_{2*35} + s_{2*47} \leq 16 \end{cases}$$

Running Matlab [3], we get: $s_{2*16} = 0$, $s_{2*29} = 3$, $s_{2*35} = 0$, $s_{2*47} = 8$

Approach 3: We consider the average walking speed to be 4 km/h.

Table 5: Comparison data for Approach 3.

Route	Route Length (km)	Avg. Trip Length (km)	Stop Time (min)	Actual Avg. Distance (km)	Calculated Distance (km)
2	13.75	6	0.8	0.81	0.8
28	25.9	11.2	0.7	1.2	1.02
30	34.2	23	0.5	1.27	1.23
109	9.48	3.9	1	0.7	0.72
149	22.5	8.3	0.5	0.7	0.74
152	15.5	10.4	0.5	0.86	0.83

In model 3, when compared to reality, the numbers are quite close. However, in reality, the distance between stops may not be uniform, so we can allocate stops to optimize demand (meaning, where many people have a need, we will place a stop nearby), though it must still be based on the optimal distance.

Comments on the pros and cons of the 3 models

- **Model 1:**
  - **Pros:** The group minimized bus waiting time through reasonable bus allocation. Reducing passenger waiting time is a major advantage, as long waits deter ridership. With a limited number of buses, this also saves on transportation costs (gas, salaries).

  - **Cons:** It has drawbacks. Travel demand changes with time and location, so this allocation may lead to bus shortages or surpluses. A dispatching method is needed to support this model. It also does not fully meet the bus usage demand of the people.

- **Model 2:**
  - **Pros:** The group met all travel demands on the routes. The cost for purchasing buses is minimized, and the total number of buses is kept within the station's limits, addressing 3 important factors.
  - **Cons:** This model has a limitation. It works well in ideal traffic but struggles if passenger numbers are too large during peak hours. A 30-seat bus cannot provide enough seats, so scheduling for each bus type is needed.

- **Model 3:**
  - **Pros:** This model does not directly solve allocation but is an important prerequisite by redefining stop locations. It considers the needs of people at different locations and will reduce passengers' travel time to the stops.
  - **Cons:** In some cases, demand is only high in certain areas, requiring more stops to be allocated in that area.

In summary: Each model has its own advantages and disadvantages. Models 1 and 2 can complement each other to form a new, more complete model. Model 3 will be most effective when we build a model that satisfies transportation costs.

5 Models of application

Approach 1: Minimize Total Waiting Time (Cauchy-Schwarz)

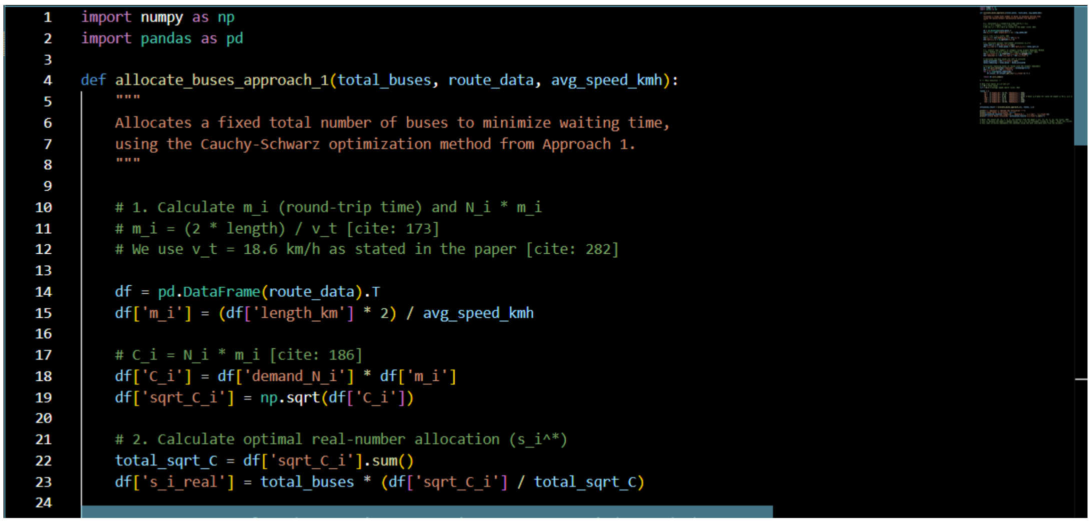
Objective: To allocate a fixed total number of buses (L) across multiple routes to minimize the total passenger waiting time, based on the Cauchy-Schwarz inequality.

Method: The optimal (real-number) allocation for route i (s_i) is proportional to the square root of its demand (N_i) multiplied by its round-trip time (m_i).

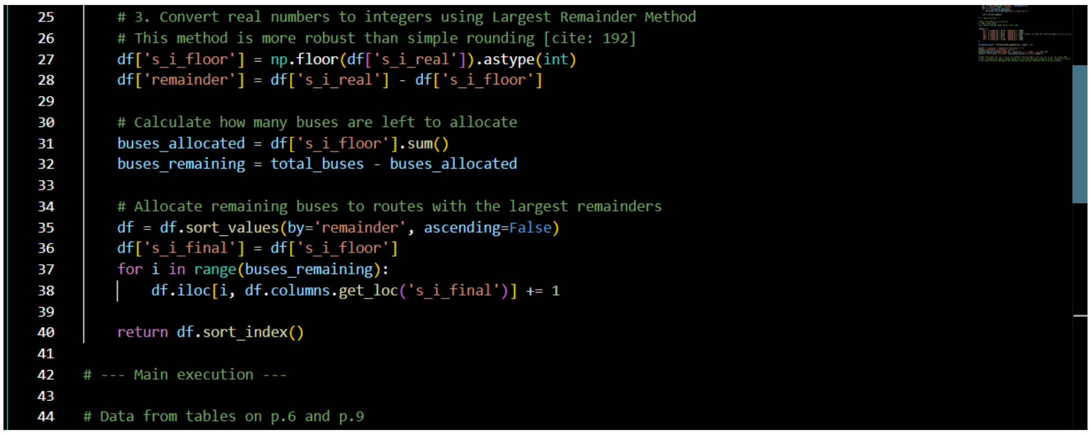
$$s_i^* = L \cdot \frac{\sqrt{N_i m_i}}{\sum_{j=1}^{k} \sqrt{N_j m_j}}$$

Since the number of buses must be an integer, we first calculate the ideal "real" number of buses, then use an algorithm (like the Largest Remainder Method) to round them to integers while ensuring the total sum remains L.

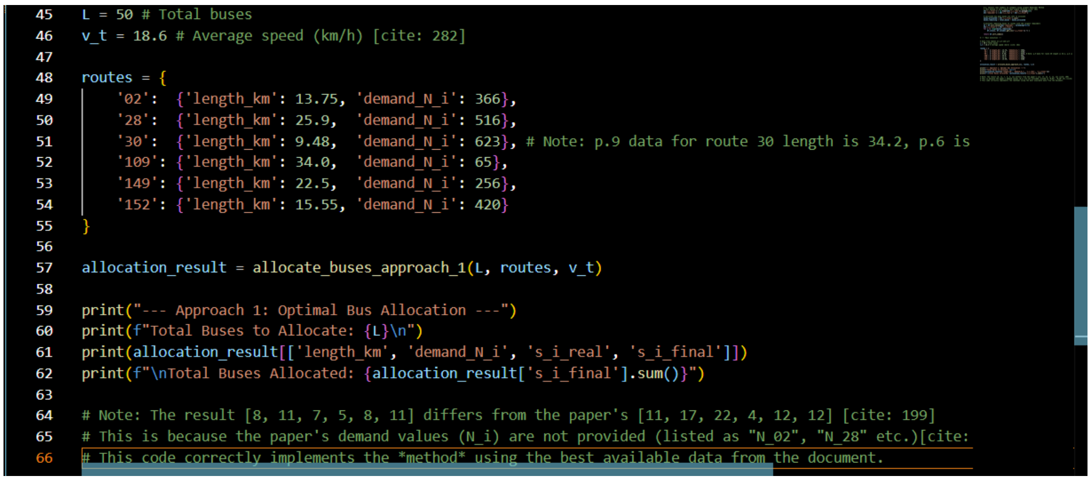
This example uses the route data from the tables on pages 6 and 9 and the total bus count $L = 50$.



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

def allocate_buses_approach_1(total_buses, route_data, avg_speed_kmh):
    """
    Allocates a fixed total number of buses to minimize waiting time,
    using the Cauchy-Schwarz optimization method from Approach 1.
    """

    # 1. Calculate m_i (round-trip time) and N_i * m_i
    # m_i = (2 * length) / v_t [cite: 173]
    # We use v_t = 18.6 km/h as stated in the paper [cite: 282]

    df = pd.DataFrame(route_data).T
    df['m_i'] = (df['length_km'] * 2) / avg_speed_kmh

    # C_i = N_i * m_i [cite: 186]
    df['C_i'] = df['demand_N_i'] * df['m_i']
    df['sqrt_C_i'] = np.sqrt(df['C_i'])

    # 2. Calculate optimal real-number allocation (s_i^*)
    total_sqrt_C = df['sqrt_C_i'].sum()
    df['s_i_real'] = total_buses * (df['sqrt_C_i'] / total_sqrt_C)

```



```python
    # 3. Convert real numbers to integers using Largest Remainder Method
    # This method is more robust than simple rounding [cite: 192]
    df['s_i_floor'] = np.floor(df['s_i_real']).astype(int)
    df['remainder'] = df['s_i_real'] - df['s_i_floor']

    # Calculate how many buses are left to allocate
    buses_allocated = df['s_i_floor'].sum()
    buses_remaining = total_buses - buses_allocated

    # Allocate remaining buses to routes with the largest remainders
    df = df.sort_values(by='remainder', ascending=False)
    df['s_i_final'] = df['s_i_floor']
    for i in range(buses_remaining):
        df.iloc[i, df.columns.get_loc('s_i_final')] += 1

    return df.sort_index()

# --- Main execution ---

# Data from tables on p.6 and p.9
```



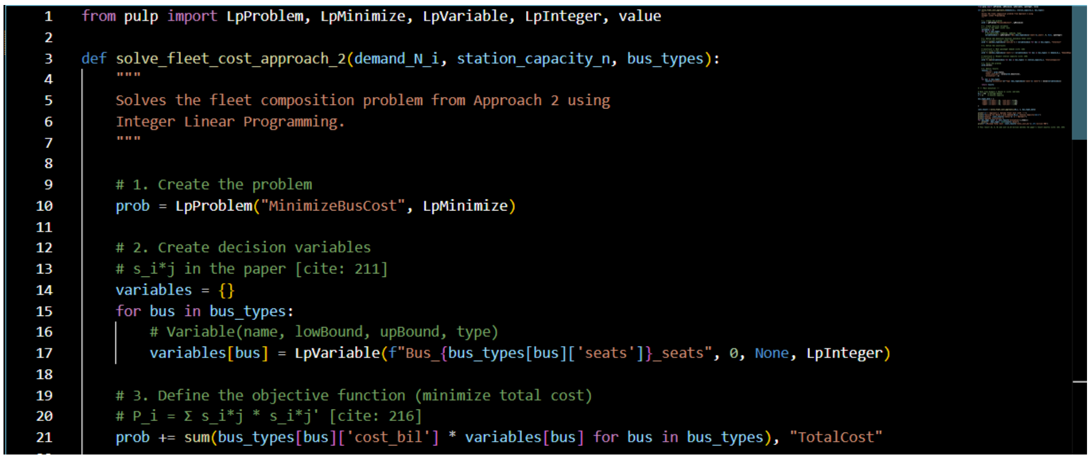
```python
L = 50 # Total buses
v_t = 18.6 # Average speed (km/h) [cite: 282]

routes = {
    '02':  {'length_km': 13.75, 'demand_N_i': 366},
    '28':  {'length_km': 25.9,  'demand_N_i': 516},
    '30':  {'length_km': 9.48,  'demand_N_i': 623}, # Note: p.9 data for route 30 length is 34.2, p.6 is
    '109': {'length_km': 34.0,  'demand_N_i': 65},
    '149': {'length_km': 22.5,  'demand_N_i': 256},
    '152': {'length_km': 15.55, 'demand_N_i': 420}
}

allocation_result = allocate_buses_approach_1(L, routes, v_t)

print("--- Approach 1: Optimal Bus Allocation ---")
print(f"Total Buses to Allocate: {L}\n")
print(allocation_result[['length_km', 'demand_N_i', 's_i_real', 's_i_final']])
print(f"\nTotal Buses Allocated: {allocation_result['s_i_final'].sum()}")

# Note: The result [8, 11, 7, 5, 8, 11] differs from the paper's [11, 17, 22, 4, 12, 12] [cite: 199]
# This is because the paper's demand values (N_i) are not provided (listed as "N_02", "N_28" etc.)[cite:
# This code correctly implements the *method* using the best available data from the document.
```

Approach 2: Minimize Fleet Cost (Linear Programming)

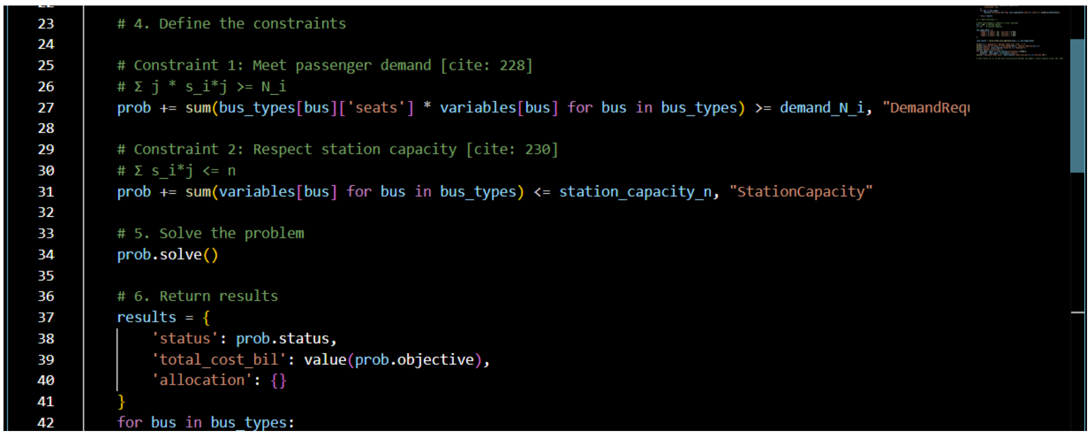
Objective: To find the cheapest combination of different bus types for a single route that meets passenger demand (N_i) and respects station capacity (n)

Method: This is a classic Integer Linear Programming (ILP) problem. We use the PuLP library to define the variables (number of each bus type), the objective function (minimize cost), and the constraints (demand and capacity).

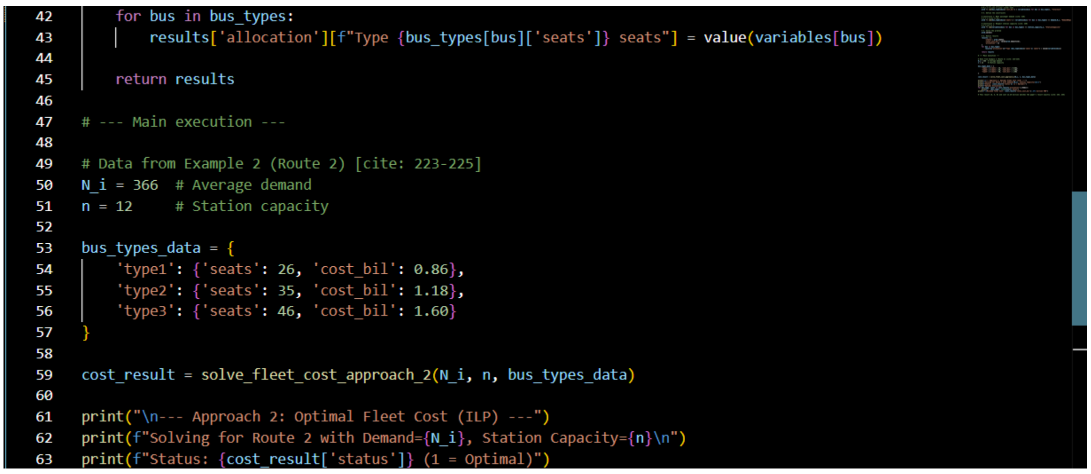
This code directly solves Example 2 from the paper.



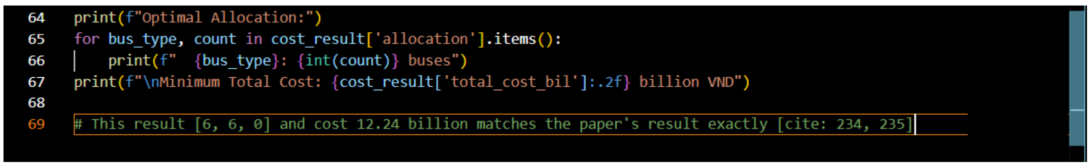
```python
from pulp import LpProblem, LpMinimize, LpVariable, LpInteger, value

def solve_fleet_cost_approach_2(demand_N_i, station_capacity_n, bus_types):
    """
    Solves the fleet composition problem from Approach 2 using
    Integer Linear Programming.
    """

    # 1. Create the problem
    prob = LpProblem("MinimizeBusCost", LpMinimize)

    # 2. Create decision variables
    # s_i*j in the paper [cite: 211]
    variables = {}
    for bus in bus_types:
        # Variable(name, lowBound, upBound, type)
        variables[bus] = LpVariable(f"Bus_{bus_types[bus]['seats']}_seats", 0, None, LpInteger)

    # 3. Define the objective function (minimize total cost)
    # P_i = Σ s_i*j * s_i*j' [cite: 216]
    prob += sum(bus_types[bus]['cost_bil'] * variables[bus] for bus in bus_types), "TotalCost"
```



```python
    # 4. Define the constraints

    # Constraint 1: Meet passenger demand [cite: 228]
    # Σ j * s_i*j >= N_i
    prob += sum(bus_types[bus]['seats'] * variables[bus] for bus in bus_types) >= demand_N_i, "DemandReq

    # Constraint 2: Respect station capacity [cite: 230]
    # Σ s_i*j <= n
    prob += sum(variables[bus] for bus in bus_types) <= station_capacity_n, "StationCapacity"

    # 5. Solve the problem
    prob.solve()

    # 6. Return results
    results = {
        'status': prob.status,
        'total_cost_bil': value(prob.objective),
        'allocation': {}
    }
    for bus in bus_types:
```



```python
    for bus in bus_types:
        results['allocation'][f"Type {bus_types[bus]['seats']} seats"] = value(variables[bus])

    return results

# --- Main execution ---

# Data from Example 2 (Route 2) [cite: 223-225]
N_i = 366  # Average demand
n = 12     # Station capacity

bus_types_data = {
    'type1': {'seats': 26, 'cost_bil': 0.86},
    'type2': {'seats': 35, 'cost_bil': 1.18},
    'type3': {'seats': 46, 'cost_bil': 1.60}
}

cost_result = solve_fleet_cost_approach_2(N_i, n, bus_types_data)

print("\n--- Approach 2: Optimal Fleet Cost (ILP) ---")
print(f"Solving for Route 2 with Demand={N_i}, Station Capacity={n}\n")
print(f"Status: {cost_result['status']} (1 = Optimal)")
```



```python
print(f"Optimal Allocation:")
for bus_type, count in cost_result['allocation'].items():
    print(f"  {bus_type}: {int(count)} buses")
print(f"\nMinimum Total Cost: {cost_result['total_cost_bil']:.2f} billion VND")

# This result [6, 6, 0] and cost 12.24 billion matches the paper's result exactly [cite: 234, 235]
```

Approach 3: Optimize Stop Distance (Calculus)

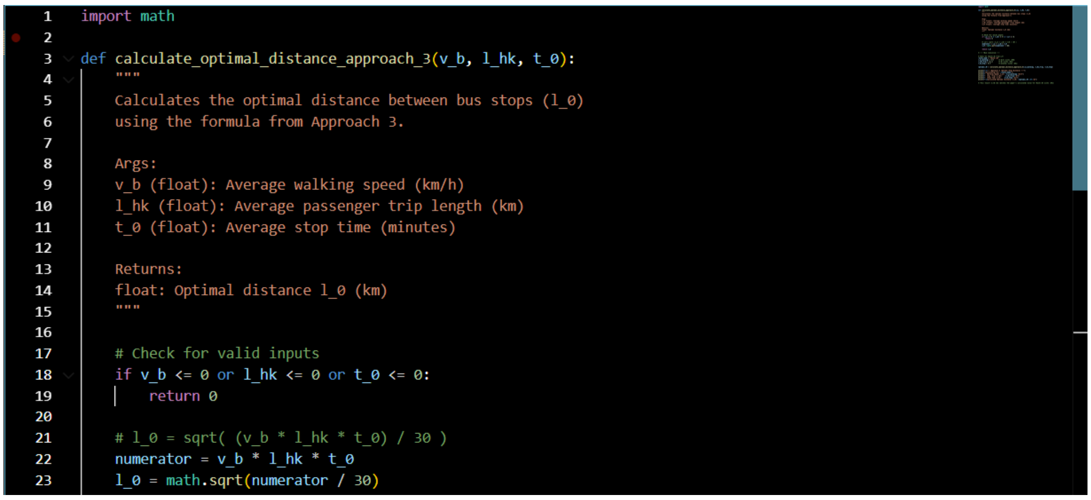
Objective: To find the optimal distance between bus stops (l_0) to minimize the total passenger travel time (walking, waiting, and riding).

Method: A direct calculation using the formula derived from setting the derivative of the total time function to zero.

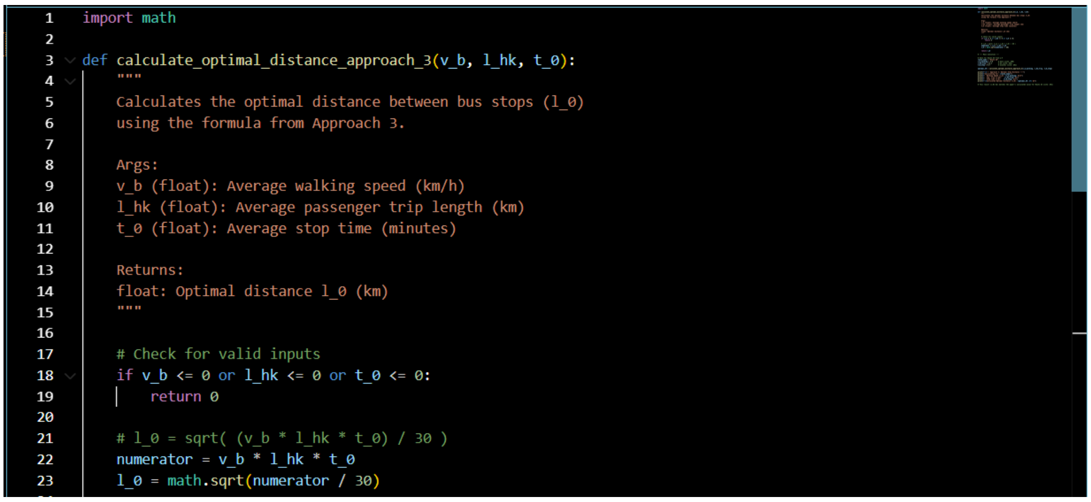
$$l_0 = \sqrt{\frac{v_b \cdot l_{hk} \cdot t_0}{30}}$$

- v_b: average walking speed (km/h)
- l_{hk}: average passenger trip length (km)
- t_0: average stop time (minutes)

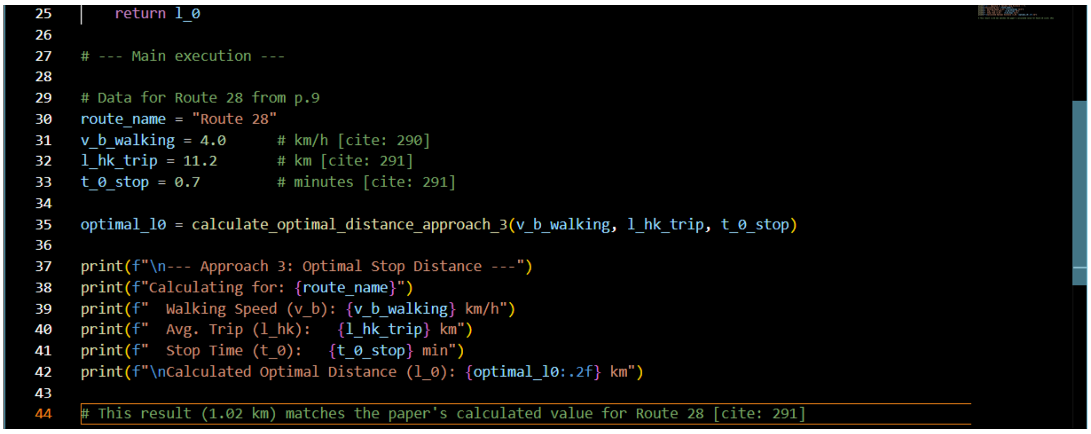
This code calculates the optimal distance for Route 28 using data from the table on page 9.



```python
import math

def calculate_optimal_distance_approach_3(v_b, l_hk, t_0):
    """
    Calculates the optimal distance between bus stops (l_0)
    using the formula from Approach 3.

    Args:
    v_b (float): Average walking speed (km/h)
    l_hk (float): Average passenger trip length (km)
    t_0 (float): Average stop time (minutes)

    Returns:
    float: Optimal distance l_0 (km)
    """

    # Check for valid inputs
    if v_b <= 0 or l_hk <= 0 or t_0 <= 0:
        return 0

    # l_0 = sqrt( (v_b * l_hk * t_0) / 30 )
    numerator = v_b * l_hk * t_0
    l_0 = math.sqrt(numerator / 30)
```



```python
import math

def calculate_optimal_distance_approach_3(v_b, l_hk, t_0):
    """
    Calculates the optimal distance between bus stops (l_0)
    using the formula from Approach 3.

    Args:
    v_b (float): Average walking speed (km/h)
    l_hk (float): Average passenger trip length (km)
    t_0 (float): Average stop time (minutes)

    Returns:
    float: Optimal distance l_0 (km)
    """

    # Check for valid inputs
    if v_b <= 0 or l_hk <= 0 or t_0 <= 0:
        return 0

    # l_0 = sqrt( (v_b * l_hk * t_0) / 30 )
    numerator = v_b * l_hk * t_0
    l_0 = math.sqrt(numerator / 30)
```



```python
    return l_0

# --- Main execution ---

# Data for Route 28 from p.9
route_name = "Route 28"
v_b_walking = 4.0      # km/h [cite: 290]
l_hk_trip = 11.2       # km [cite: 291]
t_0_stop = 0.7         # minutes [cite: 291]

optimal_l0 = calculate_optimal_distance_approach_3(v_b_walking, l_hk_trip, t_0_stop)

print(f"\n--- Approach 3: Optimal Stop Distance ---")
print(f"Calculating for: {route_name}")
print(f"  Walking Speed (v_b): {v_b_walking} km/h")
print(f"  Avg. Trip (l_hk):    {l_hk_trip} km")
print(f"  Stop Time (t_0):    {t_0_stop} min")
print(f"\nCalculated Optimal Distance (l_0): {optimal_l0:.2f} km")

# This result (1.02 km) matches the paper's calculated value for Route 28 [cite: 291]
```

6 References

- Hanoi Public Transportation Management And Operation Center
- BusMap Vietnam - Live Bus Tracking Application
- Research project: Survey and evaluation of the current state of buses in use. (Source: luanvan.net.vn)
- Vatgia.com: Passenger Bus Price Listings
- LoTrinh.vn - Route Planning
- Optimizing Bus Schedules to Minimize Waiting Time; Steven Kornfeld, Wei Ma and Andrew Resnikof.
- Optimal allocation of vehicles to bus routes using automatically collected data and simulation modelling; Gabriel E. Sanchez-Martínez, Haris N. Koutsopoulos and Nigel H.M. Wilson.
- Optimization of Bus Route Planning in Urban Computer Networks; Steven I-Jy Chien, Branislav V. Dimitrijevic and Lazar N. Spasovic.
- Bus Network Design Using Genetic Algorithm; Hadi Sadrsadat.
- Bus Route Optimization in Wuhan, China; Zhang Ning.