PLANNING ELECTRIC SCOOTERS IN SMART CITIES

DENİZ FISTIKCI

ALİ SENCER BÖLÜKBAŞ

SANEM KILIÇ

DORUK DEMİR

ADVISOR: ÇİLEM KAHRAMAN TOPGÜLOĞLU

SUMMARY SHEET

In big cities with multiple districts, public transport cannot be sufficiently allocated to various areas due to differing demands, needs, and existing infrastructure scattered around the city. Therefore, allocating vehicles in these areas would result in a disproportionate and problematic distribution. In order to mitigate these issues, a suitable model that classifies the need of each area through different factors and monitors daily changes is needed to ensure transportation functions as required.

In the given situation, the model is not only supposed to ensure that the scooters are distributed evenly, but it also has to be able to mitigate unexpected changes in factors and daily imbalances. The assigned model provides various solutions to the distribution problem, analyses the daily and weekly cost of charging and redistribution, and considers potential relief plans during unexpected situations such as strikes and festivals. The model consists of - parts:

First is the analysis of each factor and how much they affect the scooter demand. This was needed mainly due to the fact that the importance of each factor was unknown and had to be defined. This was done through the AHP method, which helped compare the coefficients of each factor and comparing their differing values in each region. Then, the consistency of the AHP method had to be checked in order to validate the used eigenvector in the method, which was found to be consistent. Thus, the scooter numbers of each region were approximately found and then rounded to the nearest integer.

The second section consists of the daily and weekly analysis of cost and location changes. This was done by comparing the daily influence of each area through tourist density and using a differential-equation / jump-process formulation in order to define the daily imbalance of scooters. Thereafter, the daily and weekly cost of charging and relocating the scooters had to be calculated. In this process the Monte Carlo Method was used, as it provided example daily values of how many scooters had to be carried or charged. Later, an another more detailed model was run with more iterations, whose values were used to find the weekly cost.

The last part includes a Linear Programming (LP) Model, which optimized overnight scooter movements. In this process, the movement costs were prioritized due to its higher cost compared to charging costs. This model was utilized to ensure maximum feasibility and expected number of trips. The LP was solved across repeated cost caps, generating a Pareto Curve which yielded how much total cost changed compared to the number of total trips. The curve was then used to determine the regions with the maximum output. Finally, the performance of the model was tested in 2 different scenarios: a festival and a strike, in order to determine if the framework can mitigate the impacts of atypical situations with minimal intervention.

TABLE OF CONTENTS

1. INTRODUCTION

1.1. DEFINITION OF THE SMART CITIES AND ELECTRICAL SCOOTERS

To plan the use of electrical scooters in smart cities, different definitions need to be made. Firstly, a smart city can be defined as a city that integrates digital technology into its networks, infrastructure, and services in order to become more efficient and livable. In the Problem Guideline, it is stated that the use of electrical scooters needs to be considered. From this perspective, electrical scooters can be defined as personal mobility devices that utilize electric power for short-distance transportation.

1.2. ASSUMPTIONS

To simplify the problem, the following assumptions have been made. These assumptions have been used throughout the report paper to ensure it reflects the real-life situation as closely as possible. The assumptions and their justifications are presented in Table 1.

Number of Assumption	Assumption	Justification
1	The number of electric scooters remains constant at 500 electric scooters and none sustain physical damage.	In real-world settings, the vehicles in use typically do not require immediate repairs, and the number of scooters remains constant. Therefore, it is reasonable to assume that none of the scooters will need to be removed or repaired during the time period considered in this paper. This assumption allows the model to focus solely on the movement and distribution of scooters without accounting for additional conditions not specified.
2	At least 35% of electric scooters change regions at least once per day.	The assumption that at least 35% of scooters will change regions is taken directly from the problem statement.
3	Interregional transfers can occur only between neighboring regions.	In real life, a land vehicle cannot travel directly to a non-neighboring area without passing through a neighboring one. Therefore, it is sensible to assume that transfers can occur only between neighboring regions.
4	The transition probabilities are inversely proportional to tourist appeal and are controlled by a variable.	As shown in the factor value tables and in the calculation of the weight coefficients, the algorithm functions only when

Number of Assumption	Assumption	Justification
		positive values are used. Therefore, transition probabilities and tourist appeal are taken to be inversely proportional: when one has the highest value (7), the other takes the lowest value (1).
5	In matrix N, neighboring regions have a value of 1, while non-neighboring regions have a value of 0.	For the matrix N to function as intended, each value in the matrix must carry the value 1 or 0, hence 1 was assigned for the neighboring regions (positive value) and 0 for non-neighboring regions.
6	Usage intensity does not differ between days.	Due to the lack of information on day-to-day changes in usage intensity, it is assumed that usage intensity does not vary across days.
7	Every scooter goes with the same speed which is 10 ms^{-1}.	To determine the daily distance traveled by the scooters, their speed must be known. Based on typical average scooter speeds, the speed is set accordingly.
8	The scooters will be charged when their batteries are below 20%.	To find the scooters' range, the charge percentage is needed. The charge percentage is assumed based on daily-life conditions.
9	Each scooter has the same probability of being charged.	The scooters' charging process depends on whether they need charging (1) or not (0). This is treated as a binary classification.

Table 1: Assumptions and Justifications

1.3. DETERMINING FACTORS AND REASONINGS

The factors for the first question are specified in the Preliminary Screening Problem guideline. These factors are presented in Table 2.

Area	Daily Average Passenger Traffic	Student Rate	Tourist Rate	Public Transportation Access
A	Very High	Medium	Low	Very Good
B	Medium	High	Low	Medium
C	Low	Low	Very High	Weak
D	Medium	Medium	Medium	Good

Area	Daily Average Passenger Traffic	Student Rate	Tourist Rate	Public Transportation Access
E	Low	High	Low	Weak

Table 2: Factors Selected by IMMC Guidelines

Abbreviations have been used for the factor names to simplify the notation; these are presented in Table 3.

Daily Average Passenger Traffic	PT
Student Rate	S
Tourist Rate	T
Public Transportation Access	PA

Table 3: Notation Used for Factors

Daily average passenger traffic, student rate, and tourist rate are directly proportional to the demand for electric scooters, whereas public transportation access is inversely proportional. To model the demand for electric scooters more effectively, all the given properties (e.g. weak, low, medium) of the factors are converted into numerical values. For this conversion, a score is assigned for each property and presented in Table 4.

Weak	1
Low	2
Medium	3
Good	4
Very Good	5
High	6
Very High	7

Table 4: Scores Given to Factors

After the scores are assigned, Public Transportation Access is converted into Need for Public Transportation. To perform this conversion, the following calculation is used.

$$8 - PA = Need\ For\ Public\ Transportation$$

After applying equation to all scores, the final factor values are obtained. The notation for Need for Public Transportation is defined as NP. The finalized factors are presented in Table 5.

Area	Daily Average Passenger Traffic (PT)	Student Rate (S)	Tourist Rate (T)	Need for Public Transportation (NP)
A	7	3	2	3
B	3	6	2	5
C	2	2	7	7
D	3	3	3	4
E	2	6	2	7

Table 5: Finalized Factors and Relation Scores

2. CREATION OF MODELS

2.1. PRELIMINARY MODEL 1: DETERMINING WEIGHTS WITH AHP METHOD

2.1.1. DETERMINING WEIGHTS WITH AHP METHOD

- A hierarchical structure among the factors was established by rating and comparing the importance of each factor.
- A pairwise comparison matrix (A) was created for the factors based on the ratios of their values.
- The weight vector W was calculated using the normalized eigenvector.
- The consistency of the method was evaluated using the Consistency Index (CI) and Consistency Ratio (CR). If these values do not indicate acceptable consistency, the process is repeated with revised values, starting again from Step 1.

To determine the weights, the relative importance of the factors was rated. The relative importance ratings are shown in Table 6.

Criteria	PT	T	NP	S
PT	1	1.8	3.2	4.05
T	$\frac{1}{1.8}$	1	$\frac{1}{1.6}$	1.4
NP	$\frac{1}{3.2}$	$\frac{1}{1.6}$	1	1.4
S	$\frac{1}{4.05}$	$\frac{1}{2.2}$	$\frac{1}{1.4}$	1

Table 6: Relative Importance Rankings

After ranking the relative importance of the factors, the table is presented in a 4×4 matrix form.

$$A = \begin{bmatrix} 1.00 & 1.80 & 3.20 & 4.05 \\ 0.55 & 1.00 & 0.62 & 1.40 \\ 0.31 & 0.62 & 1.00 & 1.40 \\ 0.24 & 0.45 & 0.71 & 1.00 \end{bmatrix}$$

The values in each row are summed.

$$R_{1_{sum}} = \sum_j a_{1j} = PT_{sum} = 1.00 + 1.80 + 3.20 + 4.05 = 10.05$$

$$R_{2_{sum}} = \sum_j a_{2j} = T_{sum} = 0.55 + 1.00 + 0.62 + 1.40 = 5.36$$

$$R_{3_{sum}} = \sum_j a_{3j} = NP_{sum} = 0.31 + 0.62 + 1.00 + 1.40 = 3.34$$

$$R_{4_{sum}} = \sum_j a_{4j} = S_{sum} = 0.24 + 0.45 + 0.71 + 1.00 = 2.42$$

In the following equation, all rows are summed.

$$R_{total} = \sum_i \sum_j a_{ij} = R_{1_{sum}} + R_{2_{sum}} + R_{3_{sum}} + R_{4_{sum}} = 21.16$$

Therefore, the initial weights can be obtained using the normalization method by applying the following calculations with the general formula.

$$W_{n_1} = \frac{R_{n_{sum}}}{R_{total}} = \frac{\sum_j a_{1j}}{\sum_i \sum_j a_{ij}}$$

$$W_{1_1} = \frac{10.05}{21.16} = 0.4749$$

$$W_{1_2} = \frac{5.36}{21.16} = 0.2531$$

$$W_{1_3} = \frac{3.31}{21.16} = 0.1577$$

$$W_{1_4} = \frac{2.42}{21.16} = 0.1142$$

This method is repeated seven times to normalize the weights and obtain the optimal values. The final weights after normalization are as follows:

$$W_{1_7} = \frac{1.8973}{4.0014} = 0.47416$$

$$W_{2_7} = \frac{1.0221}{4.0014} = 0.25505$$

$$W_{3_7} = \frac{0.6243}{4.0014} = 0.1560$$

$$W_{4_7} = \frac{0.4592}{4.0014} = 0.11476$$

The sum of the coefficients is calculated to verify the normalization process.

$$W_{1_7} + W_{2_7} + W_{3_7} + W_{4_7} = 1$$

The sum of the coefficients is 1, which indicates that the normalization process was performed correctly. After repeating the procedure seven times, the weights were

observed to stabilize and converge to their final values. To confirm this convergence, the maximum difference between the final values was calculated.

$$\Delta^{(k)} = \max_i \left| w_i^{(k)} - w_i^{(k-1)} \right|$$

$$k: \ value\ in\ k^{th}\ step$$

$$i: factor\ index$$

The maximum difference is calculated at each step. Only the final maximum-difference calculation is shown here; the others are provided in the appendix. The maximum difference between normalization steps 7 and 6 is found by subtracting the corresponding values, taking the absolute value of each difference, and then selecting the largest one. Absolute values are used to ensure the differences are non-negative.

$$\Delta^7 = 8.8204 \times 10^{-12}$$

The maximum difference is smaller than 10^{-10}, indicating that the change is sufficiently small to conclude that the next normalized values will not change significantly. This shows that the weights have converged and align with the eigenvector direction. Finally, λ_{max} is calculated to assess the consistency of the AHP results.

$$\lambda_i = \frac{(Aw)_i}{w_i}$$

$$A: \ AHP\ Matrix$$

$$w_i: \ weight\ obtained\ after\ normalization\ (for\ the\ i-th\ factor)$$

$$\lambda_i = 4.00138$$

The spread is calculated to determine how much the λ_i values differ from one another:

$$spread = \ \max(\lambda_i) - \min(\lambda_i)$$

The spread indicates whether the values are close to the eigenvector solution. Below, the spread for the 7th and 6th normalization steps is shown. The spread values for the other normalization steps are provided in the appendix.

$$\max(4.00138) - \min(4.00138) = 4.00854 \times 10^{-11}$$

Due to rounding, the exact values of λ_i are not fully represented. The convergence of the spread values across iterations indicates that the model has reached its optimal eigenvector value.

2.1.2. CONTROLLING THE CONSISTENCY OF THE AHP

To assess the consistency of the AHP model, the Consistency Index (CI) and the Consistency Ratio (CR) are calculated. The Consistency Index (CI) measures the degree of inconsistency in the pairwise comparison matrix. To compute CI, $\lambda_{\max}$ is first determined.

Constituency index (CI) is a statistical measurement used to assess the reliability and stability. (Science Direct, 2026)

$$\lambda_{max} = \frac{1}{n}\sum_{i=1}^{n} \lambda_i$$

$$\lambda_{max} = 4.00138$$

$$CI = \frac{\lambda_{max} - n}{n - 1}$$

$$n = \ size\ of\ the\ matrix$$

$$CI = \frac{4.00138 - 4}{3} = 0.00046$$

The Consistency Index is close to 0, indicating that the pairwise comparisons are highly consistent (i.e., the AHP matrix is nearly perfectly consistent).

$$CR = \frac{CI}{RI}$$

The Consistency Ratio (CR) indicates how inconsistent the judgments are compared with purely random judgments, as represented by the Random Index (RI). The Random Index (RI) is a constant that depends on the size (order) of the matrix. Since the AHP pairwise comparison matrix A has size n = 4, the corresponding RI value is 0.90.

$$CR = \frac{0.00046}{0.9} = 0.00051 = \ 0.0512$$

The CR value is smaller than 0.1, which indicates that the matrix is consistent. The near-zero values of both CI and CR confirm that the decision-making process is logically consistent, thereby validating the extracted eigenvector as the optimal weight vector for the model's regional allocations.

2.1.3. DETERMINING NUMBER OF SCOOTERS FOR EACH AREA

The general formula for the demand function is given in the following equation:

$$y = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$$

The calculated weights are substituted into the equation, and the final demand function is obtained, as shown below:

$$y = 0.47416x_{PT} + 0.25505x_T + 0.15601x_{NP} + 0.11476x_S$$

$$x_{PT} : x\ values\ of\ public\ transportation\ need$$

$$x_T : x\ values\ of\ tourist\ density$$

$$x_{NP} : x\ values\ of public\ transportation\ need$$

$$x_S : x\ values\ of student\ proportion$$

The calculated weights are substituted into the demand function, as shown in the general formula. Tables 7, 8, 9, and 10 present the calculations used to determine the demand score for each factor.

Areas	Passenger Volume	Weight	Score
A	7	0.474163291	3.319143034
B	3	0.474163291	1.422489872
C	2	0.474163291	0.948326581
D	3	0.474163291	1.422489872
E	2	0.474163291	0.948326581

Table 7: Calculations Of Passenger Volume

Areas	Tourist density	Weight	Score
A	2	0.255058433	0.510116866
B	2	0.255058433	0.510116866
C	7	0.255058433	1.785409033
D	3	0.255058433	0.765175300
E	2	0.255058433	0.510116866

Table 8: Calculations Of Tourist Density

Areas	Public Transportation Need	Weight	Score
A	3	0.156013971	0.468041913
B	5	0.156013971	0.780069855
C	7	0.156013971	1.092097797
D	4	0.156013971	0.624055884
E	7	0.156013971	1.092097797

Table 9: Calculations Of Public Transportation Need

Areas	Student Proportion	Weight	Score
A	3	0.114764305	0.344292915
B	6	0.114764305	0.688585831
C	2	0.114764305	0.22952861
D	3	0.114764305	0.344292915
E	6	0.114764305	0.688585831

Table 10: Calculations Of Student Proportion

As a result of summing the values in Tables 6, 7, 8, and 9, the final demand score (y) for each area is calculated and presented in Table 11.

Area	Result (y)
A	4.641594729
B	3.401262424
C	4.055362021
D	3.156013971
E	3.239127076

Table 11: Results Of Demand Function of Area

Area	Expended Y values	Rounded Y Values
A	125.4935467	125
B	91.95901620	92
C	109.6437309	110
D	85.32829981	85
E	87.57540644	88
Total	**27.03673070**	**500**

Table 12: Final Value of Distributed Number of Scooters

Table 12 presents the expanded demand scores (y values) for each area and their rounding to the nearest integer. Accordingly, the final number of scooters allocated to each area is determined. These values also represent the areas' demand scores. The maximum total demand is 500, because 500 is the total number of scooters available. Therefore, the number of scooters allocated to each area is calculated in proportion to each area's demand score.

2.2. PRELIMINARY MODEL 2: MODELLING END-OF-DAY IMBALANCE WITH DIFFERENTIAL EQUATIONS

To model the end-of-day imbalance using differential equations, the numbers of scooters in five regions are defined as variables.

$$S_A(t) \triangleq Number\ of\ Scooter\ Varies\ in\ Region\ A$$

$$S_B(t) \triangleq Number\ of\ Scooter\ Varies\ in\ Region\ B$$

$$S_C(t) \triangleq Number\ of\ Scooter\ Varies\ in\ Region\ C$$

$$S_D(t) \triangleq Number\ of\ Scooter\ Varies\ in\ Region\ D$$

$$S_E(t) \triangleq Number\ of\ Scooter\ Varies\ in\ Region\ E$$

After defining the variables, they are expressed in vector form.

$$S(t) = \begin{bmatrix} S_A(t) \\ S_B(t) \\ S_C(t) \\ S_D(t) \\ S_E(t) \end{bmatrix}$$

The initial values obtained previously in Model 1 are expressed in vector form.

$$S(0) = \begin{bmatrix} 125 \\ 92 \\ 110 \\ 85 \\ 88 \end{bmatrix}$$

To determine the imbalance between regions, neighboring regions are assumed, as also stated in the Assumptions and Justifications section. The following notation shows the neighborhood relationships between regions. The arrows indicate neighboring regions; for example, Region A's neighbors are Region B and Region D.

$$A \rightarrow B, D$$

$$B \rightarrow A, C, D$$

$$C \rightarrow B, D, E$$

$$D \rightarrow A, B, C, E$$

$$E \rightarrow C, D$$

The relationships are represented in matrix form as N. In this matrix, 1 indicates that two regions are neighbors, and 0 indicates otherwise.

$$N = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

According to the challenge guidelines, the largest scooter transfers should be directed to the regions with the highest tourist rate (T). The previously computed tourist-rate scores are used, as shown in the following equation:

$$(T_A, T_B, T_C, T_D, T_E) = (1, 1{,}5{,}3{,}1)$$

Since scooter movement depends on the tourist rate, the relative weights of the regions are calculated using the following equation:

$$P_{ij} = \begin{cases} \dfrac{w_{ij}}{\sum_{k \in N(i)} w_{ik}} & if\ j \in N(i) \\ 0 & otherwise \end{cases}$$

$$\sum_j P_{ij} = 1$$

Using this equation and the assumed neighborhood relationships between regions, the calculations are performed as follows. The results are shown in Table 12.

A	B	$w_{AB} = \frac{1}{1+3} = \frac{1}{4}$
	D	$w_{AD} = \frac{3}{1+3} = \frac{3}{4}$
B	A	$w_{BA} = \frac{1}{1+5+3} = \frac{1}{9}$
	C	$w_{BC} = \frac{5}{1+5+3} = \frac{5}{9}$
	D	$w_{BD} = \frac{3}{1+5+3} = \frac{3}{9}$
C	B	$w_{CB} = \frac{1}{1+3+1} = \frac{1}{5}$
	D	$w_{CD} = \frac{3}{1+3+1} = \frac{3}{5}$
	E	$w_{CE} = \frac{1}{1+3+1} = \frac{1}{5}$
D	A	$w_{DA} = \frac{1}{1+1+5+1} = \frac{1}{8}$
	B	$w_{DB} = \frac{1}{1+1+5+1} = \frac{1}{8}$
	C	$w_{DC} = \frac{5}{1+1+5+1} = \frac{5}{8}$
	E	$w_{DE} = \frac{1}{1+1+5+1} = \frac{1}{8}$
E	C	$w_{EC} = \frac{5}{5+3} = \frac{5}{8}$
	D	$W_{ED} = \frac{3}{5+3} = \frac{3}{8}$

Table 12: Relations Between Regions

The calculated weights are presented in the transition matrix P.

$$P = \begin{bmatrix} 0.00 & 0.25 & 0.00 & 0.75 & 0.00 \\ 0.11 & 0.00 & 0.55 & 0.33 & 0.00 \\ 0.00 & 0.20 & 0.00 & 0.60 & 0.20 \\ 0.12 & 0.12 & 0.62 & 0.00 & 0.12 \\ 0.00 & 0.00 & 0.62 & 0.37 & 0.00 \end{bmatrix}$$

As it is shown below, change can be defined as input minus output.

$$Change = Input \ - \ Output$$

After defining the change, the inflow and outflow must be specified. The flow between regions i and j, as well as the inflow and outflow, are defined as follows:

$$Flow_{i \to j}(t) = k \times P_{ij} \times S_i(t)$$

$$Output_j(t) = k \times S_j(t)$$

$$Input_j(t) = \sum_i k \times P_{ij} \times S_i(t)$$

After defining the inflow and outflow, the following differential equation is obtained.

$$\frac{dS_j}{dt} = (k \times \sum_i P_{ij} \times S_i(t)) - (k \times S_j(t))$$

In our model, inter-regional movement is treated as a jump (repositioning) process with rate parameter k. Therefore, the time-dependent change in the number of scooters can be expressed as follows. These equations can also be written in vector form as:

$$\frac{dS}{dt} = k \times (P^T - I) \times S(t)$$

The closed-form solution of the equation is:

$$S(1) = e^{k \times (P^T - I)} \times S(0)$$

$$S(1) = e^{-k} \sum_{n=0}^{\infty} \frac{k^n}{n!} \times (P^T)^n \times S(0)$$

$$Pr(N \geq 1) = 0.35 \rightarrow Pr(N = 0) = e^{-k} = 0.65$$

The final form of the formula is:

$$S_{End\ of\ the\ Day} = 0.65 \times S_0 + 0.35 \times P^T \times S_0$$

To calculate the end-of-day values, the transpose of matrix P is taken.

$$P^T = \begin{bmatrix} 0.00 & 0.11 & 0.00 & 0.12 & 0.00 \\ 0.25 & 0.00 & 0.20 & 0.12 & 0.00 \\ 0.00 & 0.55 & 0.00 & 0.62 & 0.62 \\ 0.75 & 0.60 & 0.60 & 0.00 & 0.37 \\ 0.00 & 0.00 & 0.20 & 0.12 & 0 \end{bmatrix}$$

To obtain the end-of-day values, the following calculations are performed and then substituted into the function.

$$I = P^T \times S_0$$

$$I_A = \frac{1}{9} \times S_{B0} + \frac{1}{8} \times S_{D0} = \frac{92}{9} + \frac{85}{8} \approx 20.8472$$

$$I_B = \frac{1}{4} \times S_{A0} + \frac{1}{5} \times S_{C0} + \frac{1}{8} \times S_{D0} = \frac{125}{4} + \frac{110}{5} + \frac{85}{8} = 63.875$$

$$I_C = \frac{5}{9} \times S_{B0} + \frac{5}{8} \times S_{D0} + \frac{5}{8} \times S_{E0} = \frac{460}{9} + \frac{425}{8} + 55 \approx 159.2361$$

$$I_D = \frac{3}{4} \times S_{A0} + \frac{1}{3} \times S_{B0} + \frac{3}{5} \times S_{C0} + \frac{3}{8} \times S_{E0} = \frac{375}{4} + \frac{92}{3} + 66 + 33 \approx 223.4167$$

$$I_E = \frac{1}{5} \times S_{C0} + \frac{1}{8} \times S_{D0} = \frac{110}{5} + \frac{85}{8} = 32.625$$

$$Input\ Vector = P^T \times S_0 \approx \begin{bmatrix} 20.84 \\ 63.87 \\ 159.2 \\ 223.4 \\ 32.62 \end{bmatrix}$$

$$S_{End\ of\ the\ Day} = 0.65 \times S_0 + 0.35 \times P^T \times S_0$$

$$S_A^{End} = (0.65 \times 125) + (0.35 \times \frac{1501}{72}) = 88.5465$$

$$S_B^{End} = (0.65 \times 92) + \left(0.35 \times \frac{511}{8}\right) = 82.15625$$

$$S_C^{End} = (0.65 \times 110) + (0.35 \times \frac{11465}{72}) = 127.2326$$

$$S_D^{End} = (0.65 \times 85) + \left(0.35 \times \frac{2681}{12}\right) = 133.4458$$

$$S_E^{End} = (0.65 \times 88) + \left(0.35 + \frac{261}{8}\right) = 68.61875$$

Thus, the end-of-day values are obtained as follows, and the daily change in the number of scooters in each region is presented in Table 13.

$$S_{End} = (89, 82, 127, 133, 69)$$

Area	Start-of-Day Numbers	End-of-Day Numbers	Daily Change
A	125	89	-36
B	92	82	-10
C	110	127	+17
D	85	133	+48
E	88	69	-19

Table 13: Daily Change of Scooter Numbers in each Area

2.3. USING MONTE CARLO SIMULATION FOR COST ANALYSIS

2.3.1. DAILY COST ANALYSIS

To analyze the daily cost of maintaining the structures defined in the previous sections, the Monte Carlo method is used. (Harrison, 2010) The coefficients used in the method, which were specified earlier, are presented in Table 14.

Cost Type	Cost (per unit)
Scooter Transfer	12
Scooter Charge	8

Table 14: Scooter Maintenance Costs

According to the previous models, the inputs are listed in Table 15.

Target Distribution (Morning)		Neighboring Relations	
A	125	**A**	B, D
B	92	**B**	A, C, D
C	110	**C**	B, D, E
D	85	**D**	A, B, C, E
E	88	**E**	C, D

Table 15: Inputs Found in Previously Models

Therefore, the daily cost function is defined as follows.

$$Daily\ Cost = 12M + 8C$$

$$M = Daily\ amount\ of\ scooters\ transfered$$

$$N = Number\ of\ scooters\ charged$$

To obtain M, the minimum number of scooters that must be transferred back to their original locations is determined using the following formula.

$$M = \frac{1}{2} \sum_{i \in \{A,B,C,D,E\}} \left| x_i^{end} - x_i^{target} \right|$$

In the formula, moving one scooter from a surplus zone to a deficit zone reduces both the surplus and the deficit by 1. Therefore, if M denotes the total number of transfers required, the total absolute imbalance equals 2M.

$$\left| x_a^{end} - x_a^{target} \right| = |91 - 125| = 34$$

$$\left| x_b^{end} - x_b^{target} \right| = |93 - 92| = 1$$

$$\left| x_c^{end} - x_c^{target} \right| = |128 - 110| = 18$$

$$\left| x_d^{end} - x_d^{target} \right| = |115 - 85| = 30$$

$$\left| x_e^{end} - x_e^{target} \right| = |73 - 88| = 15$$

$$M = \frac{1}{2}(34 + 1 + 18 + 30 + 15\ )$$

$$M = 49$$

Thus, the number of scooters that need to be transferred each day is found to be 49.

2.3.2. DAILY COST DISTRIBUTION

The number of scooters transported is assumed to be constant in the daily distribution model.

$$49 \times 12 = 588\,TL$$

A Monte Carlo analysis is used to determine which scooters need charging.

$$D = v \times \frac{T}{60}$$

D: distance

T: $average\ daily\ driving\ time$

To calculate the distance traveled by the scooters, the equation above is used. It is assumed that every scooter travels at the same speed, $v = 10ms^{-1}$.

$$R_{eff} = (1 - \alpha) \times R$$

α: charge threshold

R: $range\ in\ km$

$$R_{eff} = (1 - 0.2) \times 25 = 20\ km$$

Scooters are charged when their battery percentage falls below 20%. The calculations above represent the maximum distance a scooter can travel before needing to be charged.

$$q = \min\left(1, \frac{\mathrm{E}(D)}{R_{eff}}\right)$$

$E(D)$: $travelled\ kilometers\ by\ a\ scooter$

R_{eff}: $total\ km\ avaible\ per\ charge\ cycle$

The ratio of the distance a scooter travels in a day to its usable range determines its likelihood of needing charging. The probability is calculated using a function to ensure it does not exceed 1. For each scooter, a random value is assigned.

$$X_i = \begin{cases} 1, & if\ the\ scooter\ i\ needs\ charging \\ 0, & otherwise \end{cases}$$

$$i = 1,2, \dots \dots, N$$

$$\mathrm{P}(X_i = 1) = q$$

$$\mathrm{P}(X_i = 0) = 1 - q$$

It is assumed that every scooter has the same probability of needing a charge. Here, q represents the scooters that require charging, and $1 -$ q represents the scooters that do not require charging.

$$M = \sum_{i=1}^{N} X_i$$

The number of scooters that need charging is the sum of the 0–1 indicator variable across all scooters.

$$P(M = m) = \binom{N}{m} q^m (1-q)^{N-m}$$

$$m\text{: } number\ of\ scooters\ X_i = 1$$

$$N\text{: } number\ of\ scooter\ (500)$$

$$N - m\text{: } number\ of\ scooters\ X_i = 0$$

$$q = 0.6$$

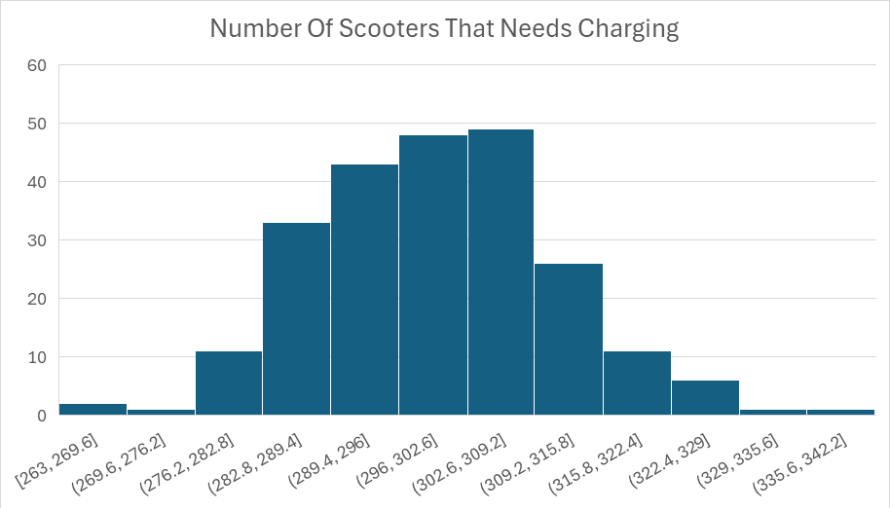
$$m = 0,1,2 \dots, N$$

A binomial distribution is used to calculate the probability that exactly m scooters need charging. Here, m represents a specific number of scooters that require charging. Under the assumption of independence, the probability that a particular scooter does not need charging is $1 - q$, and the probability that it does need charging is q. Therefore, the probability that a specific set of m scooters needs charging while the remaining $N - m$ scooters do not is $q^m(1-q)^{N-m}$. However, the statement "m scooters need charging" does not refer to a fixed group of scooters, so the number of possible groups is given by the binomial coefficient $\binom{N}{m}$. Since each group has the same probability, the total probability is obtained by multiplying $\binom{N}{m}$ by $q^m(1-q)^{N-m}$.

And therefore:

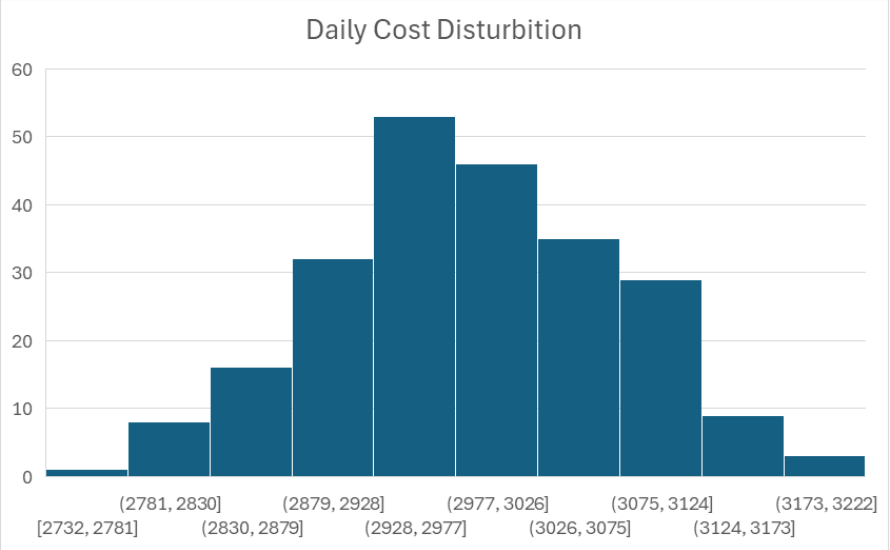
$$M \sim Binomial(N,\ q)$$

The Monte Carlo calculations are performed in Excel. A random value is generated for each scooter, and the simulation is iterated to produce the histogram.



Graph 1: Number Of Scooters That Needs Charging

The peak of the distribution occurs between 296 and 309.2, indicating that nearly 300 scooters need to be recharged. The distribution is not widely spread; therefore, the number of scooters that need charging each day does not vary greatly.



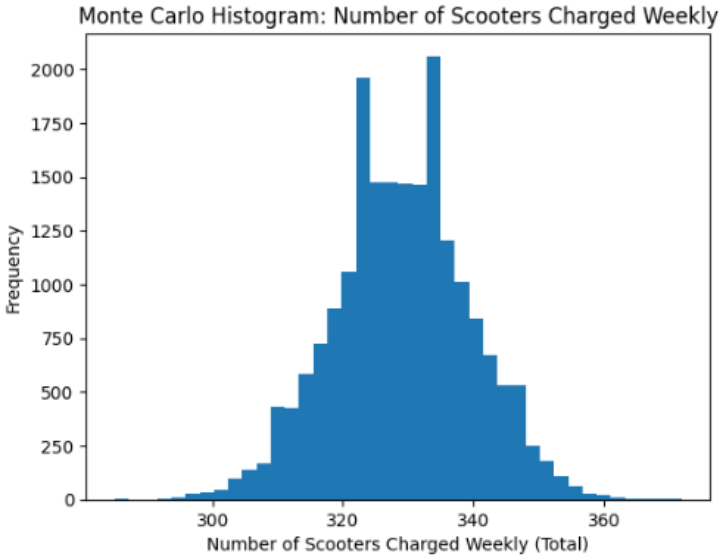
Graph 2: Daily Cost Disturbition

Compared with the Monte Carlo analysis for scooter charging, the daily cost analysis shows taller bars, indicating that certain cost ranges occur more frequently and that the cost is less stable. The daily cost is approximately 2928–2977 TL.

2.3.3. WEEKLY COST ANALYSIS

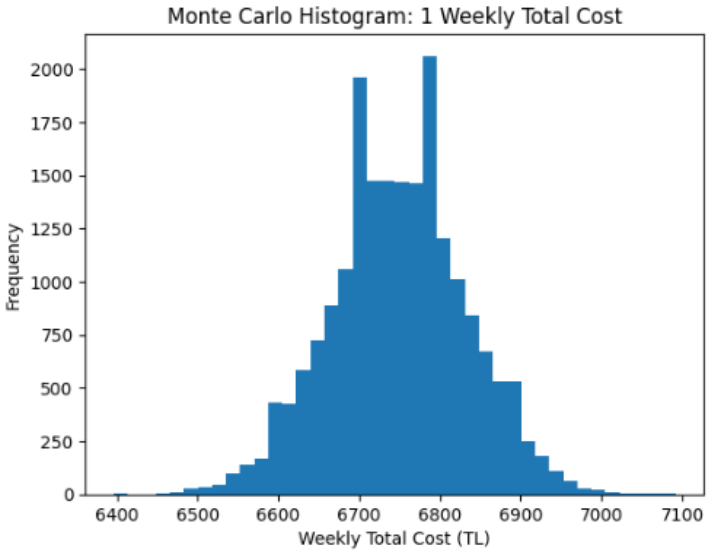
The weekly cost analysis is calculated using Python. The code runs a Monte Carlo simulation with 20,000 iterations. It is based on the assumptions stated in the task: each scooter is used six times per day, while the duration of each trip is random. The line `trip_minutes = rng.gamma(shape=trip_shape, scale=trip_scale, size=(N, K))` generates random trip durations. Here, N and K denote the total number of scooters

(`N = 500`) and number of rides per scooter (`K = 6`) respectively. The line `needs_charge = (B < threshold)` applies a condition to identify scooters that require charging. The full code is provided in the Appendix.



Graph 3: Weekly Charged Scooters

The number of scooters clusters around the average. The total number of scooters that require charging is approximately between 290 and 370.



Graph 4: 1 Week Total Cost

The total weekly cost of scooter transportation and battery charging is approximately 6,800 TL. The two graphs are similar because the transportation cost is assumed to be stable. Since the conditions do not change over the week, the transportation cost is calculated as shown below:

$$7 \times 588 = 4116\,TL$$

This also indicates a linear relationship between the number of scooters that need charging and the total cost.

2.4. Optimization Problem

To optimize the model using linear programming and simulation are applied. The calculations are performed in Python. The notation used for the optimization is given below:

$$I = \{A, B, C, D, E\} \quad |I| = 5$$
$$i\text{: calculated, target area}$$
$$j\text{: } other\ area$$
$$E = \{(i,j) : j \in Neighbors\ (i)\}$$

Set E represents the neighborhood relationships between the areas. This set is based on the neighborhood assumptions made in the end-of-day imbalance model.

2.4.1 Normalization

The P^T, transition matrix is normalized to ensure the total probability of every column is equal to 1. `col_sums = PT_raw.sum(axis=0) PT = PT_raw / col_sums` has the function to normalize the matrix.

$$P_{ij}^T = \frac{(P^T)_{ij}}{\sum_{k \in I}(P^T)_{kj}}$$

The matrix is normalized to ensure model consistency, where normalization ensures that the total number of scooters is preserved.

$$e = P^T x$$

Here, e represents the end of the day distribution in the code.

$$\sum_i e_i = \sum_i \sum_j P_{ij}^T x_j = \sum_j x_j \sum_i P_{ij}^T = \sum_j x_j = N = 500$$

The given equation is true if only the sum of every column is 1.

2.4.2 Linear Programming (LP)

Linear programming is a method used to find the best outcome under a set of constraints. The LP model is defined using the following decision variables:

Beginning of the day distribution	$x_i \geq 0\ (i \in I)$
The day after night transport	$y_i \geq 0\ (i \in I)$
Number of transported scooters at night	$m_{ij} \geq 0\ \big((i,j) \in E\big)$

Table 16: Decision Variables in LP system

In the LP model, e (the end-of-day distribution) is not defined as a separate decision variable; instead, it is computed as:

$$e_i = \sum_{j \in I} P_{ij}^T x_j \qquad i \in I$$

This equation represents the redistribution that occurs during the day. The overnight update is then written as:

$$y_i = e_i + \sum_{j:(j,i)\in E} m_{ji} - \sum_{j:(i,j)\in E} m_{ij} \qquad i \in I$$

This equation is a flow balance constraint. y_i denotes the final number of scooters in region i after overnight transportation. The term $\sum m_{ji}$ represents scooters transported into region i from other regions, while $\sum m_{ij}$ represents scooters transported out of region i to other regions. Here, e_i is the number of scooters in region i at the end of the day (before overnight transport).

$$e_i \geq s_i^{req} \quad (i \in I)$$

s_i^{req}: number of scooters that need charging.

The inequality above ensures that enough scooters are available to meet the charging requirement. The values s_i^{req} are obtained from the daily Monte Carlo analysis. The mean value across iterations is used to estimate the average number of scooters that require charging.

2.4.3 Cost Function:

$$\boldsymbol{C} = \boldsymbol{C_{move}} + \boldsymbol{C_{charge}}$$

$$C_{move} = 12 \times \sum_{(i,j) \in E} m_{ij}$$

$$C_{charge} = 8 \times \sum_{(i,j) \in E} m_{ij}$$

The movement cost is optimized because it is higher than the charging cost. Moreover, scooter movement is a decision variable in the LP (Linear Programming) model, so it must be optimized. The number of scooters that need charging is estimated using proportional allocation. An example calculation for the distribution of scooters requiring charging is shown for Area A.

The average number of scooters that need charging: 300.612

$$s_A^{req} = \frac{125 \times 300.108}{500} = 75.14$$

The initial number of scooters is used in the proportioning process because these initial scooter numbers also represent the demand scores that were calculated.

Area	Scooters Need Charging
A	75.14

B	55.30
C	66.12
D	51.09
E	52.89

Table 17: Scooters that Need Charging per Area

2.4.4 Aim Function- Trips:

$$T = \sum_{i \in I} t_i \, y_i$$

The objective function is defined in terms of y_i and t_i. If the next day's initial distribution is favorable, the expected total number of trips increases. The value t_i is calculated by multiplying 6 by the initial scooter distribution, since each scooter makes an average of six trips per day.

Area	t_i
A	750
B	552
C	660
D	510
E	528

Table 18: Number of Trips per Area

2.4.5 Pareto Chart

The code generates a Pareto grid by solving the LP (Linear Programming) model separately for different total cost values.

$$max\ T \quad s.t.\ C \leq \varepsilon$$

$$C_{move} + C_{charge} \leq \varepsilon$$

$$C_{move} \leq \varepsilon - C_{charge}$$

So the real constraint added to the LP (Linear Programming)

$$12 \times \sum_{(i,j) \in E} m_{ij} \leq \varepsilon - C_{charge}$$

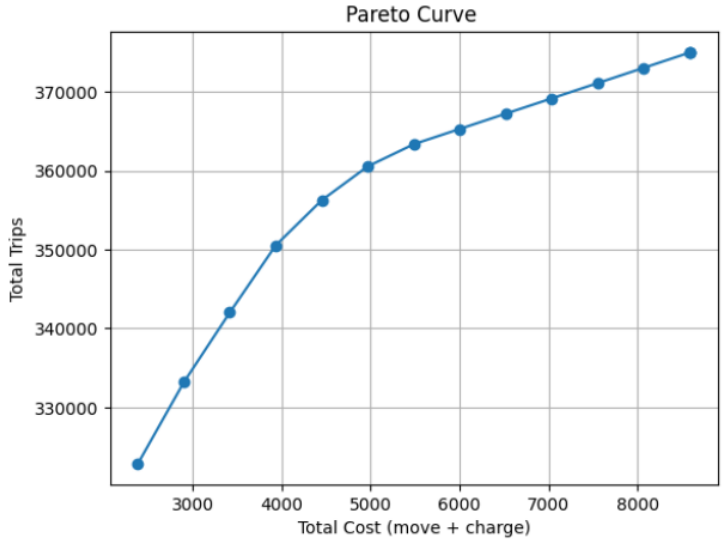
This formula gives different ε to plot the pareto curve.

The Pareto curve is generated from different values of ε . Each point on the curve is given by $\big(C(\varepsilon),\ T(\varepsilon)\big)$.

C(ε)	T(ε)
2387.24	322842.15
2903.61	333169.50
3419.98	342012.88
3936.34	350532.94

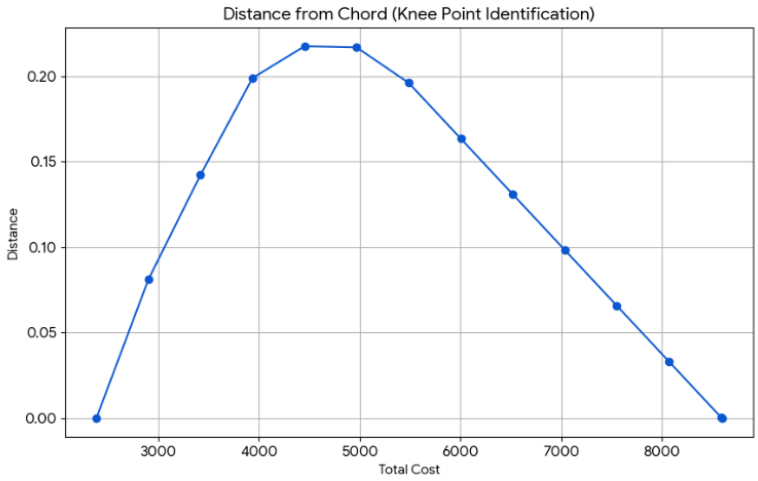
C(ε)	T(ε)
4452.71	356243.15
4969.08	360533.63
5485.45	363344.55
6001.81	365280.93
6518.18	367217.31
7034.55	369153.69
7550.91	371090.06
8067.28	373026.44
8583.65	374962.82
8593.56	375000.00
8593.56	375000.00

Table 19: Code Outputs, Pareto points



Graph 5: Pareto Curve

Trips and cost are positively correlated: as cost increases, T(ε) also increases. This implies that more scooters are moved overnight, allowing the model to shift scooters toward higher-demand (higher-trip) regions. At lower cost levels, the system is more efficient because the slope is steeper—meaning more trips are gained per unit cost. However, once the cost exceeds about 6000 TL, the slope begins to decline, indicating diminishing returns. After roughly 5000 TL, the efficiency of the system starts to decrease. The optimal point on this curve is referred to as the knee point.



Graph 6: The Knee Point of the Pareto Curve

The knee point is determined using Python code based on derivatives. The best value-for-money point occurs at 4452.71 TL, and the model provides a balanced performance at 4969.08 TL. At 4452.71 TL, the model achieves 356243 trips; this point has the maximum curvature. At 4969.08 TL, 360,533 trips can be made, and beyond this point the curve begins to flatten. The maximum capacity of the model is reached at 8593.56 TL, corresponding to 37000 trips.

2.5. WHAT-IF SCENARIO ANALYSIS

2.5.1. FESTIVAL IN REGION C SITUATION

2.5.1.1. MODELLING THE SITUATION WITH NEW VALUES

To analyse the festival scenario, the differential equation model (Model 2) can still be used; however, the scores and inter-regional relationships must be updated because Region C's score increases by 70%. The following equation shows the updated score for Region C and the revised scores for all regions.

$$T_C = 5 \rightarrow T_C' = 5 + \frac{5 \times 70}{100} = 8.5$$

$$(T_A, T_B, T_C, T_D, T_E) = (1, 1, 8.5, 3, 1)$$

Based on the previously defined neighborhood assumptions, the weights between neighboring regions are recalculated using the following equation and presented in Table 20.

$$P_{ij} = \begin{cases} \dfrac{w_{ij}'}{\sum_{k \in N(i)} w_{ik}'} & if\ j \in N(i) \\ 0 & otherwise \end{cases}$$

$$\sum_j P_{ij}' = 1$$

A	B	$w_{AB} = \frac{1}{1+3} = \frac{1}{4}$
	D	$w_{AD} = \frac{3}{1+3} = \frac{3}{4}$
B	A	$w_{BA} = \frac{1}{1+8.5+3} = 0.08$
	C	$w_{BC} = \frac{8.5}{1+8.5+3} = 0.68$
	D	$w_{BD} = \frac{3}{1+8.5+3} = 0.24$
C	B	$w_{CB} = \frac{1}{1+3+1} = \frac{1}{5}$
	D	$w_{CD} = \frac{3}{1+3+1} = \frac{3}{5}$
	E	$w_{CE} = \frac{1}{1+3+1} = \frac{1}{5}$
D	A	$w_{DA} = \frac{1}{1+1+8.5+1} = 0.08$
	B	$w_{DB} = \frac{1}{1+1+8.5+1} = 0.08$
	C	$w_{DC} = \frac{8.5}{1+1+8.5+1} = \frac{5}{8}$
	E	$w_{DE} = \frac{1}{1+1+8.5+1} = 0.08$
E	C	$w_{EC} = \frac{8.5}{8.5+3} = 0.73$
	D	$W_{ED} = \frac{8.5}{8.5+3} = 0.26$

Table 20: Relations Between Regions

The updated relationships between regions are shown in matrix P'.

$$P' = \begin{bmatrix} 0.0000000 & 0.2500000 & 0.0000000 & 0.7500000 & 0.0000000 \\ 0.0800000 & 0.0000000 & 0.6800000 & 0.2400000 & 0.0000000 \\ 0.0000000 & 0.2000000 & 0.0000000 & 0.6000000 & 0.2000000 \\ 0.0869565 & 0.0869565 & 0.7391304 & 0.0000000 & 0.0869565 \\ 0.0000000 & 0.0000000 & 0.7391304 & 0.2608696 & 0.0000000 \end{bmatrix}$$

After calculating the updated weights for the given scenario, the same equation used in Model 2 is applied to obtain the new distribution.

$$S'_{End\ of\ the\ Day} = 0.65 \times S_0 + 0.35 \times P^{T\prime} \times S_0$$

$$S_0 = (125,92,110,85,88)$$

$$S'_{End\ of\ the\ Day} \approx (86,81,138,127,68)$$

2.5.1.2. ANALYSING CHANGE IN MODEL

To analyse the changes in the model under the Festival Scenario, a table is created summarizing all calculated values.

Area	Start-of-Day Numbers	First End-of-Day Numbers	Festival Situation End-of-the-Day Numbers
A	125	89	86
B	92	82	81
C	110	127	138
D	85	133	127
E	88	69	68

Table 20: Values Founded in Models

To determine the change in the model, the following equations are used:

$$\Delta S = S^{festival} - S^{Base\ Model}$$

$$\Delta S_A = 86.4130 - 88.5465 = -2.1335$$

$$\Delta S_B = 81.0240 - 82.1563 = -1.1323$$

$$\Delta S_C = 138.1500 - 133.4458 = 10.9174$$

$$\Delta S_D = 126.9250 - 133.4458 = -6.5208$$

$$\Delta S_E = 67.4870 - 68.6188 = -1.1318$$

$$\Delta S \approx (-2.134, -1.132, +10.918, -6.521, -1.132)$$

The following equations are used to determine the percentage change:

$$r_i = \frac{S_i^{Festival} - S_i^{Base\ Model}}{S_i^{Base\ Model}} \times 100\%$$

$$r_A = \frac{-2.1335}{88.5465} \approx -2.41\%$$

$$r_B = \frac{-1.1323}{82.1563} \approx -1.38\%$$

$$r_C = \frac{10.9174}{127.2326} \approx 8.58\%$$

$$r_D = \frac{-6.5208}{133.4458} \approx -4.89\%$$

$$r_E = \frac{-1.1318}{68.6188} \approx -1.65\%$$

2.5.2 The Strike in Region B

As in the previous scenario, the existing model can still be used, but the scores must be updated. In this case, the change occurs in Region B's score, since an extraordinary increase in demand is assumed and the region's total demand is doubled.

2.5.2.1 Deriving the Target Distribution

To compare the current distribution with the distribution required to mitigate the impact of the strike, a new scooter distribution must be computed by adjusting the coefficients. Here, the same y values shown in Table 11 can be used, except that the y value for Region B is doubled. Therefore, the revised vector is obtained as follows:

$$x = (125,92,110,85,88), \sum x_i = 500$$

Here, the matrix from the standard scenario is provided. To model a doubling of demand in Region B, the y value for Region B can be doubled and then the totals can be rescaled so that the overall sum returns to 500.

$$x' = (125,184,110,85,88), \sum x_i' = 592$$

$$a = \frac{500}{\sum x_i'} = \frac{500}{592}$$

$$S_{Target} = ax \quad \approx (106,155,93,72,74)$$

Region	Target y value
A	106
B	155
C	93
D	72
E	74

Table 21: Target y Values

2.5.2.2 Modelling the Current Situation

Based on the previously defined neighborhood assumptions, the weights between neighboring regions are recalculated using the following equation and presented in Table 12, with the only difference being that the tourist rate of region B has been doubled. In this process, the same formula can be used.

$$P_{ij} = \begin{cases} \frac{w_{ij}'}{\sum_{k \in N(i)} w_{ik}'} & if\ j \in N(i) \\ 0 & otherwise \end{cases}$$

$$\sum_j P_{ij}' = 1$$

Using this formula, the final matrix P is found as such:

$$P' = \begin{bmatrix} 0.0000000 & 0.4000000 & 0.0000000 & 0.6000000 & 0.0000000 \\ 0.1111111 & 0.0000000 & 0.5555555 & 0.3333333 & 0.0000000 \\ 0.0000000 & 0.3333333 & 0.0000000 & 0.5000000 & 0.1666666 \\ 0.1111111 & 0.2222222 & 0.5555555 & 0.0000000 & 0.1111111 \\ 0.0000000 & 0.0000000 & 0.6250000 & 0.3750000 & 0.0000000 \end{bmatrix}$$

After calculating the updated weights for the given scenario, the same equation used in Model 2 is applied to obtain the new distribution.

$$S'_{End\ of\ the\ Day} = 0.65 \times S_0 + 0.35 \times P^{T'} \times S_0$$

$$S_0 = (106,155,93,72,74)$$

$$S'_{End\ of\ the\ Day} \approx (77,132,121,113,57)$$

Therefore, the new end-of-day values can be found as such:

Region	Start-of-Day Values	End-of Day Values
A	106	77
B	155	132
C	93	121
D	72	113
E	74	57

Table 22: Start and End Values

2.5.2.3 Comparing the Target Distribution and Current Situation

To determine the change in the model, the following equations are used:

$$\Delta S = S^{festival} - S^{Base\ Model}$$

$$\Delta S_A = 88.1333 - 88.5465 = -0.4132$$

$$\Delta S_B = 96.7444 - 82.1563 = 14.5881$$

$$\Delta S_C = 125.1667 - 133.4458 = -2.0659$$

$$\Delta S_D = 123.033 - 133.4458 = -10.4125$$

$$\Delta S_E = 66.9222 - 68.6188 = -1.6966$$

$$\Delta S \approx (-0.4132, +14.5881, -2.0659, -10.4125, -1.6966)$$

The following equations are used to determine the percentage change:

$$r_i = \frac{S_i^{Festival} - S_i^{Base\ Model}}{S_i^{Base\ Model}} \times 100\%$$

$$r_A = \frac{-0.4132}{88.5465} \approx -0.47\%$$

$$r_B = \frac{14.5881}{82.1563} \approx -17.76\%$$

$$r_C = \frac{-2.0659}{127.2326} \approx -1.62\%$$

$$r_D = \frac{-10.4125}{133.4458} \approx -7.80\%$$

$$r_E = \frac{-1.6966}{68.6188} \approx -2.47\%$$

3. CONCLUSION

This study aimed to present a three-layer modelling framework to plan and operate a fleet of 500 electric scooters across five urban regions by combining (i) demand-based allocation, (ii) end-of-day imbalance dynamics, and (iii) cost-aware optimization. The approach begins by converting qualitative factor levels into numerical scores and then applying AHP to obtain consistent factor weights. Using these weights in a demand function produced an initial (morning) distribution of scooters across the five areas as A: 125, B: 92, C: 110, D: 85, E: 88 (where the total is 500), representing proportional demand-based allocations.

To capture daily drift and imbalance, the study modelled inter-regional movement with a differential-equation / jump-process formulation constrained by neighbourhood relations. The resulting end-of-day distribution shows systematic shifts toward high-attraction regions, yielding A: 89, B: 82, C: 127, D: 133, E: 69, and quantifying daily surpluses/deficits that motivate repositioning. Based on the imbalance measure, the minimum overnight transfers required to restore the target distribution was found as 49 scooters per day.

Operational sustainability was evaluated using Monte Carlo simulation for charging needs and total maintenance cost. Under the stated assumptions (usage patterns, battery threshold, constant movement volume), the charging demand concentrates around an average near 300 scooters/day (approximately 290–370 in the weekly simulation), indicating limited day-to-day variability. The daily total cost distribution falls roughly in the 2928–2977 TL range, while weekly total cost is approximately 6800 TL, and the results imply an approximately linear relationship between the number of scooters requiring charge and total cost (movement cost treated as stable).

Finally, a linear programming (LP) model was introduced to optimize overnight scooter movements—prioritizing movement cost because it dominates charging cost—while ensuring feasibility constraints (including charging requirements) and maximizing expected trips. Solving the LP repeatedly across different cost caps generated a Pareto curve (cost vs. trips). The curve exhibits diminishing returns beyond mid-range budgets: the knee point (maximum curvature) occurs at approximately 4452.71 TL, achieving about 356243 trips, while a balanced operating point appears near 4969.08 TL with 360,533 trips; the maximum-output region approaches 8593.56 TL and 375000 trips, where the curve flattens and additional spending yields marginal gains.

Scenario analyses demonstrated that the framework remains usable under disruptions by updating scores and/or target allocations. In the festival scenario (Region C +70% score), the model predicts a stronger pull toward Region C at day's end, while in the strike scenario (Region B demand doubled), the target distribution is recomputed by rescaling adjusted demand scores back to a 500-scooter total. These results show the proposed system can be recalibrated quickly for atypical demand shocks without rebuilding the model structure.

Overall, the combined AHP–dynamics–Monte Carlo–LP pipeline provides a coherent decision-support tool: it allocates scooters proportional to multi-factor demand, predicts daily imbalance, estimates charging and cost uncertainty, and selects cost-effective repositioning plans via Pareto/knee-point analysis. Potential improvements include incorporating day-to-day variability in demand intensity, heterogeneous scooter speeds/usage, time-dependent transition matrices, and richer neighbourhood graphs (e.g., travel time or capacity constraints), which would further increase realism and operational robustness.

REFERENCES

Blanchet, T., & Piketty, T. (2021). GENERALIZED PARETO CURVES: THEORY AND APPLICATIONS. *Review of Income and Wealth*.

Harrison, L. (2010). Introduction To Monte Carlo Simulation. *National Library of Medicine*.

Science Direct. (2026). *Consistency Index.* A quantitative analysis of model predictive control as energy management strategy for hybrid electric vehicles: A review: https://www.sciencedirect.com/topics/engineering/consistency-index#:~:text=The%20consistency%20ratio%2C%20the%20ratio,as%20having%20an%20acceptable%20consistency. adresinden alındı

Vanderbei, R. (2020). Linear Programming. International Series in Management Science/Operations Research.

APPENDIX

APPENDIX 1: MONTE CARLO WEEKLY COST ANALYSIS PYTHON CODE

```
import numpy as np
import matplotlib.pyplot as plt

N = 500
R = 25.0
v = 10.0
threshold = 0.2 * R
move_cost_day = 588.0
move_cost_week = 7 * move_cost_day
charge_cost_per = 8.0
S = 20000
rng = np.random.default_rng(42)
K = 6
trip_shape = 2.0
trip_scale = 1.5
def simulate_one_week():
    B = np.full(N, R, dtype=float)
    total_charges = 0
    for d in range(days):
        trip_minutes = rng.gamma(shape=trip_shape, scale=trip_scale,
size=(N, K))
        T = trip_minutes.sum(axis=1)
        D = v * (T / 60.0)
        B = B - D
        needs_charge = (B < threshold)
        M_d = int(needs_charge.sum())
        total_charges += M_d
        B[needs_charge] = R
        B = np.maximum(B, 0.0)
    weekly_cost = move_cost_week + charge_cost_per * total_charges
    return weekly_cost, total_charges
weekly_costs = np.empty(S)
weekly_charges = np.empty(S, dtype=int)
for s in range(S):
    c, m = simulate_one_week()
    weekly_costs[s] = c
    weekly_charges[s] = m

mean_cost = weekly_costs.mean()
p5, p95 = np.percentile(weekly_costs, [5, 95])
```

```
mean_M = weekly_charges.mean()
p5_M, p95_M = np.percentile(weekly_charges, [5, 95])
print(f"Weekly transportation cost (fixed): {move_cost_week:.0f} TL/week")
print(f"Average weekly charge for scooter (total): {mean_M:.2f}
(P5={p5_M:.0f}, P95={p95_M:.0f})")
print(f"Average weekly total cost: {mean_cost:.2f} TL  (P5={p5:.2f},
P95={p95:.2f})")
plt.figure()
plt.hist(weekly_costs, bins=40)
plt.xlabel("Weekly Total Cost (TL)")
plt.ylabel("Frequency")
plt.title("Monte Carlo Histogram: 1 Weekly Total Cost")
plt.show()
plt.figure()
plt.hist(weekly_charges, bins=40)
plt.xlabel("Number of Scooters Charged Weekly (Total)")
plt.ylabel("Frequency")
plt.title("Monte Carlo Histogram: Number of Scooters Charged Weekly")
plt.show()
```

APPENDIX 2: PYTHON CODE FOR PARETO ANALYSIS

```
import numpy as np
from scipy.optimize import linprog
import matplotlib.pyplot as plt

zones = ["A","B","C","D","E"]
n = len(zones)
zi = {z:i for i,z in enumerate(zones)}

PT_raw = np.array([
    [0.00, 0.25, 0.00, 0.75, 0.00],
    [0.11, 0.00, 0.55, 0.60, 0.00],
    [0.00, 0.20, 0.00, 0.60, 0.20],
    [0.12, 0.12, 0.62, 0.00, 0.12],
    [0.00, 0.00, 0.62, 0.37, 0.00],], dtype=float)

col_sums = PT_raw.sum(axis=0)
PT = PT_raw / col_sums
print("Column sums (should be 1):", PT.sum(axis=0))
N_total = 500

t = np.array([750.0, 552.0, 660.0, 510.0, 528.0], dtype=float)
s_req = np.array([74.6012931, 54.90655172, 65.64913793, 50.72887931,
52.51931034], dtype=float)
cost_move = 12.0
```

```
cost_charge = 8.0
charge_cost_const = cost_charge * s_req.sum()

neighbors = {
    "A": ["B","D"],
    "B": ["A","C","D"],
    "C": ["B","D","E"],
    "D": ["A","B","C","E"],
    "E": ["C","D"]}

arcs = []
for i in zones:
    for j in neighbors[i]:
        arcs.append((i,j))
m = len(arcs)

print("Directed arcs:", m)

def idx_x(i): return zi[i]
def idx_y(i): return n + zi[i]
def idx_m(k): return 2*n + k

num_vars = 2*n + m

def solve_lp(eps=None, objective="trips"):
    """
    objective:
      - "trips": maximize trips (linprog minimize -> minimize -trips)
      - "cost" : minimize total cost (move + constant charge)
    eps:
      - None: no upper bound on cost
      - number: total cost <= eps (epsilon constraint)
    """
    c = np.zeros(num_vars)

    if objective == "trips":
        for i,z in enumerate(zones):
            c[idx_y(z)] = -t[i]

    elif objective == "cost":
        for k in range(m):
            c[idx_m(k)] = cost_move
    else:
        raise ValueError("objective must be 'trips' or 'cost'")

    A_eq, b_eq = [], []
```

```
    A_ub, b_ub = [], []


    row = np.zeros(num_vars)
    for z in zones:
        row[idx_x(z)] = 1.0
    A_eq.append(row)
    b_eq.append(N_total)

    for i,z in enumerate(zones):
        row = np.zeros(num_vars)
        row[idx_y(z)] = 1.0

        for k,(a,b) in enumerate(arcs):
            if b == z:
                row[idx_m(k)] -= 1.0
            if a == z:
                row[idx_m(k)] += 1.0

        for j,zz in enumerate(zones):
            row[idx_x(zz)] -= PT[i,j]

        A_eq.append(row)
        b_eq.append(0.0)

    for i,z in enumerate(zones):
        row = np.zeros(num_vars)
        for j,zz in enumerate(zones):
            row[idx_x(zz)] = -PT[i,j]
        A_ub.append(row)
        b_ub.append(-s_req[i])

    if eps is not None:
        row = np.zeros(num_vars)
        for k in range(m):
            row[idx_m(k)] = cost_move
        A_ub.append(row)
        b_ub.append(float(eps) - charge_cost_const)

    bounds = [(0, None)] * num_vars
    for z in zones:
        bounds[idx_x(z)] = (0, N_total)

    res = linprog(
        c=c,
        A_ub=np.array(A_ub) if A_ub else None,
```

```
        b_ub=np.array(b_ub) if b_ub else None,
        A_eq=np.array(A_eq),
        b_eq=np.array(b_eq),
        bounds=bounds,
        method="highs" )

    if not res.success:
        return None

    sol = res.x
    x = np.array([sol[idx_x(z)] for z in zones])
    y = np.array([sol[idx_y(z)] for z in zones])
    msol = np.array([sol[idx_m(k)] for k in range(m)])

    e = PT @ x
    move_cost = cost_move * msol.sum()
    total_cost = move_cost + charge_cost_const
    trips = (t * y).sum()

    return {
        "x": x, "e": e, "y": y, "m": msol,
        "move_cost": move_cost,
        "charge_cost": charge_cost_const,
        "cost": total_cost,
        "trips": trips,
        "res": res}

sol_costmin = solve_lp(objective="cost")
if sol_costmin is None:
    raise RuntimeError("Cost minimization infeasible!")

C_min = sol_costmin["cost"]

sol_tripsmax = solve_lp(objective="trips")
if sol_tripsmax is None:
    raise RuntimeError("Trips maximization infeasible!")

C_at_Tmax = sol_tripsmax["cost"]
T_max = sol_tripsmax["trips"]

print("\nC_min =", C_min)
print("C_at_Tmax =", C_at_Tmax, " | T_max =", T_max)



eps_grid = np.linspace(C_min, C_at_Tmax, 15)
pareto = []
```

```
for eps in eps_grid:
    sol = solve_lp(objective="trips", eps=eps)
    if sol is not None:
        pareto.append((sol["cost"], sol["trips"], sol))

pareto = sorted(pareto, key=lambda x: x[0])
costs = [p[0] for p in pareto]
trips = [p[1] for p in pareto]

print("\nPareto points (cost, trips):")
for cst, trp, _ in pareto:
    print(f"{cst:.2f}\t{trp:.2f}")



plt.figure()
plt.plot(costs, trips, marker="o")
plt.xlabel("Total Cost (move + charge)")
plt.ylabel("Total Trips")
plt.title("Pareto Curve ")
plt.grid(True)
plt.show()



mid = pareto[len(pareto)//2][2]
print("\n--- Example (mid-epsilon) solution ---")
print("x (start):", dict(zip(zones, mid["x"])))
print("e (end-of-day):", dict(zip(zones, mid["e"])))
print("y (after night moves):", dict(zip(zones, mid["y"])))
print("Total moved:", mid["m"].sum())
print("Move cost:", mid["move_cost"], "Charge cost (const):",
mid["charge_cost"])
print("Total cost:", mid["cost"], "Trips:", mid["trips"])
```

Python Code For Findind the Knee Point:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = [
    (2387.24, 322842.15),
    (2903.61, 333169.50),
    (3419.98, 342012.88),
    (3936.34, 350532.94),
    (4452.71, 356243.15),
```

```
    (4969.08, 360533.63),
    (5485.45, 363344.55),
    (6001.81, 365280.93),
    (6518.18, 367217.31),
    (7034.55, 369153.69),
    (7550.91, 371090.06),
    (8067.28, 373026.44),
    (8583.65, 374962.82),
    (8593.56, 375000.00)
]

df = pd.DataFrame(data, columns=['cost', 'trips'])

df['trips_per_cost'] = df['trips'] / df['cost']

df['delta_cost'] = df['cost'].diff()
df['delta_trips'] = df['trips'].diff()
df['marginal_gain'] = df['delta_trips'] / df['delta_cost']

cost_norm = (df['cost'] - df['cost'].min()) / (df['cost'].max() -
df['cost'].min())
trips_norm = (df['trips'] - df['trips'].min()) / (df['trips'].max() -
df['trips'].min())

p1 = np.array([cost_norm.iloc[0], trips_norm.iloc[0]])
p2 = np.array([cost_norm.iloc[-1], trips_norm.iloc[-1]])
points = np.column_stack((cost_norm, trips_norm))

def get_distance(p, p1, p2):
    return np.abs(np.cross(p2-p1, p1-p)) / np.linalg.norm(p2-p1)

df['dist_from_chord'] = [get_distance(p, p1, p2) for p in points]


print(df[['cost', 'trips', 'trips_per_cost', 'marginal_gain',
'dist_from_chord']])



plt.figure(figsize=(10, 6))
plt.plot(df['cost'], df['dist_from_chord'], marker='o')
plt.title('Distance from Chord (Knee Point Identification)')
plt.xlabel('Total Cost')
plt.ylabel('Distance')
plt.grid(True)
plt.savefig('knee_detection.png')
```