algorithms

MDPI

Article

Predicting the Aggregate Mobility of a Vehicle Fleet within a City Graph

J. Fernando Sánchez-Rada [1,2,*], Raquel Vila-Rodríguez [2], Jesús Montes [3] and Pedro J. Zufiria [2,4,*]

1 Departamento de Ingeniería de Sistemas Telemáticos, ETSI Telecomunicación, Universidad Politécnica de Madrid, 28006 Madrid, Spain
2 Cátedra Cabify, ETSI Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain; raquel.vilarod@alumnos.upm.es
3 Cabify, 28002 Madrid, Spain; jesus.montes@cabify.com
4 Departamento Matemática Aplicada a las TIC, Information Processing and Telecommunications Center (IPTC), ETSI Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain
* Correspondence: jf.sanchez@upm.es (J.F.S.-R.); pedro.zufiria@upm.es (P.J.Z.)

Abstract: Predicting vehicle mobility is crucial in domains such as ride-hailing, where the balance between offer and demand is paramount. Since city road networks can be easily represented as graphs, recent works have exploited graph neural networks (GNNs) to produce more accurate predictions on real traffic data. However, a better understanding of the characteristics and limitations of this approach is needed. In this work, we compare several GNN aggregated mobility prediction schemes to a selection of other approaches in a very restricted and controlled simulation scenario. The city graph employed represents roads as directed edges and road intersections as nodes. Individual vehicle mobility is modeled as transitions between nodes in the graph. A time series of aggregated mobility is computed by counting vehicles in each node at any given time. Three main approaches are employed to construct the aggregated mobility predictors. First, the behavior of the moving individuals is assumed to follow a Markov chain (MC) model whose transition matrix is inferred via a least squares estimation procedure; the recurrent application of this MC provides the aggregated mobility prediction values. Second, a multilayer perceptron (MLP) is trained so that—given the node occupation at a given time—it can recursively provide predictions for the next values of the time series. Third, we train a GNN (according to the city graph) with the time series data via a supervised learning formulation that computes—through an embedding construction for each node in the graph—the aggregated mobility predictions. Some mobility patterns are simulated in the city to generate different time series for testing purposes. The proposed schemes are comparatively assessed compared to different baseline prediction procedures. The comparison illustrates several limitations of the GNN approaches in the selected scenario and uncovers future lines of investigation.

Keywords: mobility in graphs; Markov chain; graph neural network (GNN)

check for updates

Citation: Sánchez-Rada, J.F.; Vila-Rodríguez, R.; Montes, J.; Zufiria, P.J. Predicting the Aggregate Mobility of a Vehicle Fleet within a City Graph. *Algorithms* **2024**, *17*, 166. https://doi.org/10.3390/a17040166

Academic Editors: Enrico Corradini and Domenico Ursino

Received: 19 March 2024
Revised: 16 April 2024
Accepted: 17 April 2024
Published: 19 April 2024

1. Introduction

Predicting the mobility of vehicles can be of great help to smart city management (e.g., location-based services, sensing, etc.) and smart vehicle applications (e.g., transportation systems, communications, etc.) [1–6].

Depending on the application and domain, the prediction can be done based on the data of each element (disaggregated data) or using aggregated values for a given region or way. Although disaggregated data can potentially lead to more accurate results, personal mobility data may not always be available Storing this type of information long-term or sharing it with third parties may lead to privacy and legal issues. Thus, this work focuses on directly applicable and privacy-preserving models on aggregated data.

In recent years, many real-world datasets have been published, and they have become the norm in GNN-based and broader ML-based traffic forecasting studies [7]. Evaluating with real data eliminates any potential questions about the validity of the data and the applicability of the model in real scenarios. However, traffic simulations have two potential benefits. Firstly, simulations have the ability of modeling unseen graphs, such as a planned road topology [7]. Secondly, evaluating predictive models on synthetic and extreme scenarios can help to better understand them and discover their limitations [8].

Taking into account that asset (vehicle) mobility in a city can be formalized using the graph that defines the streets (edges), intersections (nodes), and directions (associated with edges), in this work we address the prediction of *aggregated* asset distribution along the nodes of this graph, based only on past aggregated distribution history. Aggregated patterns are a result of individual asset behavior, which in real-life scenarios can be very complex. This in turn hinders the ability to reason about the emergent behavior or to construct extreme scenarios that test the limits of a given model. Hence, this paper focuses on a very simple mobility model based on Markov Chains (MC). This model is parameterized by a transition matrix, which can be easily tuned. Then, we propose three prediction schemes: a method based on a Markov Chain (MC) model [9], and two methods based on Multilayer Perceptrons (MLP) [10] and graph neural networks (GNNs) [11].

The rest of this paper is organized as follows. Section 2 presents relevant works in the areas of mobility models, mobility prediction, and graph neural networks (GNNs). In Section 3, we formalize the problem as a time series prediction task. A first prediction scheme based on a Markov Chain model is developed in Section 4. In Section 5 two alternative neural network prediction schemes are designed: one based on a multilayer perceptron (MLP) and the other on graph neural networks (GNNs). Simulations performed to illustrate the applicability of the proposed schemes are presented in Section 6. Concluding remarks and future lines of work to be addressed are stated in Section 7.

2. State-of-the-Art Literature Review

As mentioned above, existing works are framed in the context of available individual (i.e., disaggregated) mobility data. In [1], to develop a communication infrastructure for connected and automated vehicles (CAVs), a clustering algorithm is proposed, whose evaluation is carried out via simulations that make use of a Gauss–Markov mobility (GMM) model [12,13] as a complementary tool for estimating future locations of single vehicles. In [2], the mobility prediction for individual vehicles is addressed within a grid partition of the city's geographical space, and a second-order Markov model is assumed to characterize each driver's mobility preferences within the grid. The employed hybrid architecture that combines a convolutional neural network (CNN) and a recurrent neural network (RNN) makes use of disaggregated historical grid occupation values, which are available for each individual vehicle; the resulting scheme improves the quality of vehicle mobility prediction, especially for those vehicles that have a strong individual mobility preference. In [3], the same authors consider a similar scenario to address individual mobility prediction to support intelligent vehicle applications; they propose a deep RNN-based algorithm whose results match the theoretical analysis and improve state-of-the-art previous results. In [5], a deep RNN making use of a long short-term memory (LSTM) architecture is proposed to predict individual vehicle mobility on a grid, based on individual sensor data; the prediction step is followed by a vehicle recruiting algorithm to optimize crowdsensing; the results improve the quantity of collected sensing data versus existing algorithms. In [4], a model based on a general-purpose sequence prediction model, named the compact prediction tree (CPT), is proposed for driver route prediction; the results show an efficient noise tolerance of the proposed algorithm. The prediction of individual vehicle trajectories, based on consecutive previous GPS locations combined with a statistical inference module for online mobility prediction, is proposed in [6]. This inference module is based on a hidden Markov model (HMM), and the prediction stage employs an improved version of the Viterbi algorithm. Such improvements reduce computational costs. In [14], the authors present a hierarchical trajectory prediction structure using a capsule neural network (CapsNet) that embeds local temporal and global spatial correlations through hierarchical capsules represented on a grid map. This procedure outperforms state-of-the-art methods.

So far, all the existing proposals address individual vehicle prediction making use of disaggregated available data. The restriction of only having access to aggregate data is one of the novel contributions of the work presented here.

3. Problem Statement

This paper addresses the comparative assessment of different schemes for the prediction of the aggregated mobility of a fleet of vehicles that move along a graph describing a city. In this type of graph, the nodes represent crossroads in the city, and the directed edges represent the roads (and their directions) between the crossings. Simple simulation-controlled scenarios will be considered for focusing on the main objective of the comparative assessment.

The aggregated mobility will be defined by the distribution of the assets along the graph nodes represented by a vector $y_t \in (\mathbb{N} \cup \{0\})^N$ for a set of discrete time stamps $t = 0, 1, \ldots$ Assuming available a sequence of occupation values $y_t,\ t = 0, 1, \ldots, T-1$, the objective is to predict the next value y_{T}.

Three approaches are considered to perform such predictions. On the one hand, motivated by the underlying graph structure, a Markov chain (MC) model, defined by a transition matrix Π, is assumed to model the mobility of each vehicle; then, such a matrix (denoted as $\widehat{\Pi}$) is estimated using past mobility data $y_t,\ t = 0, 1, \ldots, T-1$. Ultimately, the prediction is performed using $\widehat{\Pi}$:

$$y^p_{\mathrm{T}} = \widehat{\Pi} \cdot y_{\mathrm{T}-1}. \tag{1}$$

The second approach considers an MLP architecture trained in a supervised manner using the past mobility data $y_t,\ t = 0, 1, \ldots, T-1$, so that:

$$y^p_{\mathrm{T}} = \mathrm{MLP}_{\boldsymbol{\Theta}}(y_{\mathrm{T}-1}, \ldots, y_{\mathrm{T}-\mathrm{d}}) \tag{2}$$

where $\boldsymbol{\Theta}$ is the set of adjustable parameters and $d \geq 1$ defines the size of the time window of previously selected values as predictors.

Finally, the third approach proposes a GNN that takes advantage of both the underlying graph structure and the proven effectiveness of graph neural networks (GNNs) for time series prediction [15]. Such GNN is also trained in a supervised manner with the past mobility data $y_t,\ t = 0, 1, \ldots, T-1$, so that in general:

$$y^p_{\mathrm{T}} = \mathrm{GNN}(y_0, \ldots, y_{\mathrm{T}-1}). \tag{3}$$

The details of the MLP and GNN implementations and the use of the available data are provided in Section 5.2.

The three proposed approaches are comparatively assessed together with other simpler prediction schemes employed as baselines.

4. A Markov Chain Model

This aggregate model assumes that the mobility of each individual vehicle in the graph can be modeled via a Markov chain, defined by the transition probabilities between the connected nodes. We formalize this assumption in the remainder of this section.

4.1. Markov Chains and Transition Matrices

Let $\{X_t\}_{t=0,1,\dots,T}$ be a Markov Chain (MC) [9,16], which represents a discrete-time stochastic process with finite state space $X_t \in S = \{s_1, \dots, s_N\}$, whose associated probability distribution satisfies the *Markov property*:

$$P(X_{t+1}{=}s_j \mid X_t{=}s_i, X_{t-1}{=}s_k, \dots, X_0{=}s_l) = P(X_{t+1}{=}s_j \mid X_t{=}s_i). \tag{4}$$

Let Π be the Markov chain transition matrix so that given two states s_i, s_j, the term (j,i) of such matrix is denoted as $\Pi_{ji} = \Pi(s_j \leftarrow s_i) = P(X_{t+1}{=}s_j \mid X_t{=}s_i)$ and represents the transition probability from state i to state j. Defined this way, the columns of Π are stochastic vectors so that $\forall i = 1, \dots, N$ we have $\Pi_{\cdot i} = [\Pi_{1i}, \dots, \Pi_{Ni}]$ satisfies $\Pi_{ji} \geq 0, \forall j \in \{1, \dots, N\}$ and $\sum_{j=1}^{N} \Pi_{ji} = 1$.

For each t, let us denote P_t as the (discrete) probability distribution corresponding to X_t. P_t is defined as a stochastic vector $P_t = [p_t^{(1)}, \dots, p_t^{(N)}]$ with $p_t^{(i)} \geq 0, \forall i \in \{1, \dots, N\}$ and $\sum_{i=1}^{N} p_t^{(i)} = 1$. Note that P_0 corresponds to the distribution of the initial condition or initial state X_0; in case such an initial state is known to take a deterministic value $X_0 = s_j$, for some $j \in \{1, \dots, N\}$, then we have that $P_0 = \delta_j = [\delta_{1,j}, \dots, \delta_{j,j}, \dots, \delta_{N,j}]$ meaning that $p_0^{(j)} = 1$ and $p_0^{(i)} = 0$, $\forall i \neq j$.

According to (4), the distributions P_t, $t = 0, \dots, T-1$ for $T \in \mathbb{N}$, $T > 1$ satisfy the condition

$$P_{t+1} = \Pi \cdot P_t, \; t = 0, \dots, T-1. \tag{5}$$

Note that (5) defines a dynamical system that—starting from P_0—allows to recursively compute for each time, t, the probability distribution, P_t, corresponding to X_t.

According to (4), $\{X_t\}, t \in \{0, \dots, T\}$ are not independent variables, but their joint distribution can be characterized stepwise making use of the corresponding conditional distributions. Hence, we can take a sample of the Markov chain by relying on such conditional distributions, which are completely defined with P_0 and matrix Π.

4.2. Binary Vector Codification of States

Let us code any state s_j via column vector $\delta_j = [\delta_{j1}, \dots, \delta_{jj}, \dots, \delta_{jN}]^\top \in \{0,1\}^N$ (where $^\top$ denotes the transpose of a vector/matrix) whose elements are all zero but the j-th one. Then, considering this codification, we have that $X_t \in \{0,1\}^N$ so that (with some abuse of notation) $E[X_t] = P_t$, the distribution of X_t (when considering $X_t \in S$). Furthermore, the distribution of $X_{t+1}|X_t{=}s_j$, can be computed as $E[X_{t+1}|X_t{=}s_j] = \Pi \cdot \delta_j = \Pi_{\cdot j}$, the j-th column of $\Pi = [\Pi_{\cdot 1}| \dots |\Pi_{\cdot n}]$.

4.3. Model for Aggregated Data

Let us consider that the data derived from M realizations of the Markov chain $\{x_t^{(m)}\}_{t=0,\dots,T}^{m=1,\dots,M}$ is only available in an aggregated form so that for each $t = 0, \dots, T$ only a sample of the random variable (using the binary vector notation):

$$Y_t = \sum_{m=1}^{M} X_t^{(m)} \in (\mathbb{N} \cup \{0\})^N \tag{6}$$

is available. Note that

$$E[Y_t] = \sum_{m=1}^{M} E[X_t^{(m)}] = \sum_{m=1}^{M} P_t = M \cdot P_t, \tag{7}$$

which will allow to design of some model estimation procedures.

4.4. Estimation of Transition Matrix with Aggregated Data

Let us consider that we only have available a sample $y_t = \sum_{m=1}^{M} x_t^{(m)}$ of Y_t defined in (6). Then, following (7) we can construct the following:

$$\widehat{P}_t = \frac{1}{M} y_t, \; t = 0, \ldots, T, \tag{8}$$

an estimate of P_t using only y_t, the single sample of Y_t. Hence, according to (5), we can expect Π to satisfy

$$y_{t+1} \approx \Pi \cdot y_t, \; t = 0, \ldots, T-1, \tag{9}$$

which leads to the following least squares estimation (LSE) scheme [17] for estimating Π:

$$\widehat{\Pi} = \arg\min_{\Pi} \sum_{t=0}^{T-1} \|y_{t+1} - \Pi \cdot y_t\|^2, \; \text{ subject to } \mathbf{1}_N \cdot \Pi = \mathbf{1}_N, \text{ and } \Pi \geq 0, \tag{10}$$

where $\mathbf{1}_N = [1, 1, \ldots, 1, 1]$ is the N-dimensional vector of all ones. Note that if the data could be broken down or disaggregated by each individual vehicle m, all such available time series $x_t^{(m)}$ could be consecutively stacked into a single one $z = [x_t^{(1)} | \ldots | x_t^{(M)}]$; then the application of this LSE estimator given by (10) to z as a new time series, would lead to the known MLE estimator of Π for disaggregated data (see Appendix A.)

Concerning the size of the parameter space of this MC model, the parameters to be estimated (i.e., learned) are the coefficients of the corresponding transition matrix Π, whose size amounts to N^2.

Problem (10) is a quadratic optimization problem with positivity and linear constraints. This type of problem is denoted as *Quadratic programming* or more specifically as a *Constrained least squares* problem. The canonical quadratic form for (10) and the corresponding Karush–Kuhn–Tucker's necessary conditions are derived in Appendix B.

Remark 1. *The LSE estimate of* Π *proposed in* (10) *has been selected among other possible estimates (based on alternative distance measures between vectors), because the quadratic cost in* (10) *results in an efficiently solvable optimization problem. For instance, alternative costs could have been defined by making use of other norms* $\|y_{t+1} - \Pi \cdot y_t\|_q$*; furthermore, considering that* y_{t+1} *and* $\Pi \cdot y_t$ *can be normalized according to* (8) *to become a discrete distribution, some cost involving the Kullback-Leibler divergences* $KL(\frac{y_{t+1}}{M}, \frac{\Pi \cdot y_t}{M})$ *and* $KL(\frac{\Pi \cdot y_t}{M}, \frac{y_{t+1}}{M})$ *could also be a reasonable option.*

5. Two Approaches Using Neural Networks

In this section, two schemes that make use of Neural Network (NN) architectures are presented. Neural network approaches have proven to be a very efficient solution in supervised learning scenarios with high uncertainty. The first approach directly employs an MLP-based recurrent formulation that provides a natural nonlinear learning approach to improve standard time series prediction schemes; the second approach makes use of a GNN that the capabilities of an NN while making use of the available knowledge, in the form of a network structure, of the city mobility paths.

5.1. Recursive Application of a Multilayer Perceptron

This standard approach considers an MLP structure that generalizes the nonlinear formulations of the usual recurrent schemes for time series prediction. The fundamental formulation would take the form of (2) where the set of adjustable parameters $\mathbf{\Theta}$ is determined via a training scheme. In general, such training of the MLP (i.e., estimation of $\mathbf{\Theta}$) should be appropriately performed to control bias and overfitting of the resulting model; in any case, it is interesting to note the parallelism between the following basic training scheme

$$\widehat{\mathbf{\Theta}} = \arg\min_{\mathbf{\Theta}} \sum_{t=d-1}^{T-1} \|y_{t+1} - \mathrm{MLP}_{\mathbf{\Theta}}(y_t, \ldots, y_{t-d+1})\|^2, \tag{11}$$

and the procedure for estimating Π via (10). Note that a priori the computation of $\mathbf{\Theta}$ via (11) would not exploit the available knowledge about the city graph structure as (10) does. Our evaluation has focused on an MLP with a configurable number of layers (l), where the first layer contains an input per pair of nodes (N) and feature (d), and the next layers are fully connected and of size N. The learnable parameters in this model are all the weights and biases in the network, which in this case is $\mathcal{O}(N^2 \times l)$. The specific number can be calculated as $(d \times N) \times N + (l-1) \times (N^2) + (l-1) \times N$.

5.2. Application of Graph Neural Networks

In this section, we assess the use of graph neural networks (GNNs) for the prediction problem defined in Section 3. GNNs are specially designed to process and learn from graph-structured data, the main applications being node classification, link prediction, and graph classification.

To address the prediction of the number of assets associated with each node, a node representation or *embedding* is constructed by the GNN. To perform this task, the GNN gathers and aggregates information from each node's neighbors, based on the connections within the graph. Such information is passed and updated between nodes and their neighbors through several layers of computation. Each layer typically involves two main steps: message passing, where information from neighboring nodes is aggregated, and node update, where the aggregated information is used to update the node representation.

The application of GNNs in this paper is grounded on the assumption that the learned representation can capture complex relationships and dependencies inherent in the graph that may help for the prediction of the number of assets in each node.

The main GNN architecture used in this paper is based on graph convolutional networks (GCNs) [18,19], which is formalized as follows. Let us denote $\mathbf{f}^{(i)}$ the feature vector that gathers the employed information associated with the i-th node of the network. Specifically, $\mathbf{f}^{(i)}$ will be composed by a concatenation of the following elements:

$$\mathbf{f}^{(i)} = \left(y^i_{\mathcal{F}} \parallel i_{id} \parallel T\right), \tag{12}$$

where $\mathcal{F} = \{T-W, \ldots, T-1\} \subset \{0, \ldots, T-1\}$ and $y^i_{\mathcal{F}} = \left(y^i_{T-W}, \ldots, y^i_{T-1}\right)$ is the vector of previous asset-occupation values at node i that are provided to the GNN, i_{id} is a (one-hot) encoding of the node i identification, and T is the time instant where the prediction is carried out. Note that W, the number of samples used as node features, is a parameter of the model. Hence, if we accordingly denote $(y^p_{\mathrm{T}})^i$ to represent the i-th component of y^p_{T} (i.e., the predicted occupation of such i-th node at time T), the node-wise input–output formulation of the GNN for our problem is as follows:

$$h_i = \mathrm{AF}_{\mathbf{\Theta_1}}\left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{e_{j,i}}{\sqrt{\hat{d}_j \hat{d}_i}} \mathbf{f}^{(j)}\right) \tag{13}$$

$$(y^p_{\mathrm{T}})^i = \mathrm{MLP}_{\mathbf{\Theta_2}}\left(h_i \parallel \mathbf{f}^{(i)}\right) \tag{14}$$

where AF defines a tunable affine transformation and MLP is a Multilayer Perceptron, their adjustable parameters being denoted by $\mathbf{\Theta_1}$ and $\mathbf{\Theta_2}$, respectively. The two sets of parameters are characterized by two hyper-parameters: the number of hidden layers and the number of neurons in each layer. $\mathcal{N}(i)$ is the neighborhood of node i and $\hat{d}_i = 1 + \sum_{j \in \mathcal{N}(i)} e_{j,i}$, where $e_{i,j}$ denotes the edge weight between nodes i and j. In our basic formulation, all edge weights are set to 1.

The first step carries out a weighted feature aggregation of node i neighborhood nodes according to the topology and weights of the network. Then, the $\mathrm{AF}_{\mathbf{\Theta_1}}$ processes, via a tunable affine transformation, the aggregated feature vector gathering the previous asset-occupation values, node id and instant of time, in order to provide the embedding h_i. Then, the $\mathrm{MLP}_{\mathbf{\Theta_2}}$ processes a concatenation of the AF output h_i with the node feature vector $\mathbf{f}^i$, to provide the prediction value $(y^p_{\mathrm{T}})^i$.

The weights in the model described are trained through a stochastic gradient descent strategy on the whole set of nodes on a sufficiently large time series dataset ($\mathcal{T}_{\text{train}}$).

The error in each epoch is computed by averaging the mean squared error over a randomly selected set $\mathcal{T}_i \subset \mathcal{T}_{\text{train}}$.

To complement this approach, we have also included an alternative that replaces the computation of h for each node with a Graph Attention Network (GAT) [20]. In this type of model, the weight for each neighbor is not fixed and will be learned in the training process.

In contrast with the previous model, the number of learnable parameters in the network does not depend on the number of nodes in the graph but on the dimension of the MLP, determined by $\mathbf{\Theta_1}$ and $\mathbf{\Theta_2}$. The number of parameters for $\mathbf{\Theta_1}$ is $(|f|+1)\times|h|$, where $|f|$ is the number of input features and $|h|$ is the dimensionality of h_i, which has been set to 8 in our experiments. The number of parameters for $\mathbf{\Theta_2}$ is $(|f|+|h|) \times (M^2) \times (l-1) + M + (l-1) \times M + 1$, where M is the number of neurons at every hidden layer, and l is the number of hidden layers. In this paper, we have chosen to limit our model to two hidden layers, of the same dimensionality as the h vector. Time complexity at prediction time is thus $\mathcal{O}(E + V \times ((|h|+|f|)l \times (M^2)))$. In addition to the learned parameters, the model needs to have access to the graph to generate predictions. Using a sparse representation for the adjacency matrix, the memory requirement is $\mathcal{O}(|E|)$ [18]

6. Evaluation

In order to evaluate the proposed approaches, a set of baseline prediction models was constructed for comparison purposes. The comparative evaluation was carried out on a set of synthetic time series of aggregated asset mobility. In the following section, we describe the baseline prediction models, the construction of the testing time series, and the metrics used for the evaluation.

6.1. Estimation Schemes

The following two simple prediction procedures are employed as the reference baselines:

1. A stationary predictor using the previous value in the time series:

$$y^p_{\mathrm{T}} = y_{\mathrm{T\text{-}1}} \tag{15}$$

which is equivalent to assuming an MC model with $\Pi = I_N$ (i.e., to assign probability 1 to the self-links in the graph).

2. An MC predictor assuming a transition matrix $\overline{\Pi}$, whose random column vectors follow a uniform distribution over the non-zero elements of the adjacency matrix of the graph:

$$y^p_{\mathrm{T}} = \overline{\Pi} \cdot y_{\mathrm{T\text{-}1}} \tag{16}$$

3. A predictor that is based only on the second step of the GNN approach (Equation (14)) without calculating h. This has been labeled NoGNN in the evaluation. The motivation for this predictor is to evaluate the effect of message passing in the prediction error for the selected architecture.

Alternatively, the three more elaborated approaches proposed in this paper are as follows:

4. A more elaborated MC predictor according to (1) where $\widehat{\Pi}$ is obtained from (10). Considering the model used in the generation of the evaluation datasets detailed in

Section 6.2, this predictor should achieve very high accuracy given enough training data, as its underlying Markovian model is the same as that used for generating the datasets in Section 6.2.

5. A set of predictors using an MLP architecture to be trained in the spirit of (11) (with a proper use of training and validation sets as explained below), and to be employed according to (2).
6. A set of predictors based on a GNN according to (3), where several parameters can be adjusted. First, different number of characteristics W, whether the one-hot representation of the node id has been used (using the `Id` suffix in the model name) and whether the number of time steps is included in the feature vector (using the `T` suffix in the model name). For the sake of simplicity, our evaluation uses the same hyperparameters for both MLP_{Θ_1} and MLP_{Θ_2}: two hidden layers to allow for more expressiveness and two possible configurations for the number of neurons in each layer (64 and N) to evaluate the effect of dimensionality in both training and prediction error.

Note that predictors 1, 2 and 4 rely on models based on homogeneous (i.e., time-invariant) MCs. Alternatively, predictors 5 and 6 allow modeling time-varying behaviors in some of its configurations.

6.2. Evaluation Datasets

In general, aggregated mobility data can be generated by simulating individual assets, each with its own individual mobility patterns, and calculating the aggregated occupation of each node at the desired time interval. This enables quite sophisticated behaviors that might take into account specific objectives (and the corresponding trajectories) for each driver. In this paper, we assume a common behavior on all the assets, defined by an MC. Note that this mobility behavior fits precisely the assumption of the scheme described in (1). This deliberate choice facilitates a more rigorous assessment of all other prediction schemes. Furthermore, this mobility model allows the application of a Monte Carlo simulation procedure that ends up providing an aggregated mobility behavior that suits our goal of aggregate mobility prediction. The graph used in the simulations corresponds to the center of the city of Madrid, which is characterized by a graph with $N = 2000$ nodes. The result of each simulation is a time series of aggregated occupations in each node.

Therefore, each time series is fully determined by the shared mobility model of the assets and the number of assets (M). For each of these two factors, multiple time series of sufficiently large size ($T_D = 1000$) were constructed, where each time series represents the aggregation of M samples generated following the corresponding model.

The different employed (possibly non-homogeneous) MC models are characterized by the corresponding transition probability matrices, which must be compatible with the adjacency matrix of the city graph. In particular, the designed models were:

(a) The column-wise uniform transition matrix $\overline{\Pi}$ (which favors the performance of predictor (16)).

(b) A randomly generated stochastic transition matrix Π_r.

6.3. Training and Testing Strategies

Among the predictors proposed in Section 6.1, predictor 4 based on computing $\widehat{\Pi}$ (an estimator of the transition matrix Π), predictor 5 employing an MLP, and predictor 6 using a GNN make use of T previously available values of the time series to construct a cost function. Predictor 4 uses the values of the time series to compute $\widehat{\Pi}$ according to (10), whereas the other two use them (as T_{train}) to train the weights of either the MLP or the GNN.

Hence, for every time series available of size T_D, S different sub-series of size T were constructed by randomly selecting S possible sliding windows from within the full interval of size T_D. These sub-series can be characterized as follows: $\mathcal{T}_i = \mathcal{T}_{a_i,b_i} = \{t_{a_i}, \ldots, t_{b_i}\}$, $i = 1, \ldots, S$, where $a_i + T - 1 = b_i < T_D$. Each of these sub-series is used to train each model ($\mathcal{T}_{\text{train}} = \mathcal{T}_{a_i,b_i}$), and the results are evaluated by producing predictions of type $y^p_{t_{b_j}+1}$ based on $\mathcal{T}_{\text{test}} = \mathcal{T}_{a_j,b_j}$, for any $j \neq i$, which are then compared to the true value $y_{\text{test}} = y_{t_{b_j+1}}$. After extensive experimentation, the values $S = 10$ and $T_D = 300$ were selected for the final evaluation.

6.4. Evaluation Metrics

As far as the definition of errors is concerned, different measures can be defined in line with the comments included in the Remark at the end of Section 4.4. On the one hand, the $\|\cdot\|_2$ norm was considered following the quadratic cost imposed in the estimation and training procedures; on the other hand, alternative measures not aligned with the quadratic cost, such as the $\|\cdot\|_1$ norm, were also considered.

Alternatively, we can bypass the stochasticity inherent in the Markov model (reflected in each different Monte Carlo implementation), by comparing the performance of the predictors with the optimal prediction (as an expected value) that could be calculated assuming available the *true* transition matrix Π at $T-1$. This ideal predictor serves to define a performance upper bound (i.e., a prediction error lower bound). Such unavoidable lower bound error, which comes from the intrinsic randomness of vehicle mobility (inherent in any Markovian model), can be removed for comparative purposes.

6.5. Results

The results of the evaluation are summarized in Table 1. A more illustrative representation is included in Figure 1, where the distribution of the predictive error for each model and each set of evaluation parameters can be seen. Baseline predictor 1, denoted as I_N, and baseline predictor 2, denoted $\overline{\Pi}$, are defined, respectively, in Equations (15) and (16). Each column corresponds to a different type of model employed to generate the evaluation time series; as explained above, each one of those time series is constructed using a corresponding transition matrix: generator (a) with $\overline{\Pi}$ and generator (a) with Π_r.

It is important to note that the time series generation schemes (a) and (b) proposed in Section 6.2 make use of an MC-type formulation. Hence, predictor 4 in Section 6.1, based on an MC assumption, performs especially well in such favorable scenarios. The MLP-based predictor 5, even though it can capture Markovian behaviors, cannot compete with the MC optimal predictor in these scenarios since it does not exploit the available graph knowledge. The GNN-based predictor 6, when incorporating the node id and an attention mechanism, performs better than the proposed baseline schemes 1 and 2, and its performance is very close to that of the MC-assumption-based predictor 4.

In all scenarios, GNN models outperform their NoGNN counterparts, which demonstrates the effectiveness of the GNN approach. The simpler MLP variants show higher prediction errors than any of the GNN models. This is especially true of the MLP with more hidden neurons. These results can be explained because MLP models do not incorporate any a priori knowledge about graph structure. Although nodes that are not connected in the graph should not affect each other, the layers in the MLP are fully connected. All weights, including those that are zero, need to be learned from training data However, the chosen values for T_S may not be enough, especially when the number of tunable weights is very high.

As for other factors to take into account, it is worth mentioning that low asset counts, which imply low node occupancy (i.e., the average occupancy of each node close to zero), lead to higher errors in general; therefore, in that scenario, all predictors behave almost indistinguishably, due to the stochasticity of time series generative models. More specifically, it is worth noting the relatively high error of the uniform predictor in datasets generated using a column-wise uniform transition matrix ($\overline{\Pi}$). This is due to the stochastic nature of the generative models explained above (see Section 6.2).

Table 1. Table of mean errors and their standard deviation for each evaluation dataset. Baseline models are included first. The lowest error (up to the second decimal place) for each row is highlighted.

	#assets	**Trans.**	**Model**	I_N	$\overline{\Pi}$	**MC**	**NoGNN**	**NoGNNId**	**GNN1**	**GNN3**	**GNN5**	**GNNId1**	**GNNIdT1**	**GATId1**	**MLP1x2_64**	**MLP1x2_len**
error	10^2	$\overline{\Pi}$	mean	11.1	8.1	8.2	10.6	9.3	8.0	10.6	10.6	8.0	8.0	8.0	10.4	10.6
			std	0.7	0.2	0.3	0.3	0.4	0.3	0.3	0.2	0.3	0.3	0.3	0.3	0.3
		Π_r	mean	10.4	8.2	7.5	11.0	8.9	8.0	11.0	11.0	7.6	7.6	7.5	10.4	11.0
			std	0.6	0.3	0.4	0.5	0.4	0.3	0.4	0.4	0.3	0.3	0.3	0.3	0.4
	10^3	$\overline{\Pi}$	mean	35.2	26.9	25.0	47.5	29.5	26.1	26.1	26.0	25.2	25.1	25.2	33.7	47.3
			std	1.2	0.9	0.8	1.9	0.8	0.7	0.8	0.7	0.7	0.7	0.7	1.2	2.1
		Π_r	mean	33.3	31.0	23.1	56.1	28.3	27.4	26.2	25.6	24.1	24.0	23.6	34.5	56.1
			std	1.2	1.6	0.8	4.8	1.0	1.2	1.0	0.9	1.0	0.9	0.9	2.5	5.3
	10^4	$\overline{\Pi}$	mean	111.4	118.9	79.1	112.3	94.6	92.2	87.1	85.3	80.1	79.9	79.9	148.3	366.6
			std	3.3	6.0	2.2	5.1	2.7	3.0	2.6	2.5	2.1	2.1	2.1	16.4	24.7
		Π_r	mean	105.0	201.4	73.0	120.1	91.3	108.9	90.0	86.9	76.7	81.1	76.2	131.5	474.6
			std	3.8	18.6	2.7	13.5	3.7	9.4	6.1	5.3	2.9	6.9	3.3	18.5	60.3
optimal prediction error	10^2	$\overline{\Pi}$	mean	7.9	2.0	2.3	7.1	5.0	1.7	7.1	7.1	1.7	1.7	1.8	6.7	7.0
			std	0.3	0.1	0.3	0.2	0.2	0.1	0.3	0.3	0.1	0.1	0.1	0.2	0.3
		Π_r	mean	7.5	3.9	2.1	8.2	5.1	3.5	8.2	8.2	2.3	2.2	2.1	7.4	8.2
			std	0.4	0.3	0.3	0.5	0.3	0.3	0.5	0.4	0.2	0.2	0.2	0.3	0.5
	10^3	$\overline{\Pi}$	mean	24.9	10.4	3.1	40.4	16.0	8.0	7.5	7.1	4.4	4.1	4.1	22.8	40.3
			std	0.8	0.7	0.1	2.1	0.6	0.5	0.4	0.4	0.3	0.3	0.4	1.4	2.4
		Π_r	mean	24.0	20.9	2.8	51.2	16.6	14.7	12.4	11.5	7.3	7.0	5.8	25.6	51.1
			std	1.1	1.8	0.2	5.1	0.8	1.0	0.7	0.7	1.4	1.4	1.2	3.0	5.8
	10^4	$\overline{\Pi}$	mean	78.6	88.9	7.3	80.0	52.5	48.1	37.7	33.2	15.4	14.4	14.3	125.3	357.9
			std	2.5	6.8	0.3	6.1	2.2	2.8	1.6	1.5	1.3	1.3	1.4	18.8	25.3
		Π_r	mean	75.6	187.5	6.7	94.4	54.3	79.9	53.1	47.0	23.5	33.6	21.5	109.4	468.8
			std	3.0	19.6	0.4	16.8	3.7	11.8	8.5	7.2	2.8	12.4	4.9	21.3	61.1

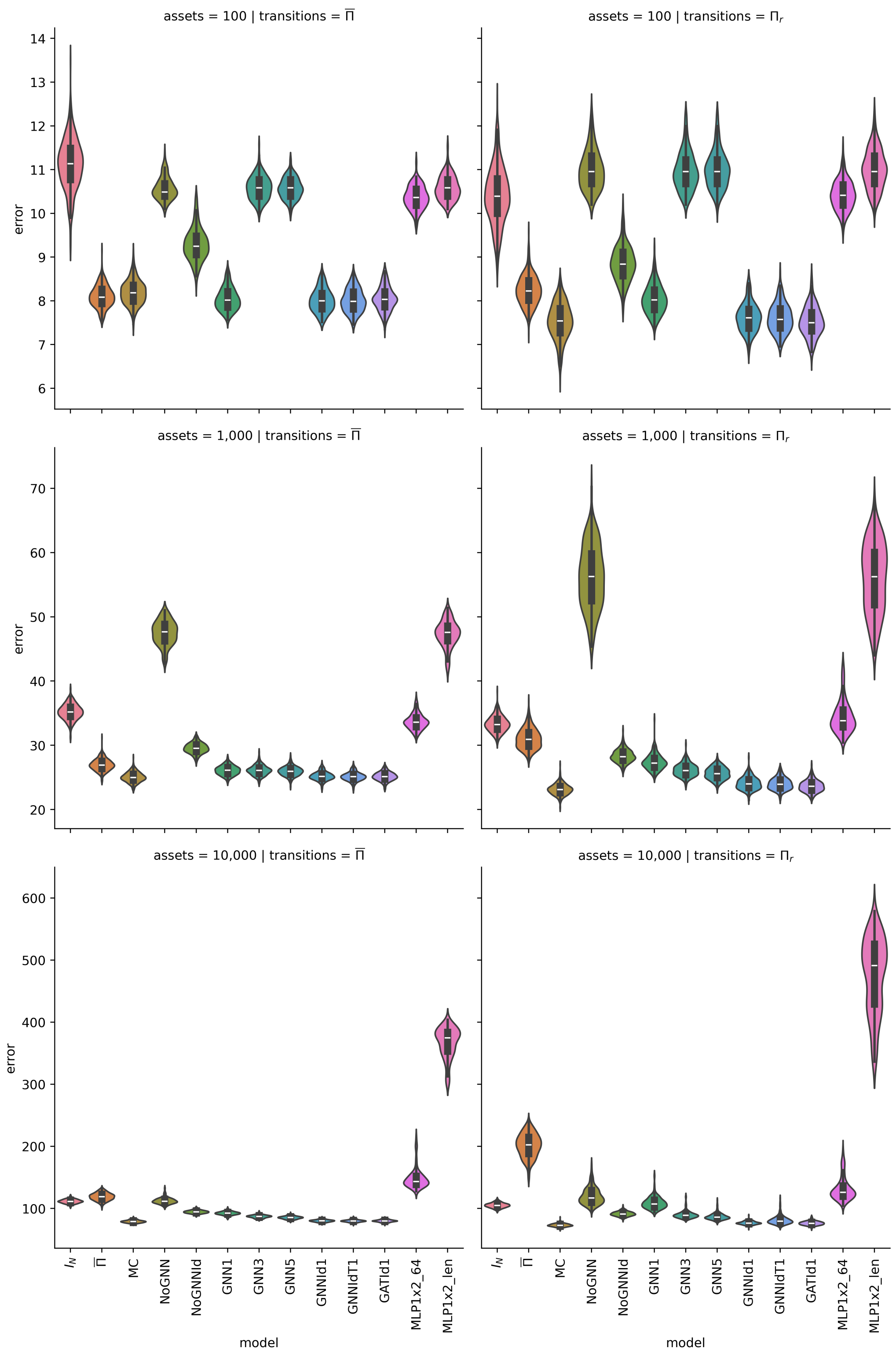

Figure 1. Errors for each model and type of dataset.

7. Concluding Remarks and Future Work

The motivation behind this paper is to study the capabilities and limitations of GNN models in the prediction of aggregate mobility of a vehicle fleet in a city using only historical aggregate mobility information. An evaluation scenario is proposed, where asset mobility is governed by a Markovian model that is formalized and described in detail. Three schemes are proposed and evaluated. Two of these approaches exploit the underlying graph structure used to define vehicle mobility in a city. One of them further exploits the assumption of an underlying Markov model.

The first method is based on the estimation of the transition probability matrix of a Markov model that is supposed to characterize the mobility behavior of each individual. This estimation-based method shows good performance in scenarios where the mobility time series are generated using MC models, such as the ones used in the paper; in these cases, the Markov-estimation-based method outperforms other prediction schemes.

The second proposed method employs an MLP in a recurrent formulation to generate predictions for the next node occupation as a function of previous time series values. Even though this approach can capture Markovian behaviors, its performance is not competitive since it does not exploit the available graph knowledge.

The third proposed prediction method is based on a graph neural network trained in a supervised manner. In general, this method performs much better than the proposed baseline prediction schemes, and in some configurations outperforms or competes with the first proposed method (GNN). To do so, the GNN model needs enough representation power to discriminate between different nodes and neighbors. Our results show that, in the types of scenarios proposed, this can be achieved by using a higher value of characteristics W, or by including node identifiers in the feature vector. It should be noted that for the GNN models to achieve competitive performance, node identifiers need to be provided as part of the node feature vector. This is true even in scenarios with a uniform transition matrix.

Our study highlights the effectiveness of the Markov model, partly due to our reliance on Markov processes for synthetic data generation. However, the current synthetic data approach may not reflect some of the intricate time series patterns inherent in complex traffic data. Urban traffic dynamics inherently exhibit repetitive patterns and deviations, since humans very often tend to follow pre-established routes, and/or variations of these. GNNs may excel at modeling complex time series patterns, but the absence of such nuances in our synthetic data generation suggests untapped potential in our GNN approach. Future research should incorporate synthetic models tuned with real-world traffic data. By capturing the complexities of urban movement patterns, we can fully leverage GNNs in traffic analysis. In addition, more information could be incorporated into both Markov-based and GNN models to address the problem of aggregate prediction in these new scenarios. Specifically, the GNN architecture presented in this paper could be improved in three different ways: by using different neighbor sampling strategies, by using a more nuanced aggregation function that better discriminates both temporally and by node, and by using more expressive node identification features.

Author Contributions: Conceptualization, J.F.S.-R., J.M. and P.J.Z.; methodology, J.F.S.-R. and P.J.Z.; software, J.F.S.-R., R.V.-R. and P.J.Z.; validation, J.F.S.-R.; formal analysis, J.F.S.-R. and P.J.Z.; investigation, J.F.S.-R. and P.J.Z.; resources, J.F.S.-R. and P.J.Z.; writing—original draft preparation, P.J.Z.; writing—review and editing, J.F.S.-R., J.M. and P.J.Z.; visualization, J.F.S.-R.; supervision, J.F.S.-R., J.M. and P.J.Z.; project administration, J.F.S.-R., J.M. and P.J.Z.; funding acquisition, J.F.S.-R., J.M. and P.J.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by Cabify Matriz S. L. through the Cátedra Cabify (Cabify Chair) at Escuela Técnica Superior de Ingenieros (ETSI) Telecomunicación of the Universidad Politécnica de Madrid, in part by the Ministerio de Ciencia e Innovación of Spain under grant PID2020-112502RB/AEI/10.13039/501100011033, and it is also a part of the project TED2021-129189B-C21/TED2021-129189B-C22, financed by MCIN/AEI/10.13039/501100011033 and the European Union "NextGenerationEU"/PRTR.

Data Availability Statement: The data employed in this paper were synthetically generated via defined random procedures.

Acknowledgments: The authors are grateful for the help and support of the researchers and those responsible for the Cátedra Cabify (Cabify Chair) at ETSIT-UPM.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MC	Markov chain
MLP	multi-layer perceptron
GNN	graph neural network
GCN	graph convolutional network
GAT	graph attention network

Appendix A. Relationship with the Estimator of the Transition Matrix Assuming Vehicle Disaggregated Data

Many scenarios in the literature involve making individual predictions using the dataset $\{x_t^{(m)}\}_{t=0,\ldots,T}^{m=1,\ldots,M}$, which consists of M finite samples (up to time T) in disaggregated form. In this case, one can construct $\widehat{\Pi}$, an estimate of matrix Π, following several well-known procedures.

Appendix A.1. MLE Estimator of the Transition Matrix

Let us denote $M_i = \#\{x_t^{(m)}, t = 0, \ldots, T-1; m = 1, \ldots, M,$ such that $x_t^{(m)} = \delta_i\}$ and $M_{ji} = \#\{x_t^{(m)}, t = 0, \ldots, T-1; m = 1, \ldots, M,$ such that $x_t^{(m)} = \delta_i$ and $x_{t+1}^{(m)} = \delta_j\}$. So, we have that $M_i = \sum_{i=1}^N M_{ji}$ and $\sum_{i=1}^N M_i = M \cdot T$.

Let us consider that $M_i \neq 0$, $i = 1, \ldots, N$. Then, $\widehat{\Pi}$, and the MLE estimator of Π is given by the following:

$$\widehat{\Pi}_{ji_{\mathrm{MLE}}} = \frac{M_{ji}}{M_i}, \ \forall i, j = 1, \ldots, N. \tag{A1}$$

Appendix A.2. Proposed Estimator of the Transition Matrix Particularized to Disaggregated Data

Note that if we apply the results from Section 4.4 to the case where $M = 1$ (which is equivalent to considering that the data have been disaggregated and then concatenated), we have that $y_t = x_t = \delta_j$ for some $j \in \{1, \ldots, n\}$. In this case, (10) becomes the following:

$$\begin{aligned}\widehat{\Pi} &= \arg\min_{\Pi} \sum_{t=0}^{T-1} \|x_{t+1} - \Pi \cdot x_t\|^2 = \arg\min_{\Pi} \sum_{i=1}^{N}\sum_{j=1}^{N} M_{ji}\|\delta_j - \Pi \cdot \delta_i\|^2 \\ &= \arg\min_{\Pi} \sum_{i=1}^{N}\sum_{j=1}^{N} M_{ji}\|\delta_j - \Pi_i\|^2, \\ &\text{subject to } \Pi \geq 0, \text{ and } \mathbf{1}_N \cdot \Pi = \mathbf{1}_N\end{aligned} \tag{A2}$$

so that the overall optimization problem can be disclosed in the following separate problems:

$$\begin{aligned}\widehat{\Pi}_i = \arg\min_{\Pi_i} \sum_{j=1}^{N} M_{ji}\|\delta_j - \Pi_i\|^2 = \arg\min_{\Pi_i} \sum_{j=1}^{N} M_{ji}\Big((1-\Pi_{ji})^2 + \sum_{k\neq j}\Pi_{ki}^2\Big), \\ \text{subject to } \Pi_i \geq 0, \text{ and } \mathbf{1}_N \cdot \Pi_i = 1.\end{aligned} \tag{A3}$$

Applying the Karush–Kuhn–Tucker conditions, we have that $\widehat{\Pi}_i$ is the solution of the following set of equations:

$$\begin{aligned}
&\frac{\partial}{\partial \Pi_{pi}}\left[\sum_{j=1}^{N} M_{ji}\Big((1-\Pi_{ji})^2+\sum_{k\neq j}^{N}\Pi_{ki}^2\Big)+\lambda_i(\sum_{l=1}^{N}\Pi_{ji}-1)\right]+\lambda_i-\mu_{pi}\\
&=2M_{pi}(\Pi_{pi}-1)+\sum_{j\neq p}^{N}2M_{ji}\Pi_{pi}+\lambda_i-\mu_{pi}\\
&=2\Big[M_{pi}(\Pi_{pi}-1)+(M_i-M_{pi})\Pi_{pi}\Big]+\lambda_i-\mu_{pi}\\
&=2\Big[M_i\Pi_{pi}-M_{pi}\Big]+\lambda_i-\mu_{pi}=0,\ p=1,\dots,N
\end{aligned} \tag{A4}$$

together with the following conditions:

$$\widehat{\Pi}\geq 0, \tag{A5}$$

$$\mathbf{1}_N\cdot\widehat{\Pi}=\mathbf{1}_N \tag{A6}$$

$$\mu_{ij}\geq 0,\ i,j\in\{1,\dots,N\} \tag{A7}$$

$$\mu_{ij}\widehat{\pi}_{ij}=0,\ i,j\in\{1,\dots,N\} \tag{A8}$$

The solutions of these equations are given by the following:

$$\widehat{\Pi}_{pi}=\frac{M_{pi}}{M_i},\ \lambda_i=0,\ \mu_{pi}=0,\ \forall i,p=1,\dots,N \tag{A9}$$

which correspond with the MLE estimator presented in (A1). Since the solution of (A4) is obtained for $\lambda_i=0$, $\mu_{pi}=0$, and satisfies the restrictions, this suggests that each $\widehat{\Pi}_i$ can be obtained by solving the unconstrained version of (A3).

Appendix B. Quadratic Canonical form and the Karush–Kuhn–Tucker Conditions for the Estimation Scheme of the Transition Matrix

Appendix B.1. Canonical Form

Denoting $\Pi_{i\cdot}$, the i-th row of Π, problem (10) can be formulated in a canonical quadratic form, where Π can be rewritten in vector form as $\Pi_v=[\Pi_{1\cdot},\dots,\Pi_{N\cdot}]^\top$, as follows:

$$\begin{gathered}
\widehat{\Pi}_v=\arg\min_{\Pi_v} d+c^\top\cdot\Pi_v+\Pi_v^\top\cdot Q_N\cdot\Pi_v,\\
\text{subject to } E\cdot\Pi_v=\mathbf{1}_N^\top,\ \text{and }-I_{N^2\times N^2}\Pi_v\leq 0_{N^2},
\end{gathered} \tag{A10}$$

where

$$\begin{gathered}
d=\sum_{i=1}^{N}\sum_{t=0}^{T-1}(y_{t+1}^{(i)})^2,\quad c^\top=[c_1^\top,\dots,c_N^\top]\text{ with }c_i^\top=-2\sum_{t=0}^{T-1}y_{t+1}^{(i)}y_t^\top,\\
Q_N=\mathrm{diag}(Q,\dots,Q)\text{ with }Q=\sum_{t=0}^{T-1}y_ty_t^\top,\text{ and }E=\begin{bmatrix}\delta_1^\top & \cdots & \delta_1^\top\\ \cdots & \cdots & \cdots\\ \delta_N^\top & \cdots & \delta_N^\top\end{bmatrix}\in\mathbb{R}^{N\times N^2}
\end{gathered} \tag{A11}$$

Such a canonical formulation of the cost function is derived as follows:

$$\begin{aligned}
\sum_{t=0}^{T-1} \|y_{t+1} - \Pi \cdot y_t\|^2 &= \sum_{t=0}^{T-1} \sum_{i=1}^{N} (y_{t+1}^{(i)} - \Pi_{i\cdot} \cdot y_t)^2 = \sum_{i=1}^{N} \sum_{t=0}^{T-1} \left[(y_{t+1}^{(i)})^2 - 2y_{t+1}^{(i)} \Pi_{i\cdot} \cdot y_t + (\Pi_{i\cdot} \cdot y_t)^2 \right] \\
&= \sum_{i=1}^{N} \left[\sum_{t=0}^{T-1} (y_{t+1}^{(i)})^2 - 2 \left(\sum_{t=0}^{T-1} y_{t+1}^{(i)} y_t^\top \right) \cdot \Pi_{i\cdot}^\top + \Pi_{i\cdot} \cdot \left(\sum_{t=0}^{T-1} y_t y_t^\top \right) \cdot \Pi_{i\cdot}^\top \right] \\
&= \sum_{i=1}^{N} \left[d_i + c_i^\top \cdot \Pi_{i\cdot}^\top + \Pi_{i\cdot} \cdot Q \cdot \Pi_{i\cdot}^\top \right] = d + c^\top \cdot \Pi_v + \Pi_v^\top \cdot Q_N \cdot \Pi_v,
\end{aligned} \tag{A12}$$

This can be rewritten to only use the inequality form for the restrictions, as follows:

$$E \cdot \Pi_v \le \mathbf{1}_N, \;\; -E \cdot \Pi_v \le \mathbf{1}_N, \;\; -I_{N^2 \times N^2} \cdot \Pi_v \le 0, \tag{A13}$$

so that using standard notation, such restrictions can be formulated as follows:

$$A \cdot \Pi_v \le b, \text{ with } A = \begin{bmatrix} E \\ -E \\ -I_{N^2 \times N^2} \end{bmatrix}, \text{ and } b = \begin{bmatrix} \mathbf{1}_N \\ \mathbf{1}_N \\ 0_{N^2} \end{bmatrix}. \tag{A14}$$

Appendix B.2. Karush–Kuhn–Tucker Conditions

In order to apply the Karush–Kuhn–Tucker conditions to problem (10), we define the Lagrangian as follows:

$$\sum_{t=0}^{T-1} \|y_{t+1} - \Pi \cdot y_t\|^2 + \sum_{j=1}^{N} \lambda_j \cdot [\mathbf{1}_N \Pi_{\cdot j} - 1] - \sum_{i=1}^{N} \sum_{j=1}^{N} \mu_{ij} \pi_{ij}, \quad \text{with } \mu_{ij} \ge 0. \tag{A15}$$

Now, denoting $\Pi_{i\cdot}$ as the i-th row of Π, $\lambda = (\lambda_1, \ldots, \lambda_N)^\top$ and $\mu_{i\cdot} = (\mu_{i1}, \ldots, \mu_{iN})^\top$, we have to solve for $\widehat{\Pi}$, which satisfies the following:

$$\begin{aligned}
\frac{\partial}{\partial \Pi_{i\cdot}} \sum_{t=0}^{T-1} \|y_{t+1} - \Pi \cdot y_t\|^2 + \lambda - \mu_{i\cdot} \Big|_{\Pi = \widehat{\Pi}} &= \sum_{t=0}^{T-1} (y_t^\top \cdot \widehat{\Pi}_{i\cdot}^\top - y_{t+1}^i) y_t + \lambda - \mu_{i\cdot} \\
= \left(\sum_{t=0}^{T-1} y_t y_t^\top \right) \cdot \widehat{\Pi}_{i\cdot}^\top - \sum_{t=0}^{T-1} y_{t+1}^i y_t + \lambda - \mu_{i\cdot} &= 0, \; i = 1, \ldots N,
\end{aligned} \tag{A16}$$

together with the following conditions:

$$\widehat{\Pi} \ge 0, \tag{A17}$$

$$\mathbf{1}_N \cdot \widehat{\Pi} = \mathbf{1}_N, \tag{A18}$$

$$\mu_{ij} \ge 0, \; i, j \in \{1, \ldots, N\}, \tag{A19}$$

$$\mu_{ij} \widehat{\pi}_{ij} = 0, \; i, j \in \{1, \ldots, N\}. \tag{A20}$$

Note that all the systems (A16) for $i = 1, \ldots, N$ share the same Gramian matrix, $G = \sum_{t=0}^{T-1} y_t y_t^\top$, obtained as the sum of matrices $G_t = y_t y_t^\top$, and constructed via the outer product of vectors y_t and $y_t^\top$.

Considering condition (A18), the solutions $\widehat{\Pi}_{i\cdot}$, $i = 1, \ldots, N$ must satisfy $\sum_{i=1}^{N} \Pi_{i\cdot} = \mathbf{1}_N$, so that by adding all the equations in (A16), we obtain the following:

$$\begin{aligned}\left(\sum_{t=0}^{T-1} y_t y_t^\top\right) \cdot \mathbf{1}_N^\top - \sum_{t=0}^{T-1} y_t \sum_{i=1}^{N} y_{t+1}^i + M \cdot \lambda - \sum_{i=1}^{N} \mu_{i\cdot} \\ = M \cdot \sum_{t=0}^{T-1} y_t - \sum_{t=0}^{T-1} y_t \cdot M - M \cdot \lambda - \sum_{i=1}^{N} \mu_{i\cdot} = 0.\end{aligned} \tag{A21}$$

which defines a simple relationship between the constraint constants.

References

1. Abdel-Halim, I.T.; Fahmy, H.M.A.; Bahaa-El Din, A.M. Mobility prediction-based efficient clustering scheme for connected and automated vehicles in VANETs. *Comput. Netw.* **2019**, *150*, 217–233. [CrossRef]
2. Liu, W.; Shoji, Y. Edge-assisted vehicle mobility prediction to support V2X communications. *IEEE Trans. Veh. Technol.* **2019**, *68*, 10227–10238. [CrossRef]
3. Liu, W.; Shoji, Y. DeepVM: RNN-based vehicle mobility prediction to support intelligent vehicle applications. *IEEE Trans. Ind. Inform.* **2020**, *16*, 3997–4006. [CrossRef]
4. Amirat, H.; Lagraa, N.; Fournier-Viger, P.; Ouinten, Y. Nextroute: A lossless model for accurate mobility prediction. *J. Ambient. Intell. Humaniz. Comput.* **2020**, *11*, 2661–2681. [CrossRef]
5. Zhu, X.; Luo, Y.; Liu, A.; Tang, W.; Bhuiyan, M.Z.A. A deep learning-based mobile crowdsensing scheme by predicting vehicle mobility. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 4648–4659. [CrossRef]
6. Irio, L.; Ip, A.; Oliveira, R.; Luís, M. An adaptive learning-based approach for vehicle mobility prediction. *IEEE Access* **2021**, *9*, 13671–13682. [CrossRef]
7. Jiang, W.; Luo, J. Graph neural network for traffic forecasting: A survey. *Expert Syst. Appl.* **2022**, *207*, 117921. [CrossRef]
8. Gomes, D.; Ruelens, F.; Efthymiadis, K.; Nowe, A.; Vrancx, P. When Are Graph Neural Networks Better Than Structure-Agnostic Methods? In Proceedings of the I Can't Believe It's Not Better Workshop: Understanding Deep Learning through Empirical Falsification, New Orleans, LA, USA, 28 November–9 December 2022.
9. Kemeny, J.G.; Snell, J.L. *Finite Markov Chains*; Springer: Berlin/Heidelberg, Germany, 1960.
10. Haykin, S. *Neural Networks: A Comprehensive Foundation*; Prentice Hall PTR: Hoboken, NJ, USA, 1998.
11. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **2008**, *20*, 61–80. [CrossRef] [PubMed]
12. Camp, T.; Boleng, J.; Davies, V. A survey of mobility models for ad hoc network research. *Wirel. Commun. Mob. Comput.* **2002**, *2*, 483–502. [CrossRef]
13. Bai, F.; Helmy, A. A survey of mobility models. In *Wireless Adhoc Networks*; University of Southern California: Los Angeles, CA, USA, 2004; Volume 206, p. 147.
14. Qin, Y.; Guan, Y.L.; Yuen, C. Spatiotemporal capsule neural network for vehicle trajectory prediction. *IEEE Trans. Veh. Technol.* **2023**, *72*, 9746–9756. [CrossRef]
15. Jin, M.; Koh, H.Y.; Wen, Q.; Zambon, D.; Alippi, C.; Webb, G.I.; King, I.; Pan, S. A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. *arXiv* **2023**, arXiv:2307.03759.
16. Ibe, O. *Markov Processes for Stochastic Modeling*; Academic Press: Cambridge, MA, USA, 2013.
17. Mendel, J.M. *Lessons in Estimation Theory for Signal Processing, Communications, and Control*; Pearson Education: London, UK, 1995.
18. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv* **2017**, arXiv:1609.02907.
19. Zhang, S.; Tong, H.; Xu, J.; Maciejewski, R. Graph convolutional networks: A comprehensive review. *Comput. Soc. Netw.* **2019**, *6*, 11. [CrossRef] [PubMed]
20. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. *arXiv* **2018**, arXiv:1710.10903.