**Optimizing Public Bus Network Scheduling:**

**Model and Analysis of Bus Scheduling System of Anbessa City Bus Service Enterprise (ACBSE)\**

**Ameyaw, Kofi,**

**Erique, Wilson**

**Nematov, Abdulkhodiy**

**Shrestha, Kriti Bahadur**

**University of Niagara Falls, Canada**

**Master of Data Analytics**

**DAMO-610-7: Introduction to Operations Analytics**

**Professor: Marin Vratonjic**

**March 26, 2025**

Table of Contents

[1 Chapter 1 – Problem description 3](#_Toc193826906)

[1.1 Background and Context 3](#_Toc193826907)

[1.2 Comprehensive Problem Description 4](#_Toc193826908)

[1.2.1 Vehicle Scheduling Problem (VSP): 5](#_Toc193826909)

[1.3 Importance of the Service Industry 8](#_Toc193826910)

[2 Chapter 2 –Mathematical model and Coding 9](#_Toc193826911)

[2.1 Full Comprehensive Review of the Specific Problem Addressed 9](#_Toc193826912)

[2.1.1 Vehicle Scheduling Problem (VSP): 9](#_Toc193826913)

[2.1.2 Synthetic datasets generated 14](#_Toc193826914)

[2.1.3 Solver description 14](#_Toc193826915)

[2.1.4 Solver Results 16](#_Toc193826916)

[3 Chapter 3 – Extended applicability of the model 23](#_Toc193826917)

[3.1 Cost Optimization: 23](#_Toc193826918)

[3.2 Quality Service Level, for low demand routes 24](#_Toc193826919)

[3.3 Revenue and Profit Analysis: 24](#_Toc193826920)

[Appendix A.1- Python script implemented in this project. 32](#_Toc193826921)

[Appendix A.2- Python script for additional considerations. 35](#_Toc193826922)

[Appendix B - Individual Contribution & AI Usage sheet 38](#_Toc193826923)

[APPENDIX C 39](#_Toc193826924)

[References 42](#_Toc193826925)

# Chapter 1 – Problem description

Public Bus Service is a cornerstone of urban mobility worldwide, and its significance is set to grow dramatically. According to Statista, citing the OECD and International Transport Forum (ITF), global urban passenger travel demand by bus reached approximately 8 trillion passenger-kilometers in 2022 and is projected to more than double to 17 trillion passenger-kilometers by 2050 under a current ambition scenario that assumes ongoing decarbonization efforts. (Published by Statista Research Department, 2024). This overwhelming growth reflects the increasing reliance on buses to meet the transportation needs of expanding urban populations, yet it also emphasizes the pressure on public transportation systems. That is the case of Anbessa City Bus Service Enterprise (ACBSE) in Addis Ababa, Ethiopia, that struggles to optimize operations amid rising demand and limited resources.

## Background and Context

Anbessa City Bus Service Enterprise (ACBSE) is the sole public transport provider in Addis Ababa, Ethiopia, operating since 1943. The city has a population exceeding 4.8 million with a 3.8% annual growth rate and 8% urbanization. ACBSE manages a fleet of 690 operational buses, comprising 600 small buses (Type-I with 60-passenger capacity) and 90 large buses (Type-II with 90-passenger capacity), serving 110 routes and transporting approximately 640,000 passengers daily across 93 most significant routes.

Figure 1: Overview of ACBSE average Daily demand, Buses and Routes.

The enterprise employs a fixed bus scheduling system across four daily shifts:

|  |  |  |  |
| --- | --- | --- | --- |
| Shift | Time Interval | Duration (Min.) | Demand Proportion (%) |
| Morning Peak | 6:15-9:30 | 195 | 40% |
| First off-peak | 9:30-15:30 | 360 | 20% |
| Evening peak | 15:30-19:30 | 240 | 35% |
| Second off-peak | 19:30-21:00 | 90 | 5% |
| Total |  | 870 | 100% |

Table 1: Demand Proportion and average route duration in minutes

The Company struggles with demand variability which results in overcrowded buses during peak times and underutilized buses during off-peak periods, leading to:

* Inefficiencies in operational costs
* Poor bus utilization, and
* Reduced service quality.

## Comprehensive Problem Description

The article "Modeling and Analysis of Bus Scheduling Systems of Urban Public Bus Transport" by (Berhan, Mengistu, Beshah, & Kitaw, 2014) addresses the inefficiencies in the bus scheduling system of Anbessa City Bus Service Enterprise (ACBSE), the sole public transport provider in Addis Ababa, Ethiopia. The core problem is the use of a fixed bus scheduling system across 110 routes, since its inception in 1943 ACBSE has used a fixed bus scheduling model, where a constant number of buses is assigned to each route regardless of fluctuating passenger demand throughout the day. With 690 operational buses and transporting 640,000 passengers daily, the company faces the following operational problems:

### Vehicle Scheduling Problem (VSP):

With growing passenger demand, Anbessa City Bus Service Enterprise (ACBSE) operates a fleet of 690 buses across 110 routes, serving up to 640,000 passengers daily, yet struggles with a classic Vehicle Scheduling Problem (VSP). The fixed scheduling system leads to stark inefficiencies: some buses are overcrowded, straining service quality, while others run nearly empty, wasting resources and inflating operational costs. In public transport, VSP entails assigning buses to trips to match passenger demand and optimize resource use—a challenge intensified by Addis Ababa’s urban demand variability across peak and off-peak periods. This imbalance underscores the need for a demand-responsive scheduling approach to enhance user experience and cost-efficiency.

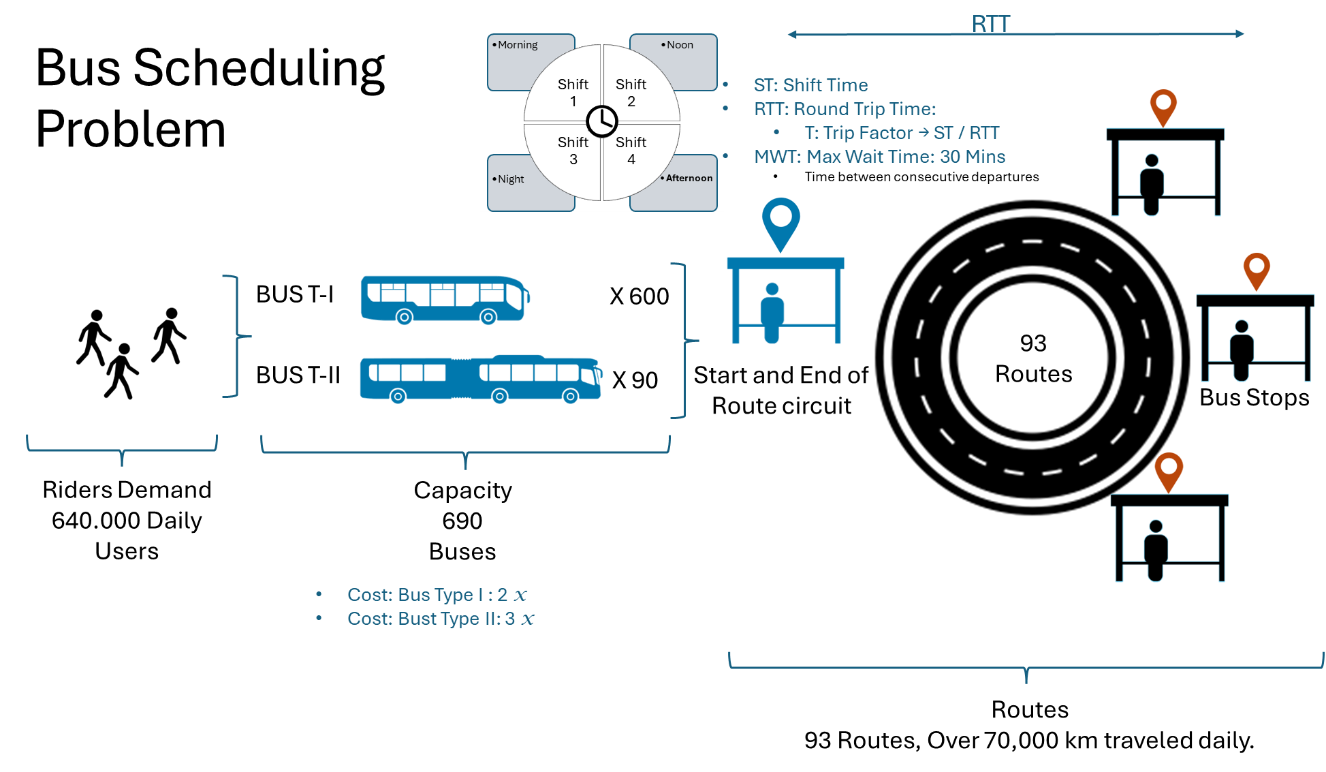


Figure 2: Vehicle Scheduling Problem (VSP)

The Problem faced by Anbessa City Bus Service Enterprise (ACBSE) involves optimizing bus assignments across 110 routes in Addis Ababa to meet fluctuating passenger demand efficiently, using two types of buses: Type-I (60-passenger capacity) and Type-II (90-passenger capacity). The enterprise's fixed scheduling system results in poor bus utilization, with some buses running empty and others overcrowded, increasing operational costs and reducing service quality. The article proposes a Linear Programming (LP) model to dynamically assign these buses based on demand across four daily shifts, aiming to minimize costs and enhance performance.

Figure 3: Problems Faced by ACBSE

The Vehicle Scheduling Problem (VSP ), also known as "truck dispatching problem," is a combinatorial optimization and logistics problem that involves determining the most efficient routes for a fleet of vehicles assigned to routes to meet the customer’s demand. The case studied proposes a Linear Programming (LP) model which will help to assign buses based on demand across four daily shifts, aiming to minimize costs and enhance performance.

The study proposes a demand-oriented Linear Programming (LP) model to optimize bus assignment across 93 selected routes, considering four daily shifts (morning peak, first off-peak, evening peak, and second off-peak). The model aims to minimize the number of trips (and thus buses) required while meeting passenger demand, improving bus utilization (targeting 60–80% range), reducing operating costs, and enhancing service quality by ensuring a bus every 30 minutes.

The following problems are not mentioned in the article, nevertheless it is important to mention that these sorts of service’s companies struggle with the following operational problems:

1. Driver Scheduling Optimization, Workforce Scheduling Problem. The article mentions related transportation scheduling problems like driver scheduling but does not address this for ACBSE.
2. Route Network Design: The article suggests that designing new routes could lead to radical performance improvements on page 7, but ACBSE’s current focus is limited to optimizing existing 93 routes and not addressing this problem in the article.
3. Maintenance Scheduling Problem: The article cites that increased distance travel impacts maintenance costs, spare part consumption, fuel consumption, and bus depreciation, however, does not address the problem.

## Importance of the Service Industry

This problem is highly relevant to the urban public transportation sector, a critical component of the service industry. Public bus transport serves economic drivers for all cities, supporting mobility and social inclusion. The inefficiencies identified in ACBSE’s scheduling system have broader implications and an optimized Scheduling model can address a suitable solution to mitigate many of the business challenges discussed:

Figure 4:Business Factors addressed by an Optimized scheduling Model

* **Economic Efficiency:** Optimizing bus schedules reduces operational costs, the study demonstrated a 13.74% cost saving, allowing reinvestment that results in benefits both the enterprise and passengers.
* **Service Quality and Customer Satisfaction:** Overcrowding and long waiting times are detriments of customer experience, while underutilized buses waste resources. In the case of study, a demand-responsive schedule model will be important to improve reliability and comfort, increasing ridership in public transport.
* **Sustainability:** Reducing 10.13% of unnecessary distance coverage reduction to 70,964 km/day, which lowers fuel consumption and emissions.
* **Scalability:** The Linear Programing model helps optimize growth planning to control costs and improve customer service, in the city of Addis Ababa, with a population exceeding 4.8 million and an 8% urbanization rate, efficient public transport is essential to manage congestion and support economic growth. The problem’s resolution can serve as a model to improve customer service and operational efficiencies.

# Chapter 2 –Mathematical model and Coding

## Full Comprehensive Review of the Specific Problem Addressed

### Vehicle Scheduling Problem (VSP):

VSP aims to minimize the number of vehicles required and reduce daily operating costs. The article’s simplified LP approach enhances practical applicability for public transport systems, addressing the trade-off between service quality and operating cost. Although the article "Modeling and Analysis of Bus Scheduling Systems of Urban Public Bus Transport" does not provide explicit data on specific costs such as the cost per kilometer run, bus startup costs, maintenance expenses, or driver wages, the proposed Linear Programming (LP) approach indirectly minimizes overall operating costs by optimizing the number of trips needed to satisfy passenger demand and resource constraints. This aligns with the fundamental objectives of the Vehicle Scheduling Problem (VSP), which aims to minimize the number of vehicles required and reduce daily operating costs, as noted in the literature.

The following Parameters and expressions allow us to describe Mathematically the problem and Model:

| Parameter | Description | Mathematical Representation |
| --- | --- | --- |
| Indices: | | |
| i | Route index, representing the 93 selected routes with historical data used in the model. | i=1,2,…,93 |
| j | Shift index, indicating the four daily operating shifts based on demand distribution.  1: Morning Peak,  2: Off-Peak 1,  3: Evening Peak,  4: Off-Peak 2 | j=1,2,3,4 |
| Parameters   * R: Number of routes (93). * S: Number of shifts (4). * C1: Capacity of Type I buses (60 passengers). * C2: Capacity of Type II buses (90 passengers). * N1: Fleet size of Type I buses (600). * N2: Fleet size of Type II buses (90). * Cost1: Cost per trip for Type I buses (200). * Cost2: Cost per trip for Type II buses (300). * ST: Shift time lengths in minutes [195, 360, 240, 90]. * M: Maximum wait time (30 minutes). * w[j]: Number of trips required per shift j, calculated as ceil(ST[j] / M). * D[i, j]: Demand (passengers) for route i in shift j. * T[i, j]: Trip factor (maximum trips a bus can make) for route i in shift j. | | |
| D i j | Average daily passenger demand for route i during shift j, derived from daily demand and shift demand proportion. | *D i j = D i× P j* |
| Ri | Routes | *Ri [R1, R2.. R93]* |
| Pi​ | Trip proportion for route i , the share of total daily demand allocated to route i. |  |
| Pj | Demand proportion for shift j, the percentage of total daily demand occurring in each shift.  40% for Shift 1,  20% for Shift 2,  35% for Shift 3,  5% for Shift 4 | *P j ​* |
| MWT | Maximum Wait time in minutes represents the max time users have to wait between consecutive trips. Assumed 30 mins. | *M* |
| T i j | Trip factor, the maximum number of trips a bus can make on route i during shift j , based on shift duration and cycle time.  For example, T1 1 =8 for Route 1, Shift 1. |  |
| W j ​ | Minimum number of trips required at shift j to ensure a maximum waiting time of 30 minutes.  Maximum waiting time 30 Minutes, defined as a constraint to guaranty the level of service. |  |
| C 1​ | Capacity of bus Type-I, the total passenger capacity (seats + standing) for small buses. | *C1=60* |
| C 2​ | Capacity of bus Type-II, the total passenger capacity (seats + standing) for large buses. | *C2=90* |
| N 1​ | Total number of available bus Type-I in the fleet. | *N1=600* |
| N 2​ | Total number of available bus Type-II in the fleet. | *N2=90* |
| Decision Variables | | |
| X i j | Number of trips made by bus Type-I  (60 - passenger capacity) for route i in shift j.  (Integer, optimized by Model) | X i j​ |
| Y i j ​ | Number of trips made by bus Type-II (90-passenger capacity) for route i in shift j.  (Integer, optimized by Model) | y i j |
| B1[i, j] | Number of Type I buses assigned to route i during shift j. | *buses\_type1[i, j]* |
| B2[i, j] | Number of Type II buses assigned to route i during shift j. | *buses\_type2[i, j]* |
| Objective Function | | |
| Minimize  Z: Total Daily trips | Total daily operating cost. Since in the article there is no detail about Costs of operation per Bus Type, per Km, the selected approach is minimizing the number of trips needed to satisfy the Demand |  |
| Alternative Minimize  Z: Total Daily Cost | Simply by using a cost factor per trip. |  |
| Constraints: | | |
| Demand Satisfaction | Demand Satisfaction:  This restriction ensures that the aggregated capacity of the available buses is sufficient to satisfy the demand on each route and shift. | ∀i,j (1) |
| Bus Reuse Constraint | Prevents overuse of buses and ensures that the number of buses assigned to a route and shift is sufficient for the required trips: | B1[i,j]×Tfactor​≥x[i,j] (2)  B2[i,j]×Tfactor≥y[i,j] (3)  where Tfactor=min(T[i,j],w[j])  ensuring buses can be reused efficiently during a shift. |
| Capacity | Fleet Availability: 600 Buses Type I and 90 Buses Type II. | (4)  (5) |
| Minimum service level | Minimum Service Frequency, or Minimum number of trips per turn, must guaranty an 30 Minutes wait for all routes, where: | x[i,j]+y[i,j]≥w[j] (7) |
| Non Negativity | **Non-Negativity:** The number of trips by both bus Type-I and Type-II must be non-negative, as negative trips are not feasible. |  |

Table 2: Mathematically Representation of Bus Scheduling Probleml

### Synthetic datasets generated

The input data for the Vehicle Schedule Optimization model is at least the demand, the capacity of vehicles, the time and distance of routes. Since the Article has incomplete demand data and a Trip factor that consolidates the information of distance of route and duration of shift in a single parameter, we have used the available data, the proportionality described in the article and randomly generated data to complete the missing values, please refer to Appendix C in this document to review the synthetic dataset generated, also a demand.csv file has been attached along with this report.

### Solver description

The solver used to find an optimized solution of the vehicle scheduling problem with the described features is SCIP (Solving Constraint Integer Programs), which is an open-source solver framework for solving mixed-integer programming (MIP) problems. It’s integrated into Google’s OR-Tools library (pywraplp), which provides a Python interface to define and solve optimization problems. SCIP combines techniques from linear programming (LP), integer programming (IP), and constraint programming (CP) to find optimal solutions. From the solvers presented in the figure our code and model is using simple SAT for constraints and MIP Mixed Integer Program (Linear Programming). Briefly the flow of the SCIP solver is described as follows:

* **Pre-solving:** Simplifies the problem by removing redundant constraints and tightening bounds.
* **Linear Relaxation:** Solves the problem as a continuous LP (ignoring integer constraints) to get a lower bound.
* **Branching:** If variables aren’t integer, SCIP splits the problem into subproblems (x[i,j] <= 2 or x[i,j] >= 3) and explores these branches.
* **Cutting Planes:** Adds constraints to eliminate infeasible non-integer solutions while preserving the optimal integer solution.
* **Heuristics:** Uses strategies (rounding, diving) to quickly find good integer solutions.
* **Backtracking:** If an iteration doesn’t lead to a better solution, SCIP prunes it and moves to another iteration.

A diagram of a diagram

AI-generated content may be incorrect.

Figure 5: Solvers studied, taken from SCIP Workshop 2018. (Hendel, 2018)

### Solver Results

Our Model with the synthetic demand Data almost reaches the following optimized KPIs from the document:

|  |  |  |  |
| --- | --- | --- | --- |
| KPI | Existing System | Improved System | Improvement (%) |
| Total Daily Operating Cost (ETB) | 1,101,283.68 | 949,991.49 | 13.74% savings |
| Total Savings in Operating Cost (ETB) | - | 151,292.19 | 13.74% |
| Savings in Total Distance Covered | 78,963.7 Km | 70,964 Km | 10.13% |
| Additional Available Trips per Day | 5,504 trips | 4,630 trips | 15.88% |
| Bus Utilization - Morning Peak | 116.1% | 89.8% | Reduced congestion |
| Bus Utilization - First Off-Peak | - | 51.19% | 19.75% increase |
| Bus Utilization - Evening Peak | - | 82.24% | - |
| Bus Utilization - Second Off-Peak | - | 42.1% | 12.15% increase |

Table 3: optimized KPIs from the document (Berhan, Mengistu, Beshah, & Kitaw, 2014).

To compare the Cost, we have considered the following calculations:

1. Average Trip: 15 Km
2. Maintenance Cost Bust Type 1: 200 ETB per Trip
3. Maintenance Costo Bust Type II: 300 ETB Per Trip
4. Daily Demand: 300,000 Riders.

|  |  |  |  |
| --- | --- | --- | --- |
| KPI | Existing System | Improved System | Improvement (%) |
| Total Daily Operating Cost (ETB) | $1,101,283.68 | $1,020,700.00 | 7.32% savings |
| Total Savings in Operating Cost (ETB) | - | $80,583.68 |  |
| Savings in Total Distance Covered | 78,963.7 Km | 72,037 Km | -8.77% |
| Additional Available Trips per Day | 5,504 trips | 4,700 trips | 14.61% |

Table 4: optimized KPIs from our solver for similar capacity and 60 Minutes of max wait time for routes/shif of low demand.

The little difference is due to the demand and synthetic data generated from the seed in the case study document, to maintain the proportionality of demand distribution for missing values in the following table of the document.

A table with numbers and lines

AI-generated content may be incorrect.

Table 5: Demand and Trip Factor dataset for the LP Model Input. (Berhan, Mengistu, Beshah, & Kitaw, 2014)

The model applied to the provided demand and Trip Factor dataset, returned an optimized solution for the problem described in the article where the main objective was to minimize the number of trips and therefore reach to lowest cost operational schedule.



Table 6Optimized Results for every Route and Shift from the Solver.

The following line chart in orange line represents the daily aggregated demand and number of trips for 93 routes, also the horizontal blue line shows the maximum capacity of 560.000 riders daily provided in the article as max capacity but distributed evenly for all routes (the area under the line is 560.000 users). In Figure 6: Bus Trips No Optimization vs Demand (Max Wait Time 60 Mins), also the green line represents the used capacity when restrictions are relaxed, to simulate a pre optimize scenario, the blue line represents the actual demand from the generated dataset. The used capacity (green) often exceeds demand (orange) dramatically, indicating over-allocation of bus trips in the pre-optimization scenario.

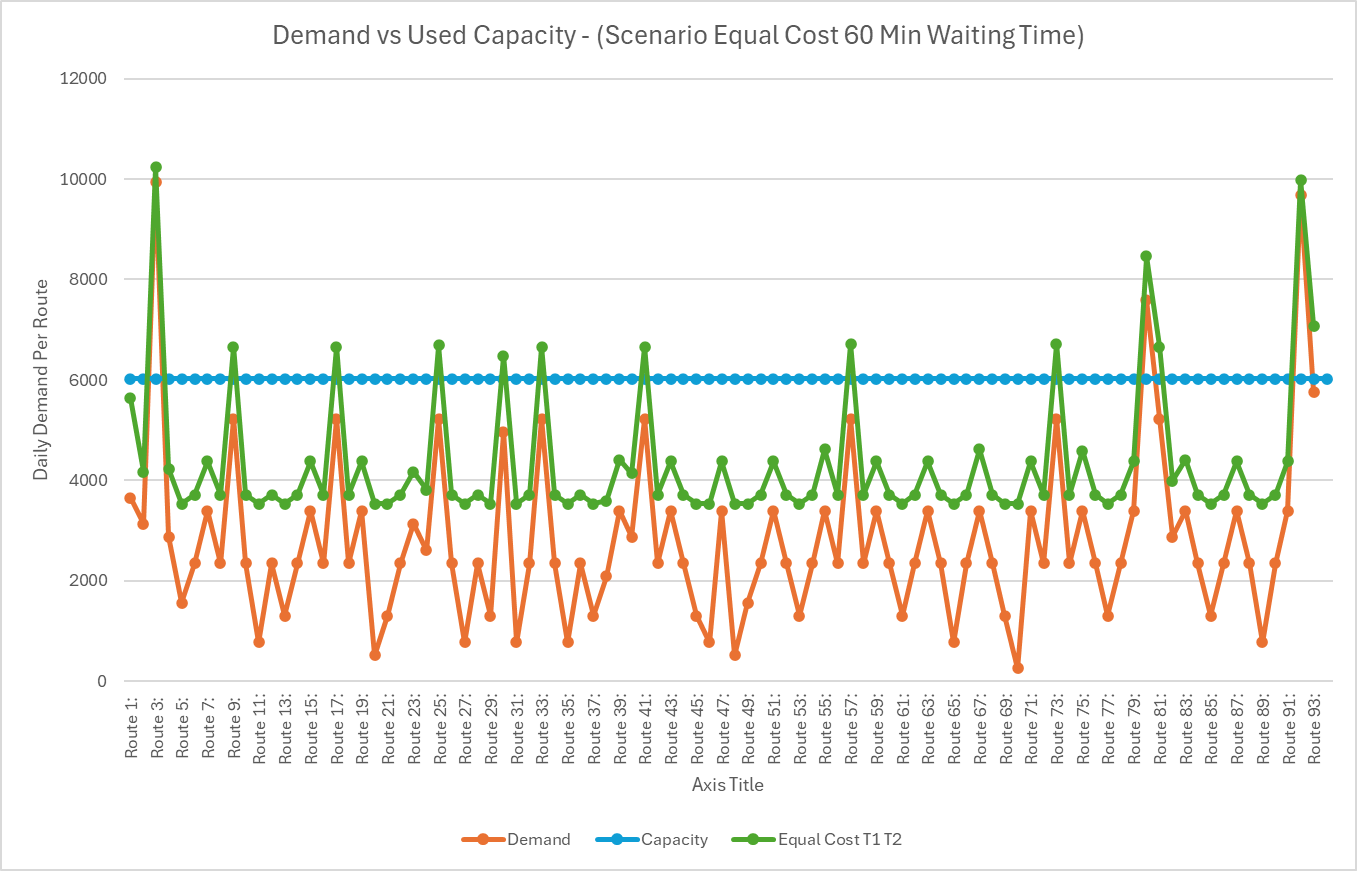


Figure 6: Bus Trips No Optimization vs Demand (Max Wait Time 60 Mins)

In Figure 7: Bus Optimized Used Capacity vs Demand (Max Wait Time 30 Mins), we have applied the solver with 60 Min wait constrain and applied prioritization based on lower operations cost in objective Function, Post-optimization, the used capacity (green) is much better aligned with demand (orange) compared to Figure 8. Used capacity rarely exceeds the total capacity line, indicating that the optimization respects the fleet constraints (N1 + N2 buses) and avoids over-allocation. There are still minor discrepancies where used capacity slightly exceeds demand, due to the minimum service level constraint (x[i,j] + y[i,j] >= w[j]), which ensures a minimum number of trips per shift, a bus every 60 minutes at most.



Figure 7: Bus Optimized Used Capacity vs Demand (Max Wait Time 30 Mins)

When we activated the service level constraint the least demand routes evidently have extra available unused capacity, The 15-minute wait time scenario for routes that have lower demand and thus more unused capacity overall, highlighting the challenge of balancing service levels with operational efficiency. To optimize, the service could reallocate resources from low-demand routes or adjust capacity dynamically based on demand patterns.



Figure 8: Bus Trips and Cost Optimization vs Demand (Max Wait Time 15 Mins)



Figure 9: Bus Trips and Cost Optimization vs Demand (Max Wait Time 120 Mins)

# Chapter 3 – Extended applicability of the model

This Model was further optimized to approach the following realistic situations that are not addressed by the article. The optimizations to the model were:

* Cost Considerations.
* Quality Service Level, for low demand routes.
* Revenue and Profit Analysis

## Cost Optimization:

For the objective function optimization, we included the variable of cost for optimization, since the Article approach is to minimize the number of trips and indirectly optimize operations cost, a more realistic situation s to minimize the operation cost since buses Type II have more expensive operations cost when compared with buses Type II. The model improved by suggesting to use cheap Buses first and expensive ones when the demand forces to, to do this we have considered the following cost considerations:

* 1. Maintenance Cost per Bus Type 1: 200 ETB per Trip
     + 10.000 ETB 5.000 Km, with an average trip of 100K, 200 ETB per Trip.
  2. Maintenance Cost per Bus Type 1: 300 ETB per Trip
     + 10.000 ETB 5.000 Km, with an average trip of 100K, 200 ETB per Trip.

|  |
| --- |
| Cost1=200 # Weight to priorize by Bus Type-I (Updated)  Cost2=300 # Weight to priorize by Bus Type-II (Updated)  # --- Objective Function ---  objective = solver.Sum(Cost1\*x[i, j] + Cost2\*y[i, j]  for i in range(R) for j in range(S)) # Minimize Cost of Trips in a day  solver.Minimize(objective) |

Table 7: Code section to include Operation Cost in the Model

## Quality Service Level, for low demand routes

The Model tries to optimize the cost based only on demand and capacity restrictions, the routes and shift with low demand are assigned too few buses and trips, to avoid service degradation, we have included a constrain that considers a minimum frequency of bus departures on all routes whether there is low or high demand, the algorithm consider first the demand constrain and the maximum wait time or trips frequency. The code is presented in the following capture:

|  |
| --- |
| ST=[195,360,240,90] # Shifts Time Length in Minutes [Shift 1, Shift 2, Shift 3, Shift 4]  M=30 # MWT: Max Wait Time: 30 Mins  w = [math.ceil(length / M) for length in ST] # Calculate trips per shift (rounding up)  # 6. Customer Experience: Minimum Service Level for low demand routes      for i in range(R):          for j in range(S):              solver.Add(x[i, j] + y[i, j] >= w[j]) |

## Revenue and Profit Analysis:

Revenue and Profit Analysis: The model seeks to optimize costs based solely on demand, aiming to minimize expenses while maintaining the highest possible service level, even for less busy routes or shifts. However, it consistently faces the challenge of insufficient revenue to sustain certain routes. This highlights the need to analyze the minimum demand required to support a financially viable transport service. For this analysis, we have utilized the revenue variable, which is inherently derived from the fares paid by riders using the service. We have included the fare variable to calculate the daily revenue and compared with the cost allowed us to understand the minimum demand levels to maintain the best possible service level, this is useful for weekends, and nights for companies to evaluate the adequate mix of fares, service level and expected demand to maintain a financially viable transport service. The following code was added in the optimization algorithm. We also have included a Dialog Box to ask for expected Daily demand (total number of riders expected), this number is then distributed proportionally to routes and shifts using the factors (Pi) and in the article:

|  |  |
| --- | --- |
|  |  |

Table 8: Proportions given in the artile to distribute the Daily demand among Routs and Shifts. (Berhan, Mengistu, Beshah, & Kitaw, 2014)

A screenshot of a computer error

AI-generated content may be incorrect.

Figure 10: Message Box to ask for Daily Demand to Software User.

|  |
| --- |
| F=5 #Fare if each trip per passenger, (equivalent to 50 cents of Dollar).  #Ask Demand D  root = tk.Tk()  root.withdraw()  DD = simpledialog.askinteger("Input Demand", "Enter daily demand:", minvalue=0)  if DD is None:      messagebox.showerror("Error", "Demand input canceled. Exiting.")      DD = 0  # Default Value  root.destroy()  D = pd.DataFrame([[round(DD \* pr \* SP[j]) for j in range(S)] for pr in Pr.iloc[:, 0]], columns=[f"D{j+1}" for j in range(S)])  # Calculate D[i,j] as ceiling of DD \* Pr[i] \* SP[j].  # --- Constraints ---  # 1. Demand Satisfaction  for i in range(R):     for j in range(S):         solver.Add(C1\*x[i, j] + C2\*y[i, j] >= D.iloc[i,j])   # 3. Capacity: Fleet Availability for Bus Type-I + Type-II  # Create summary table  summary\_data = {          "Metric": ["Total Trips T1", "Total Trips T2", "Total Buses T1", "Total Buses T2",                     "Total Trips", "Total Buses","Frequency (Mins)", "Total Cost","Daily Riders Demand","Fare","Revenue"],          "Value": [total\_trips\_t1, total\_trips\_t2, total\_buses\_t1, total\_buses\_t2,                    total\_trips, total\_buses,M, total\_cost,DD,F,F\*DD]      }      summary\_df = pd.DataFrame(summary\_data)    # Save and display summary      print("\nSummary Table:")      print(summary\_df) |

Table 9: Code for Profit Analysis

After calling the solver using different possible scenarios of Demand and Service Level, the solver returned the following results for number of trips, buses, costs, revenue and profit.

| Scenario | Demand (Riders) | Trips T1 | Trips T2 | Total Trips | Buses T1 | Buses T2 | Total Buses | Frequency (Mins) | Total Cost | Fare | Revenue | Profit |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 100000 | 5437 | 67 | 5504 | 1092 | 76 | 1168 | 15 | $ 1,107,500.00 | 5 | $ 500,000.00 | $ (607,500.00) |
| 2 | 200000 | 5266 | 475 | 5741 | 1177 | 205 | 1382 | 15 | $ 1,195,700.00 | 5 | $ 1,000,000.00 | $ (195,700.00) |
| 3 | 300000 | 6662 | 162 | 6824 | 1326 | 163 | 1489 | 15 | $ 1,381,000.00 | 5 | $ 1,500,000.00 | $ 119,000.00 |
| 4 | 400000 | 7294 | 566 | 7860 | 1525 | 265 | 1790 | 15 | $ 1,628,600.00 | 5 | $ 2,000,000.00 | $ 371,400.00 |
| 5 | 500000 | 7711 | 1165 | 8876 | 1677 | 246 | 1923 | 15 | $ 1,891,700.00 | 5 | $ 2,500,000.00 | $ 608,300.00 |
| 6 | 550000 | 8586 | 1028 | 9614 | 1714 | 258 | 1972 | 15 | $ 2,025,600.00 | 5 | $ 2,750,000.00 | $ 724,400.00 |
| 7 | 100000 | 2774 | 168 | 2942 | 685 | 158 | 843 | 30 | $ 605,200.00 | 5 | $ 500,000.00 | $ (105,200.00) |
| 8 | 200000 | 3188 | 601 | 3789 | 868 | 302 | 1170 | 30 | $ 817,900.00 | 5 | $ 1,000,000.00 | $ 182,100.00 |
| 9 | 300000 | 5122 | 205 | 5327 | 1110 | 76 | 1186 | 30 | $ 1,085,900.00 | 5 | $ 1,500,000.00 | $ 414,100.00 |
| 10 | 400000 | 5672 | 831 | 6503 | 1336 | 270 | 1606 | 30 | $ 1,383,700.00 | 5 | $ 2,000,000.00 | $ 616,300.00 |
| 11 | 500000 | 7076 | 982 | 8058 | 1553 | 290 | 1843 | 30 | $ 1,709,800.00 | 5 | $ 2,500,000.00 | $ 790,200.00 |
| 12 | 550000 | 7569 | 1199 | 8768 | 1581 | 356 | 1937 | 30 | $ 1,873,500.00 | 5 | $ 2,750,000.00 | $ 876,500.00 |
| 13 | 555000 | 7425 | 1383 | 8808 | 1692 | 340 | 2032 | 30 | $ 1,899,900.00 | 5 | $ 2,775,000.00 | $ 875,100.00 |
| 14 | 100000 | 1245 | 584 | 1829 | 510 | 226 | 736 | 60 | $ 424,200.00 | 5 | $ 500,000.00 | $ 75,800.00 |
| 15 | 200000 | 2456 | 703 | 3159 | 755 | 270 | 1025 | 60 | $ 702,100.00 | 5 | $ 1,000,000.00 | $ 297,900.00 |
| 16 | 300000 | 3893 | 807 | 4700 | 991 | 211 | 1202 | 60 | $ 1,020,700.00 | 5 | $ 1,500,000.00 | $ 479,300.00 |
| 17 | 400000 | 4515 | 1493 | 6008 | 1241 | 329 | 1570 | 60 | $ 1,350,900.00 | 5 | $ 2,000,000.00 | $ 649,100.00 |
| 18 | 500000 | 6178 | 1497 | 7675 | 1486 | 358 | 1844 | 60 | $ 1,684,700.00 | 5 | $ 2,500,000.00 | $ 815,300.00 |
| 19 | 550000 | 7631 | 1083 | 8714 | 1592 | 333 | 1925 | 60 | $ 1,851,100.00 | 5 | $ 2,750,000.00 | $ 898,900.00 |
| 20 | 555000 | 7269 | 1412 | 8681 | 1640 | 346 | 1986 | 60 | $ 1,877,400.00 | 5 | $ 2,775,000.00 | $ 897,600.00 |
| 21 | 560000 | 6896 | 1700 | 8596 | 1676 | 360 | 2036 | 60 | $ 1,889,200.00 | 5 | $ 2,800,000.00 | $ 910,800.00 |
| 22 | 25000 | 711 | 61 | 772 | 375 | 156 | 531 | 120 | $ 160,500.00 | 5 | $ 125,000.00 | $ (35,500.00) |
| 23 | 50000 | 639 | 291 | 930 | 400 | 193 | 593 | 120 | $ 215,100.00 | 5 | $ 250,000.00 | $ 34,900.00 |
| 24 | 100000 | 1214 | 382 | 1596 | 495 | 218 | 713 | 120 | $ 357,400.00 | 5 | $ 500,000.00 | $ 142,600.00 |
| 25 | 200000 | 2174 | 817 | 2991 | 749 | 328 | 1077 | 120 | $ 679,900.00 | 5 | $ 1,000,000.00 | $ 320,100.00 |
| 26 | 300000 | 4132 | 607 | 4739 | 1026 | 290 | 1316 | 120 | $ 1,008,500.00 | 5 | $ 1,500,000.00 | $ 491,500.00 |
| 27 | 400000 | 4641 | 1385 | 6026 | 1301 | 317 | 1618 | 120 | $ 1,343,700.00 | 5 | $ 2,000,000.00 | $ 656,300.00 |
| 28 | 500000 | 6323 | 1385 | 7708 | 1548 | 360 | 1908 | 120 | $ 1,680,100.00 | 5 | $ 2,500,000.00 | $ 819,900.00 |
| 29 | 550000 | 7434 | 1203 | 8637 | 1647 | 360 | 2007 | 120 | $ 1,847,700.00 | 5 | $ 2,750,000.00 | $ 902,300.00 |
| 30 | 555000 | 7225 | 1432 | 8657 | 1163 | 347 | 1510 | 120 | $ 1,874,600.00 | 5 | $ 2,775,000.00 | $ 900,400.00 |
| 31 | 560000 | 7290 | 1428 | 8718 | 1639 | 360 | 1999 | 120 | $ 1,886,400.00 | 5 | $ 2,800,000.00 | $ 913,600.00 |
| 32 | 562000 | 7256 | 1463 | 8719 | 1668 | 360 | 2028 | 120 | $ 1,890,100.00 | 5 | $ 2,810,000.00 | $ 919,900.00 |

Table 10: solver result for different demand scenarios

The following multi line chart plots profit (in millions) against daily demand for four wait time scenarios: 15 minutes (blue), 30 minutes (orange), 60 minutes (green), and 120 minutes (red). Profit increases with demand across all scenarios, but the 15-minute wait time consistently yields the highest profit, peaking at $2,744,000 at 550,000 daily demands. The 120-minute wait time yields the lowest profit, with a maximum of $919,900 at 562,000 demand. The table below the graph provides detailed data, showing the number of T1 and T2 trips, daily demand, and profit for each scenario. For example, at a 15-minute wait time with 300,000 demands, profit is $371,400, while at a 120-minute wait time with the same demand, profit drops to $491,500.



Figure 11: Profit Analysis for Service Level by Wait Time (Bus Frequency, 15', 30', 60', 120' )

**Operational Implications:** The data aligns with previous analyses, where T1 units (smaller, cheaper buses) scale with demand, while T2 units remain stable. Shorter waiting times strategy requires more T1 buses, driving up costs but also enabling higher revenue through better service levels. To maximize profit, the service should prioritize shorter waiting times in high-demand areas, using a flexible fleet of T1 buses to meet demand spikes. In low-demand areas, longer waiting times can be used to minimize losses (60 minutes or longer), relying on a smaller, stable fleet of T2 buses.

Shorter waiting times (higher bus frequency) not only improve experience but also attract more riders, increasing revenue significantly while costs grow at a slower rate, leading to higher profits. This suggests that investing in frequent service can be more profitable, especially at higher demand levels.

|  |  |
| --- | --- |
|  |  |
|  |  |

Table 11: Profit Analysis for studied scenarios.

Longer waiting times have lower operating costs, allowing the service to break even at lower demand levels. However, the profit potential is much higher with shorter waiting times as demand increases, indicating a trade-off between cost efficiency and revenue generation. The service benefits from economies of scale where higher demand leads to better utilization of buses, increasing revenue without a proportional rise in costs.

The fleet can meet a peak daily demand of around 560.000 riders daily, the 120-minute waiting time scenario relaxes the demand in less busy routes, increasing waiting times to save on costs becomes counterproductive as it stifles demand and revenue growth.

**Strategic Implications:** At very low demand, all scenarios show negative or minimal profit. The service needs a minimum demand threshold (around 50,000) to cover costs and achieve profitability. Routes with consistently low demand may need to be restructured or subsidized to remain operational.

The following line charts show the number of trips and buses for different scenarios and demand evolution, this can be useful to understand the operational approach to low and high demand routes, shifts or seasons.

|  |  |
| --- | --- |
|  |  |
|  |  |

Table 12: # Trips Analysis for Daily Demand with 15', 30’, 60’ and 120’ Mins Wait Time

Table 9: # Trips Analysis for Daily Demand with 15', 30’, 60’ and 120’ Mins Wait Time illustrates the number of daily trips (T1 and T2) required to meet varying levels of daily demand while maintaining bus frequencies with wait times of 15, 30, 60, and 120 minutes. For all waiting times, T1 trips are more sensitive to demand and wait time adjustments since they have less cost and more available units, while T2 trips are more stable, indicating the operational constraints.

|  |  |
| --- | --- |
|  |  |
|  |  |

Table 13: # Number of Buses to Meetr Daily Demand with 15', 30’, 60’ and 120’ Mins Wait Time scenarios

T1 buses are more responsive to changes in demand and wait time, due to their availability and lower cost, while T2 buses reflect operational cost constraints, maintaining a consistent fleet size across scenarios.

# Appendix A.1- Python script implemented in this project.

|  |
| --- |
| from ortools.linear\_solver import pywraplp  import math  import pandas as pd  def optimize\_bus\_routes(D,R,S,C1,C2,N1,N2,Cost1,Cost2, ST, M):      w = [math.ceil(length / M) for length in ST] # Calculate trips per shift (rounding up)      print(w)      # Create the solver SCIP      solver = pywraplp.Solver.CreateSolver('SCIP')      if not solver:          print("Impossible to create the Solver.")          exit()      # --- Variables ---      x = {}      y = {}      buses\_type1 = {}  # Number of buses Type I      buses\_type2 = {}  # Number of buses Type II      for i in range(R):          for j in range(S):              x[i, j] = solver.IntVar(0, solver.infinity(), f'x\_{i}\_{j}') # x[i][j]: Trips of Type I Bus on route i, shift j              y[i, j] = solver.IntVar(0, solver.infinity(), f'y\_{i}\_{j}') # y[i][j]: Trips of Type II Bus on route i, shift j              buses\_type1[i, j] = solver.IntVar(0, solver.infinity(), f'buses\_type1\_{i}\_{j}')  # Buses Tipo I              buses\_type2[i, j] = solver.IntVar(0, solver.infinity(), f'buses\_type2\_{i}\_{j}')  # Buses Tipo II      # Splitting Data into Demand and TripFactor      D = data.iloc[:, 1:5]  #  # Extract columns D1 to D4 (Indexes 1 to 4) Demand      T = data.iloc[:, 5:9]  # Extract columns T1 to T4 (Indexes 5 to 8) Trip Factor      # Constraint: Non-Negativity is also handled by IntVar(0, solver.infinity())      # --- Objective Function ---      objective = solver.Sum(Cost1\*x[i, j] + Cost2\*y[i, j] for i in range(R) for j in range(S)) # Minimize Cost of Trips in a day      solver.Minimize(objective)      # --- Constraints ---      # 1. Demand Satisfaction      for i in range(R):          for j in range(S):              solver.Add(C1\*x[i, j] + C2\*y[i, j] >= D.iloc[i,j])        # 2. Constraint Trip and Bus Relation to ensure Reuse of Buses during shift      for i in range(R):          for j in range(S):              T\_factor = T.iloc[i, j] # T\_factor = min(T.iloc[i, j], w[j])              # For Tipo I: buses\_type1[i, j] >= x[i, j] / capacity              solver.Add(buses\_type1[i, j] \* T\_factor >= x[i, j])              # For Tipo II: buses\_type2[i, j] >= y[i, j] / capacity              solver.Add(buses\_type2[i, j] \* T\_factor >= y[i, j])      # 3. Capacity: Fleet Availability for Bus Type-I + Type-II      for j in range(S):              solver.Add(                  solver.Sum(buses\_type1[i, j] + buses\_type2[i, j] for i in range(R)) <= (N1 + N2)              )      # 4. Capacity: Fleet Availability for Bus Type-I (Max N1)      for j in range(S):              solver.Add(                  solver.Sum(buses\_type1[i, j] for i in range(R)) <= N1              )      # 5. Capacity: Fleet Availability for Bus Type-II (Max N2)        for j in range(S):              solver.Add(                  solver.Sum(buses\_type2[i, j] for i in range(R)) <= N2              )      # 6. Customer Experience: Minimum Service Level      #for i in range(R):      #    for j in range(S):      #        solver.Add(x[i, j] + y[i, j] >= w[j])      # Solve the model      status = solver.Solve()      if status == pywraplp.Solver.OPTIMAL:     # Check if a solution was found          print("\nOptimal solution found!")            total\_cost = solver.Objective().Value()  # Optimized Cost from Solver        #--- Extracting results into a DataFrame ---          # Generate column names dynamically          column\_names = ["Route"]          for i in range(1, S + 1):              column\_names.append(f"Shift {i} - # Trips Bus T-1")              column\_names.append(f"Shift {i} - # Trips Bus T-2")              column\_names.append(f"Shift {i} - # Buses T-1")              column\_names.append(f"Shift {i} - # Buses T-2")          # Collect all results in a single DataFrame          results = []          Total\_cost=0          for i in range(R):              row\_data = [f"Route {i+1}"]              for j in range(S):                  t1\_trips = int(x[i, j].solution\_value())                  t2\_trips = int(y[i, j].solution\_value())                  buses\_T1 = int(buses\_type1[i, j].solution\_value())                  buses\_T2 = int(buses\_type2[i, j].solution\_value())                  row\_data.extend([t1\_trips, t2\_trips, buses\_T1, buses\_T2])              results.append(row\_data)          # Create the final output DataFrame          results\_df = pd.DataFrame(results, columns=column\_names)          # Append the totals row (sum for each numeric column)          totals = results\_df.iloc[:, 1:].astype(int).sum()          totals\_row = pd.DataFrame([["TOTAL"] + totals.tolist()], columns=column\_names)          # Append the totals row to the results dataframe          results\_df = pd.concat([results\_df, totals\_row], ignore\_index=True)          return results\_df, total\_cost      else:          print("\nNo optimal solution found.")          return None  # Define the file path  file\_path = r"C:\Users\Wilson\GUSCanada\Group2 Operations Analytics Project - Group2\Deliverables\demand.csv"  data = pd.read\_csv(file\_path) # Load the CSV Demand file into a pandas DataFrame  R=93 #total Routes  S=4 # daily Shifts  C1=60 #Capacity Bus Type I  C2=90 #Capacith Bus Type II  N1=600 # Fleet of Buses Type I  N2=90 # Fleet of Buses Type II  Cost1=200 # Weight to priorize by Bus Type-I (Updated)  Cost2=300 # Weight to priorize by Bus Type-II (Updated)  ST=[195,360,240,90] # Shifts Time Length in Minutes [Shift 1, Shift 2, Shift 3, Shift 4]  M=30 # MWT: Max Wait Time: 30 Mins  # Define output file path  output\_file\_path = r"C:\Users\Wilson\GUSCanada\Group2 Operations Analytics Project - Group2\Deliverables\G2Results.csv"  # Call function with default parameters  final\_df, total\_cost = optimize\_bus\_routes(data,R,S,C1,C2,N1,N2,Cost1,Cost2, ST, M)  # Save to CSV  if final\_df is not None:  # Valid DataFrame?      final\_df.to\_csv(output\_file\_path, index=False)      print(final\_df.head())  # Display the first few rows of output      print(final\_df.tail())  # Display the last few rows of output      print(f"Total Optimized Cost: {total\_cost}")  else:      print("No results to save or display.") |

# Appendix A.2- Python script for additional considerations.

|  |
| --- |
| from ortools.linear\_solver import pywraplp  import math  import pandas as pd  import tkinter as tk  from tkinter import simpledialog, messagebox  def optimize\_bus\_routes(DD,D,R,S,C1,C2,N1,N2,Cost1,Cost2, ST, M):      w = [math.ceil(length / M) for length in ST] # Calculate trips per shift (rounding up)      SP=[0.4,0.2,0.35,0.05]      # Create the solver SCIP      solver = pywraplp.Solver.CreateSolver('SCIP')      if not solver:          print("Impossible to create the Solver.")          exit()      # --- Variables ---      x = {}      y = {}      buses\_type1 = {}  # Number of buses Type I      buses\_type2 = {}  # Number of buses Type II      for i in range(R):          for j in range(S):              x[i, j] = solver.IntVar(0, solver.infinity(), f'x\_{i}\_{j}') # x[i][j]: Trips of Type I Bus on route i, shift j              y[i, j] = solver.IntVar(0, solver.infinity(), f'y\_{i}\_{j}') # y[i][j]: Trips of Type II Bus on route i, shift j              buses\_type1[i, j] = solver.IntVar(0, solver.infinity(), f'buses\_type1\_{i}\_{j}')  # Buses Type I              buses\_type2[i, j] = solver.IntVar(0, solver.infinity(), f'buses\_type2\_{i}\_{j}')  # Buses Type II      # Splitting Data into Demand and TripFactor      T = data.iloc[:, 5:9]  # Extract columns T1 to T4 (Indexes 5 to 8) Trip Factor      Pr=data.iloc[:, 9:10] # Extract columns Route Damand Portion (Indexes 5 to 8) Trip Factor      D = pd.DataFrame([[round(DD \* pr \* SP[j]) for j in range(S)] for pr in Pr.iloc[:, 0]], columns=[f"D{j+1}" for j in range(S)])  # Calculate D[i,j] as ceiling of DD \* Pr[i] \* SP[j]        # Define output file path      output\_file\_pathD = r"C:\Users\Wilson\GUSCanada\Group2 Operations Analytics Project - Group2\Deliverables\DemandPr.csv"      Pr.to\_csv(output\_file\_pathD, index=False)      # Constraint: Non-Negativity is also handled by IntVar(0, solver.infinity())      # --- Objective Function ---      objective = solver.Sum(Cost1\*x[i, j] + Cost2\*y[i, j] for i in range(R) for j in range(S)) # Minimize Cost of Trips in a day      solver.Minimize(objective)      # --- Constraints ---      # 1. Demand Satisfaction      for i in range(R):          for j in range(S):              solver.Add(C1\*x[i, j] + C2\*y[i, j] >= D.iloc[i,j])        # 3. Capacity: Fleet Availability for Bus Type-I + Type-II      for j in range(S):              solver.Add(                  solver.Sum(buses\_type1[i, j] + buses\_type2[i, j] for i in range(R)) <= (N1 + N2)              )      # 4. Capacity: Fleet Availability for Bus Type-I (Max N1)      for j in range(S):              solver.Add(                  solver.Sum(buses\_type1[i, j] for i in range(R)) <= N1              )      # 5. Capacity: Fleet Availability for Bus Type-II (Max N2)        for j in range(S):              solver.Add(                  solver.Sum(buses\_type2[i, j] for i in range(R)) <= N2              )      # 6. Customer Experience: Minimum Service Level for low demand routes      for i in range(R):          for j in range(S):              solver.Add(x[i, j] + y[i, j] >= w[j])      # 2. Constraint Trip and Bus Relation to ensure Reuse of Buses during shift      for i in range(R):          for j in range(S):              T\_factor = T.iloc[i, j] # T\_factor = min(T.iloc[i, j], w[j])              # For Tipo I: buses\_type1[i, j] >= x[i, j] / capacity              solver.Add(buses\_type1[i, j] \* T\_factor >= x[i, j])              # For Tipo II: buses\_type2[i, j] >= y[i, j] / capacity              solver.Add(buses\_type2[i, j] \* T\_factor >= y[i, j])      # Solve the model      status = solver.Solve()      if status == pywraplp.Solver.OPTIMAL:     # Check if a solution was found          print("\nOptimal solution found!")            total\_cost = solver.Objective().Value()  # Optimized Cost from Solver        #--- Extracting results into a DataFrame ---          # Generate column names dynamically          column\_names = ["Route"]          for i in range(1, S + 1):              column\_names.append(f"Shift {i} - # Trips Bus T-1")              column\_names.append(f"Shift {i} - # Trips Bus T-2")              column\_names.append(f"Shift {i} - # Buses T-1")              column\_names.append(f"Shift {i} - # Buses T-2")          # Collect all results in a single DataFrame          results = []          Total\_cost=0          for i in range(R):              row\_data = [f"Route {i+1}"]              for j in range(S):                  t1\_trips = int(x[i, j].solution\_value())                  t2\_trips = int(y[i, j].solution\_value())                  buses\_T1 = int(buses\_type1[i, j].solution\_value())                  buses\_T2 = int(buses\_type2[i, j].solution\_value())                  row\_data.extend([t1\_trips, t2\_trips, buses\_T1, buses\_T2])              results.append(row\_data)          # Create the final output DataFrame          results\_df = pd.DataFrame(results, columns=column\_names)          # Append the totals row (sum for each numeric column)          totals = results\_df.iloc[:, 1:].astype(int).sum()          totals\_row = pd.DataFrame([["TOTAL"] + totals.tolist()], columns=column\_names)          # Append the totals row to the results dataframe          results\_df = pd.concat([results\_df, totals\_row], ignore\_index=True)          return results\_df, total\_cost      else:          print("\nNo optimal solution found.")          return None  # Define the file path  file\_path = r"C:\Users\Wilson\GUSCanada\Group2 Operations Analytics Project - Group2\Deliverables\Pr.csv"  data = pd.read\_csv(file\_path) # Load the CSV Demand file into a pandas DataFrame  R=93 #total Routes  S=4 # daily Shifts  C1=60 #Capacity Bus Type I  C2=90 #Capacith Bus Type II  N1=600 # Fleet of Buses Type I  N2=90 # Fleet of Buses Type II  Cost1=200 # Weight to priorize by Bus Type-I (Updated)  Cost2=300 # Weight to priorize by Bus Type-II (Updated)  ST=[195,360,240,90] # Shifts Time Length in Minutes [Shift 1, Shift 2, Shift 3, Shift 4]  M=30 # MWT: Max Wait Time: 30 Mins  F=5 #Fare  #Ask Demand D  root = tk.Tk()  root.withdraw()  DD = simpledialog.askinteger("Input Demand", "Enter daily demand:", minvalue=0)  if DD is None:      messagebox.showerror("Error", "Demand input canceled. Exiting.")      DD = 0  # Default Value  root.destroy()  # Define output file path  output\_file\_path = r"C:\Users\Wilson\GUSCanada\Group2 Operations Analytics Project - Group2\Deliverables\G2Results.csv"  # Call function with default parameters  final\_df, total\_cost = optimize\_bus\_routes(DD,data,R,S,C1,C2,N1,N2,Cost1,Cost2, ST, M)  # Save to CSV  if final\_df is not None:  # Valid DataFrame?      final\_df.to\_csv(output\_file\_path, index=False)      print(final\_df.head())  # Display the first few rows of output      print(final\_df.tail())  # Display the last few rows of output      print(f"Total Optimized Cost: {total\_cost}")      totals\_row = final\_df.iloc[-1]  # Last row contains totals      # Calculate sums from the columns in final\_df      total\_trips\_t1 = totals\_row[[col for col in final\_df.columns if 'Trips Bus T-1' in col]].sum()      total\_trips\_t2 = totals\_row[[col for col in final\_df.columns if 'Trips Bus T-2' in col]].sum()      total\_buses\_t1 = totals\_row[[col for col in final\_df.columns if 'Buses T-1' in col]].sum()      total\_buses\_t2 = totals\_row[[col for col in final\_df.columns if 'Buses T-2' in col]].sum()      total\_trips = total\_trips\_t1 + total\_trips\_t2      total\_buses = total\_buses\_t1 + total\_buses\_t2      # Create summary table      summary\_data = {          "Metric": ["Total Trips T1", "Total Trips T2", "Total Buses T1", "Total Buses T2",                     "Total Trips", "Total Buses","Frequency (Mins)", "Total Cost","Daily Riders Demand","Fare","Revenue"],          "Value": [total\_trips\_t1, total\_trips\_t2, total\_buses\_t1, total\_buses\_t2,                    total\_trips, total\_buses,M, total\_cost,DD,F,F\*DD]      }      summary\_df = pd.DataFrame(summary\_data)        # Save and display summary      print("\nSummary Table:")      print(summary\_df)  else:      print("No results to save or display.") |

# Appendix B - Individual Contribution & AI Usage sheet

|  |  |  |  |
| --- | --- | --- | --- |
| Student Name | Role/Responsibility | Contribution Details | Use of AI Tools (if applicable) |
| Ameyaw, Kofi, | Presentation | Power Point and Coordination in Teams |  |
| Wilson Erique | Coding | Mathematical Modeling and Code Initial Dataset Excel Graphs in Excel | PyCharm, Used Chat GPT for coding suggestions, optimization, and debugging. |
| Nematov, Abdulkhodiy | Report | Research on Solvers and Coding Results and Graphs in Excel and Graphs insight report | Used Excel and PyCharm |
| Shrestha, Kriti Bahadur | Dataset | Extended applicability coding | Used Excel and Chat GPT for Missing values |

# APPENDIX C

The Following table present the daily demand of users for 93 routes and 4 Shifts of the day, the values had been obtained from the proportionality described in the case study document, also from the document the Trip Factor per route per shift, since the shifts have different durations through the day. D1,D2,D3,D4 show the number of users per route per shift in a day, while T1,T2,T3,T4 show the number of trips a single bus can complete the route in each shift. Since the article had incomplete rows, we have generated the demand respecting the proportionality of the available information with Random proportions per missed routes data.

| **Route** | **D1** | **D2** | **D3** | **D4** | **T1** | **T2** | **T3** | **T4** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1650 | 825 | 1444 | 206 | 7 | 12 | 8 | 3 |
| 2 | 1399 | 699 | 1224 | 175 | 4 | 7 | 5 | 2 |
| 3 | 4412 | 2206 | 3861 | 552 | 4 | 7 | 5 | 2 |
| 4 | 1212 | 606 | 1060 | 151 | 3 | 5 | 3 | 1 |
| 5 | 1404 | 702 | 1229 | 176 | 5 | 8 | 6 | 2 |
| 6 | 443 | 221 | 387 | 55 | 4 | 7 | 5 | 2 |
| 7 | 1404 | 702 | 1229 | 176 | 7 | 11 | 8 | 3 |
| 8 | 2009 | 1004 | 1758 | 251 | 7 | 11 | 8 | 3 |
| 9 | 1814 | 907 | 1588 | 227 | 4 | 7 | 5 | 2 |
| 10 | 292 | 146 | 255 | 36 | 6 | 10 | 7 | 3 |
| 11 | 497 | 248 | 435 | 62 | 3 | 5 | 3 | 1 |
| 12 | 346 | 173 | 302 | 43 | 6 | 10 | 7 | 3 |
| 13 | 216 | 108 | 189 | 27 | 5 | 8 | 6 | 2 |
| 14 | 810 | 405 | 709 | 101 | 6 | 10 | 7 | 3 |
| 15 | 1264 | 632 | 1106 | 158 | 7 | 11 | 8 | 3 |
| 16 | 1609 | 805 | 1408 | 201 | 4 | 7 | 5 | 2 |
| 17 | 227 | 113 | 198 | 28 | 4 | 7 | 5 | 2 |
| 18 | 367 | 184 | 321 | 46 | 4 | 7 | 5 | 2 |
| 19 | 1350 | 675 | 1181 | 169 | 4 | 7 | 5 | 2 |
| 20 | 1296 | 648 | 1134 | 162 | 6 | 10 | 7 | 3 |
| 21 | 670 | 335 | 586 | 84 | 5 | 8 | 6 | 2 |
| 22 | 670 | 335 | 586 | 84 | 7 | 11 | 8 | 3 |
| 23 | 691 | 346 | 605 | 86 | 7 | 11 | 8 | 3 |
| 24 | 929 | 464 | 813 | 116 | 7 | 11 | 8 | 3 |
| 25 | 1253 | 626 | 1096 | 157 | 5 | 8 | 6 | 2 |
| 26 | 756 | 378 | 662 | 95 | 7 | 11 | 8 | 3 |
| 27 | 972 | 486 | 851 | 122 | 5 | 8 | 6 | 2 |
| 28 | 1123 | 562 | 983 | 140 | 7 | 11 | 8 | 3 |
| 29 | 1037 | 518 | 907 | 130 | 7 | 11 | 8 | 3 |
| 30 | 324 | 162 | 284 | 41 | 7 | 11 | 8 | 3 |
| 31 | 508 | 254 | 444 | 63 | 3 | 5 | 3 | 1 |
| 32 | 918 | 459 | 803 | 115 | 7 | 11 | 8 | 3 |
| 33 | 2581 | 1291 | 2259 | 323 | 7 | 11 | 8 | 3 |
| 34 | 346 | 173 | 302 | 43 | 4 | 7 | 5 | 2 |
| 35 | 907 | 454 | 794 | 113 | 5 | 8 | 6 | 2 |
| 36 | 3002 | 1501 | 2627 | 375 | 4 | 7 | 5 | 2 |
| 37 | 2225 | 1112 | 1947 | 278 | 7 | 11 | 8 | 3 |
| 38 | 1512 | 756 | 1323 | 189 | 7 | 11 | 8 | 3 |
| 39 | 572 | 286 | 501 | 72 | 7 | 11 | 8 | 3 |
| 40 | 356 | 178 | 312 | 45 | 6 | 10 | 7 | 3 |
| 41 | 713 | 356 | 624 | 89 | 4 | 7 | 5 | 2 |
| 42 | 205 | 103 | 180 | 26 | 7 | 11 | 8 | 3 |
| 43 | 1318 | 659 | 1153 | 165 | 7 | 11 | 8 | 3 |
| 44 | 378 | 189 | 331 | 47 | 7 | 11 | 8 | 3 |
| 45 | 680 | 340 | 595 | 85 | 5 | 8 | 6 | 2 |
| 46 | 410 | 205 | 359 | 51 | 3 | 5 | 3 | 1 |
| 47 | 2041 | 1021 | 1786 | 255 | 8 | 13 | 9 | 4 |
| 48 | 1523 | 761 | 1332 | 190 | 5 | 8 | 6 | 2 |
| 49 | 961 | 481 | 841 | 120 | 7 | 11 | 8 | 3 |
| 50 | 310 | 155 | 271 | 39 | 7 | 11 | 8 | 3 |
| 51 | 950 | 475 | 832 | 119 | 7 | 11 | 8 | 3 |
| 52 | 2765 | 1382 | 2419 | 346 | 7 | 11 | 8 | 3 |
| 53 | 1361 | 680 | 1191 | 170 | 6 | 10 | 7 | 3 |
| 54 | 1037 | 518 | 907 | 130 | 6 | 10 | 7 | 3 |
| 55 | 2786 | 1393 | 2438 | 348 | 7 | 11 | 8 | 3 |
| 56 | 929 | 464 | 813 | 116 | 7 | 11 | 8 | 3 |
| 57 | 940 | 470 | 822 | 117 | 8 | 13 | 9 | 4 |
| 58 | 1944 | 972 | 1701 | 243 | 6 | 10 | 7 | 3 |
| 59 | 454 | 227 | 397 | 57 | 7 | 11 | 8 | 3 |
| 60 | 2163 | 1082 | 1893 | 270 | 7 | 11 | 8 | 3 |
| 61 | 454 | 227 | 397 | 57 | 6 | 10 | 7 | 3 |
| 62 | 605 | 302 | 529 | 76 | 3 | 5 | 3 | 1 |
| 63 | 1048 | 524 | 917 | 131 | 5 | 8 | 6 | 2 |
| 64 | 648 | 324 | 567 | 81 | 3 | 5 | 3 | 1 |
| 65 | 1663 | 832 | 1455 | 208 | 4 | 7 | 5 | 2 |
| 66 | 886 | 443 | 775 | 111 | 5 | 8 | 6 | 2 |
| 67 | 1544 | 772 | 1351 | 193 | 9 | 15 | 10 | 4 |
| 68 | 364 | 182 | 319 | 46 | 4 | 7 | 5 | 2 |
| 69 | 381 | 191 | 334 | 48 | 7 | 11 | 8 | 3 |
| 70 | 864 | 432 | 756 | 108 | 5 | 8 | 6 | 2 |
| 71 | 810 | 405 | 709 | 101 | 7 | 11 | 8 | 3 |
| 72 | 454 | 227 | 397 | 57 | 4 | 7 | 5 | 2 |
| 73 | 497 | 248 | 435 | 62 | 3 | 5 | 3 | 1 |
| 74 | 756 | 378 | 662 | 95 | 7 | 11 | 8 | 3 |
| 75 | 961 | 481 | 841 | 120 | 4 | 7 | 5 | 2 |
| 76 | 2020 | 1010 | 1767 | 252 | 6 | 10 | 7 | 3 |
| 77 | 1274 | 637 | 1115 | 159 | 7 | 11 | 8 | 3 |
| 78 | 713 | 356 | 624 | 89 | 4 | 7 | 5 | 2 |
| 79 | 572 | 286 | 501 | 72 | 7 | 11 | 8 | 3 |
| 80 | 1814 | 907 | 1588 | 227 | 4 | 7 | 5 | 2 |
| 81 | 54 | 27 | 47 | 7 | 4 | 7 | 5 | 2 |
| 82 | 691 | 346 | 605 | 86 | 7 | 11 | 8 | 3 |
| 83 | 648 | 324 | 567 | 81 | 5 | 8 | 6 | 2 |
| 84 | 1912 | 956 | 1673 | 239 | 7 | 11 | 8 | 3 |
| 85 | 1642 | 821 | 1436 | 205 | 6 | 10 | 7 | 3 |
| 86 | 562 | 281 | 491 | 70 | 6 | 10 | 7 | 3 |
| 87 | 648 | 324 | 567 | 81 | 7 | 11 | 8 | 3 |
| 88 | 1145 | 572 | 1002 | 143 | 7 | 11 | 8 | 3 |
| 89 | 1696 | 848 | 1484 | 212 | 7 | 11 | 8 | 3 |
| 90 | 2894 | 1447 | 2533 | 362 | 6 | 10 | 7 | 3 |
| 91 | 259 | 130 | 227 | 32 | 4 | 7 | 5 | 2 |
| 92 | 1534 | 767 | 1343 | 192 | 4 | 8 | 5 | 2 |
| 93 | 812 | 406 | 710 | 101 | 5 | 9 | 6 | 2 |

Table 14: Daily Demand and Trip Factor per Route and shift.

# References

Berhan, E., Mengistu, D., Beshah, B., & Kitaw, D. (2014). Modeling and analysis of bus scheduling systems of urban public bus transport. International Journal of Computer Information Systems and Industrial Management Applications, 6, 404-412. Retrieved from <http://www.mirlabs.net/ijcisim/index.html>

Hendel, G. (2018, March 6). *Introduction to SCIP* (pp. 13–14). <https://www.scipopt.org/workshop2018/SCIP-Intro.pdf>

GOOGLE OR TOOLS. (2024, August 28). Solving an assignment problem. *Google for Developers*. <https://developers.google.com/optimization/assignment/assignment_example>