Advancing Public Transit Efficiency with Graph Neural Networks and Reinforcement Learning: From Flow Prediction to Dynamic Ride-Matching

Siavash Farazmand

A Thesis

in

The Department

of

Concordia Institute for Information Systems Engineering (CIISE)

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montréal, Québec, Canada

September 2024

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Siavash Farazmand**

Entitled: **Advancing Public Transit Efficiency with Graph Neural Networks and Reinforcement Learning: From Flow Prediction to Dynamic Ride-Matching**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________ Chair
Dr. Arash Mohammadi

______________________ Examiner
Dr. Ali Ayub

______________________ Supervisor
Dr. Nizar Bouguila

______________________ Co-supervisor
Dr. Zachary Patterson

Approved by ______________________
Chun Wang, Chair
Department of Concordia Institute for Information Systems Engineering (CIISE)

__________ 2024 ______________________
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Advancing Public Transit Efficiency with Graph Neural Networks and Reinforcement Learning: From Flow Prediction to Dynamic Ride-Matching

Siavash Farazmand

Using artificial intelligence (AI) techniques, this thesis illustrates how public transportation can be optimized in line with smart city visions. Deep learning models can be used to analyze complex datasets in order to predict demand, streamline operations, and adapt dynamically to real-time conditions. The research focuses on two key components: first, a Graph Neural Network (GNN) with probabilistic node embeddings is used to predict passenger flow in bus networks, improving accuracy and optimizing resource allocation. This method is validated using Automated Passenger Counting (APC) data from Laval, Quebec. The second component employs a Reinforcement Learning (RL) algorithm to enhance ride-matching and vehicle routing in on-demand shared mobility services, particularly addressing the first-mile problem in suburban areas. Sensitivity analyses confirm the adaptability and effectiveness of both approaches, showcasing the potential of AI to improve transportation efficiency in diverse urban and suburban contexts.

Acknowledgments

I would like to extend my deepest gratitude to those who supported me throughout this journey. To my family, your unwavering encouragement and belief in me have been my greatest source of strength. My sincere thanks to Professors Zachary Patterson and Nizar Bouguila for their invaluable guidance, insightful feedback, and constant support, which have greatly shaped this work. I am also deeply grateful to the Buspas team, particularly Dr. Wissem Maazoun, for their collaboration and expertise, which have significantly contributed to the success of this research. Your collective support has been instrumental in the completion of this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

Smart cities are characterized by the harmonious integration of human capital, social assets, and Information and Communications Technology (ICT) infrastructure to address urban challenges such as traffic congestion, air pollution, and inefficient energy use (Ramaprasad, Sánchez-Ortiz, & Syn, 2017). At their core, smart cities use data and advanced technologies such as IoT sensors, AI-driven analytics, and renewable energy solutions to maximize resource utilization, reduce environmental impact, and promote long-term sustainability. Smart cities prioritize activities that address key sustainability issues, such as efficient energy management, waste reduction, public transportation improvements, and the preservation of natural resources.

One critical sector pivotal in the development of a smart city is public transportation (Guo, Tang, & Guo, 2020). Through the use of data collected from citizens, devices, and various assets, smart transportation systems can improve the monitoring, management and optimization of traffic and transportation networks. In a smart city, smart transportation systems use cutting-edge technologies to improve efficiency, safety, and sustainability of urban mobility. By using real-time traffic monitoring, intelligent transportation networks, and data-driven decision-making, these systems can help to optimize the flow of people and goods. In an idealized smart city, residents and visitors can use seamless, accessible, and eco-friendly transportation options thanks to autonomous and connected vehicles, public transportation networks, and smart infrastructure. The objective is to create an urban mobility ecosystem that improves the commuter experience while reducing environmental impact and supporting sustainable city development.

The use of artificial intelligence (AI) techniques is in line with this vision, providing powerful tools for further optimizing public transportation through the use of data-driven insights and predictive models. The term Artificial Intelligence (AI) refers to a wide range of techniques, including machine learning (ML), deep learning, and statistical techniques. The concept of machine learning refers to algorithms that are used in order to allow systems to learn from data and improve their performance over time with no explicit programming input. As a subset of machine learning, deep learning uses multiple layers of neural networks to capture complex patterns in large datasets. As opposed to traditional statistical approaches based on predetermined assumptions, artificial intelligence and machine learning provide greater flexibility by dynamically adapting to the underlying data structures, allowing more nuanced and accurate predictions.The purpose of this thesis is to demonstrate how deep learning techniques can be used in two different ways to optimize public transportation, emphasizing that artificial intelligence can be a significant contributor to the advancement of smart cities.

1.1.1 Addressing Demand and Supply in Public Transportation

Optimising smart public transportation systems requires balancing demand and supply in real time. A two-fold approach is taken in this thesis to address this challenge: first, through the development of a graph neural network (GNN)-based model that predicts passenger demand, and second, by applying a reinforcement learning (RL) algorithm to optimize ride matching and vehicle routing in on-demand transit services.

Using passenger flow prediction, the demand side is represented by identifying where and when transportation is required. As for the supply side, the RL algorithm is concerned with efficiently matching passengers to available vehicles and routing them based on real-time conditions. It is through the combination of these two components - demand prediction and supply optimization - that a dynamic system can be created in which transportation services are able to respond flexibly to fluctuating demand patterns.

To ensure that the transportation network is adequately responsive to real-time conditions, these two systems operate simultaneously. This dual approach aligns with the broader goals of smart cities, where data-driven strategies enhance the efficiency and sustainability of urban mobility solutions.

1.1.2 Passenger flow prediction

Passenger flow prediction is a powerful tool for optimizing smart public transportation systems. By accurately forecasting passenger demand, transit agencies can strategically allocate vehicles, staff, and other resources to align with expected ridership levels (Banerjee, Morton, & Akartunalı, 2020). This leads to improved service efficiency, reduced delays, and an enhanced overall passenger experience. Predictive insights during high-demand periods enable agencies to manage resources more effectively, reducing overcrowding during peak times and avoiding underutilization during off-peak hours, resulting in a more balanced and responsive transit system. Another key advantage of accurate passenger flow prediction is the enhancement of passenger experience. By proactively adjusting schedules, frequencies, and capacities based on predicted demand, transit agencies can offer shorter wait times, fewer delays, and a more reliable and comfortable travel experience (Cats & Glück, 2019). This approach not only optimizes resource allocation but also fosters passenger satisfaction and loyalty to public transportation systems. Most existing models for passenger flow prediction, however, are based on static node embeddings based on fixed node locations and static node-to-node relationships (edges). This rigid approach ignores the dynamic nature of passenger flow behaviors and the complex, evolving relationships between the nodes within the network. These simplifications significantly reduce the predictive power and adaptability of these models.

Our work attempts to fill this gap by introducing and testing a novel approach for the prediction of passenger flow using probabilistic node embeddings within a graph neural network (GNN) model. By treating node embeddings probabilistically, our model captures the inherent uncertainties and dynamic behaviors of passenger flows at various nodes. It also captures the fluctuating interrelationships between them. Additionally, although dynamic node embeddings capture the evolution of relationships and behaviors in the network, this approach is limited in terms of its expressiveness. By incorporating a series of normalizing flows, we are able to refine the posterior distribution in order to improve prediction precision and flexibility.

1.1.3 Ride-matching and vehicle routing

One noteworthy observation when addressing the passenger flow prediction problem is the high prevalence of zeros in the table of passenger flows. Such zeros reflect periods and locations in which there are no passengers, like at many bus stops located in suburban areas of large cities. In these situations, population density tends to be low, which can make operating extensive Fixed-Route Bus Transit (FRT) expensive Alsaleh and Farooq (2023); Calabrò et al. (2022). Often, FRT systems in such regions struggle to provide satisfactory service levels, as they have low frequencies Bürstlein, López, and Farooq (2021). Consequently, suburban areas are typically characterized by high levels of automobile usage for local trips, particularly for the "first-mile" segments to public transit stations in the morning, or even for entire commutes to neighboring communities Bürstlein et al. (2021). This trend reduces transit ridership and increases traffic congestion by increasing the proportion of single-occupancy vehicle trips, as well as limiting access to Park & Ride lots.

Over the last decade, rapid advancements in communication and information technology has enabled transit agencies to introduce innovative transportation services for both urban and suburban areas Diana, Quadrifoglio, and Pronello (2009); Inturri et al. (2021). On-demand transit (ODT) as an example has shown a remarkable potential for replacing FRT systems in suburban areas to serve local trips as well as act as feeders for public transportation for the "first" or "last-mile" of commutes. Different simulation-based studies have demonstrated that ODT services show the potential to reduce the total passenger travel time, vehicle distance traveled, operating costs, and GHG emissions Diana et al. (2009); Edwards and Watkins (2013); Inturri et al. (2021).

Even though these studies have demonstrated promising results, certain limitations have also been acknowledged. As an example, a case study conducted in Brooklyn, NY highlighted instances in which an ODT service might not be the best option. Furthermore, it was found that FRT systems could demonstrate superior efficiency when used under specific objectives, conditions, and constraintsYoon, Chow, and Rath (2022). Therefore, any proposed ODT service should be subjected to rigorous simulation testing in order to validate its efficacy before it is deployed.

As a result, to test the feasibility of an on-demand shared mobility solution, we conducted a study focused on the suburbs of Montreal Meshkani, Farazmand, Bouguila, and Patterson (2024).

Using simulations with varied numbers of vehicles, trip requests, and different ride-matching algorithms, we assessed the impact of these factors on the effectiveness of the service. The study demonstrated the potential effectiveness of the proposed on-demand shared mobility service, and also highlighted the critical role of the ride-matching algorithm compared to other factors, such as the number of vehicles and their relocation strategies.

For on-demand shared mobility services to be effective and efficient, ride-matching algorithms play an essential role. These algorithms are responsible for matching passengers with available vehicles, optimizing routes, minimizing waiting times and travel distances, and maximizing the number of requests fulfilled. Although traditional approaches, such as heuristics and rule-based methods, can be effective in simple situations, they often fall short in complex, real-world environments where conditions are continuously changing and demand fluctuates.

In this context, Reinforcement Learning (RL) algorithms have demonstrated superior performance due to their ability to learn optimal policies that maximize long-term benefits rather than only immediate gains. To achieve the most efficient ride-matching and routing decisions, RL algorithms continuously improve their performance by interacting with the environment, adjusting strategies based on feedback, and balancing exploration and exploitation. This capacity to dynamically adapt to changing conditions and optimize for long-term objectives, such as minimizing overall system costs, reducing vehicle idle times, and enhancing passenger satisfaction, makes RL a powerful tool for on-demand shared mobility services. As a result of using RL, on-demand transit systems may be able to improve operational efficiency and service quality, ultimately improving the appeal and sustainability of public transportation.

For RL research to advance, two key areas need to be improved: time efficiency and performance, and the flexibility of the platform. Effective time management and high performance are essential for making RL models practical, enabling them to handle complex tasks quickly. A flexible testing platform is equally important, since the fast pace of RL development demands environments that can easily adapt to and test new algorithms. It is essential to consider both efficiency and adaptability in order to enhance progress and ensure that real-world applications of RL are realized.

In response to these challenges, this research introduces a reinforcement learning (RL) algorithm specifically designed to address the complexities involved in on-demand shared mobility services. Based on real-world Automated Passenger Counting (APC) data from Laval, our model is tailored specifically for these contexts and has been tested to ensure its relevance and applicability. Additionally, implementation within the SUMO simulation environment facilitates both algorithm development and simulation in a unified manner. By addressing the critical issues of time efficiency, performance optimization, and platform flexibility, this research aims to advance the practical application of RL in optimizing on-demand public transportation systems.

1.2 Contributions

A comprehensive framework is presented in this study in order to enhance the sustainability and efficiency of urban transportation systems. By developing a probabilistic model for predicting passenger flows, the research provides a foundation for optimizing resource allocation and improving service reliability for transit agencies. In order to accomplish this, a new deep learning method for predicting passenger traffic in a bus network was developed, which extends a Graph Neural Network (GNN) model to a variational autoencoder and enhances it with normalizing flows so that spatial and temporal correlations can be captured in a non-deterministic manner. Using real data from Laval, Canada, the proposed model outperforms existing methods such as multi-layer perceptron (MLP), convolutional neural networks (CNN), spatial-temporal graph convolution networks (STGACN), and variational graph neural networks (VGRAN) in terms of mean absolute error (MAE), root mean square error (RMSE), and percent accuracy, even with small bin sizes.

The second part of this research presents a novel RL algorithm for ride-matching and vehicle routing, which is specifically designed to optimize the operation of on-demand transit services. Automated Passenger Counting (APC) data from Laval, Quebec, is used to rigorously test and benchmark the algorithm against SUMO's built-in algorithms and a Q-network, demonstrating the ability of the algorithm to enhance the efficiency and adaptability of on-demand transit services.

1.3 Thesis Overview

The thesis is structured as follows:

- Chapter 2 presents a novel method for short-term passenger flow prediction.
- Chapter 3 details the development of a new rollout RL algorithm for ride-matching and vehicle routing.
- Chapter 4 concludes the thesis with a summary of findings and a discussion of future research directions.

Chapter 2

A Variational Graph Convolution Network with Normalizing Flows for Passenger Flow Prediction[1]

Passenger flow prediction plays a crucial role in intelligent transportation systems, aiming to facilitate affordable and eco-friendly public transportation. In this context, we introduce a comprehensive framework known as the Variational Graph Convolution Network with Normalizing Flows for Passenger Flow Prediction (VGConvNF). By combining variational graph convolution with the acquisition of a probabilistic latent variable that encodes changes in passenger flows, this innovative approach enables the modeling of dynamic passenger flows and bus stop graph structures. Furthermore, we use the power of normalizing flows to extend the range of possible distributions in order to enhance the posterior approximation. A real-world dataset of passenger flows was used to evaluate the effectiveness of our proposed method. Results demonstrate the effectiveness of our approach in extending graph neural networks with probabilistic node embeddings and enriching the core graph neural networks. This framework significantly improves the accuracy and reliability of passenger flow prediction, thus contributing to the development of efficient and sustainable transportation systems.

[1]This paper was presented at the IEEE AI for Industries 2023 conference.

2.1 Literature Review

Several methods have been proposed to predict passenger flow using statistical and classical machine learning algorithms. Initially, passenger flow prediction development was done by simple mathematical methods such as historical averages, ordinary least squares, logistic regression, Kalman filters (S. Yang, 2011), Support Vector Machines (SVM) (Sun, Leng, & Guan, 2015), and K-nearest neighbors (L. Yang, Yang, Li, & Feng, 2019). Since most of the data in passenger flow prediction is spatio-temporal in nature, Auto Regressive Integrated Moving Average (ARIMA) (Ye, Chen, & Xue, 2019) has been used by many researchers. As a result of low precision and inability to capture complex non-linear patterns of passenger flow over time and space (e.g., over consecutive stops), most of these models were abandoned. As a consequence, Deep Learning, specifically Neural Networks (NNets), have become popular due to their ability to capture non-linear relationships among features.

A wide variety of methodologies have been combined with neural networks (NNets), including Markov chains (Xiao, Mao, & Zhang, 2018), Radial Basis Functions (RBFs) (Li, Wang, Xu, Qin, & Zhang, 2019) and Principal Component Analysis (PCAs) (S. Zhang, Liu, Shen, Wang, & Yang, 2020) to enhance their performance and versatility. It is important to note that while NNets can increase accuracy, failure to select the appropriate hyper-parameters may result in overfitting. Moreover, a common issue is that of adding layers to neural networks that increases the number of parameters significantly, resulting in a slower training process.

As passenger flow prediction has evolved, traditional neural networks (NNets) have given way to more sophisticated architectures such as recurrent neural networks (RNNs) and their variants LSTMs J. Zhang, Chen, Cui, Guo, and Zhu (2021) and GRUs (Cho et al., 2014), as well as convolutional neural networks (CNNs) (Albawi, Mohammed, & Al-Zawi, 2017; G. Liu, Yin, Jia, & Xie, 2017). The combination of CNNs and RNNs enhances the ability of RNNs to capture temporal dependencies and the ability of CNNs to extract spatial features. As a result of this integration, forecasting models have been enhanced, improving their accuracy and robustness (Ma, Zhang, Du, Ding, & Sun, 2019; W. Yu, Zhifei, Hongye, Junfeng, & Ruilong, 2019).

In spite of their substantial success, CNNs do not adequately capture the intricate topological structures and nuanced physical characteristics inherent in passenger flow data graphs due to their dependency on data arranged in Euclidean grids (S. Zhang, Tong, Xu, & Maciejewski, 2019). This limitation has been overcome through the use of graph convolutions, which provide a generalized approach for handling graph-structured data that extends beyond traditional convolutions. This adaptation has made Graph Neural Networks (GNNs) particularly suitable for applications in passenger flow prediction. Through the use of graph convolutions, GNNs are able to effectively analyze and extract meaningful insights from complex relationships and dynamics inherent in passenger flow data, thereby allowing for more accurate and robust prediction models to be developed.

Graph Neural Networks (GNNs) brought us exciting opportunities in solving flow forecasting problems because of their ability to capture complex spatial information hidden in non-Euclidean structured data.Han et al. (2019) introduced a Spatio-temporal Graph Convolution Network (STGCN) based solely on convolutions performed on graph data, predicting both inflow and outflow and achieving state of the art results on various city passenger flow data. Wang et al. Wang et al. (2021) used Hyper Graph Convolution Networks for metro passenger flow predictions. Various studies have proven the usefulness of GNNs in passenger flow predictions Yin et al. (2022).

Even though Graph Neural Networks (GNNs) have advantages over RNNs and CNNs, their deterministic node embeddings have limitations in the analysis of dynamic passenger flow data. Variational autoencoders (VAEs), as a probabilistic approach, offer a compelling alternative. As demonstrated by Kipf and Welling (2016), VAEs can be applied to graph-structured data, making use of their ability to learn the distribution of node representations within a graph. VAEs provide a probabilistic framework to capture the uncertainty inherent in passenger flow dynamics by encoding graph structure and node features into latent variables and optimizing the evidence lower bound. Probabilistic modeling is essential for modeling the highly nonlinear relationships among multiple input features, which is a key challenge faced by deterministic models. The challenge with this approach lies in the inadequacy of practically any type of distributions to model the highly nonlinear relationships between multiple input features.

In order to solve this challenge, Kobyzev, Prince, and Brubaker (2021) proposed the method of normalizing flows. The idea is to start with a simple probability distribution and transform it into a more complex one by applying a series of invertible transformations until a desired set of complexity is achieved. Here we use an extended GNN model with a series of normalizing flows for passenger flow prediction that encodes the uncertainty of node embedding and the spatio-temporal correlations among passenger flows at various bus stops.

2.2 Methodology

2.2.1 Problem Definition

We model bus passenger flow data as a graph $G = (V, E, A)$, where $V = \{v_1, v_2, ..., v_N\}$ is the set of nodes. E is the set of edges between nodes, which represents their connectivity. $A \in R^{N \times N}$ is the adjacency matrix of G. Each element $A_{i,j} \in A$ represents whether nodes i and j are connected or not.

The number of passengers in each bus leaving station v_i at time step t is denoted as x_t^i. We can aggregate this information for different nodes at time step t as $x_t = \{x_t^1, ..., x_t^N\} \in R^N$. Also matrix $X_{1:T} = (x_1, ..., x_T) \in R^{N \times T}$ represents the information of the whole network for T consecutive time steps. Our model uses this data to predict passenger flow for the next T' steps $\hat{Y}_{1:T'}$.

2.2.2 Model Architecture

Our model consists of two main parts. The first part is a variational directed graph autoencoder. This network is responsible for capturing the spatial and temporal correlations among passenger flows detected at different locations. As the second part, another variational autoencoder is used to capture the variation in the output and encode it in a latent factor. A normalizing flow based module is used to make the posterior distribution of this latent variable more expressive, enabling it to generate more intricate distributions. First we introduce the GNN model, which is used as the core, to detect spatial dependencies. We then extend the GNN with the uncertainty of latent node representations.

Graph Neural Network

A graph is a versatile data structure that holds a collection of objects (nodes) and their relationships (edges). Due to the remarkable expressive capacity of graphs, machine learning has recently gained a great deal of attention for its application to graph analysis. As a powerful representation, graphs are used across a wide range of domains, including traffic management and the prediction of passenger flow (Rahmani, Baghbani, Bouguila, & Patterson, 2023). GNNs are used to predict passenger flows due to their inherent message-passing framework. The framework implies that a node's state at a given time step is influenced by the states of its neighbors at a previous time step. In the same way, the volume of passengers at a bus stop during a specific period is intimately related to the number of passengers at adjacent bus stops, which also contribute to its traffic influx.

The GNN used in our model consists of two consecutive different blocks as shown in figure 2.1. The first part is a Gated Temporal Convolution Layer, the second and third modules are Graph Attention and Graph Convolution Layers respectively. We'll elaborate on these three parts in the following.

To capture the temporal trends of the input data, we use dilated casual convolution layers. The layer works in a similar manner to standard 1D convolution, however, a fixed number of inputs are skipped at each stage. The skipping distance is called the dilation factor. A series of these layers with different dilation factors can exponentially increase the receptive field, but with less computational power compared to RNN-based models and simple convolution.

Figure 2.2 presents the idea of dilated casual convolution. This convolution is formulated as:

$$x \star f(t) = \sum_{s} f(s) x(t - d \times s) \tag{1}$$

where x is the input, $f(t)$ is the convolution signal and d is the dilation factor. In order to capture more complex temporal dependencies, we use a simple version of the gating mechanism:

$$h = \tanh(\Theta_1 * X + b) \odot \sigma(\Theta_2 * X + c) \tag{2}$$

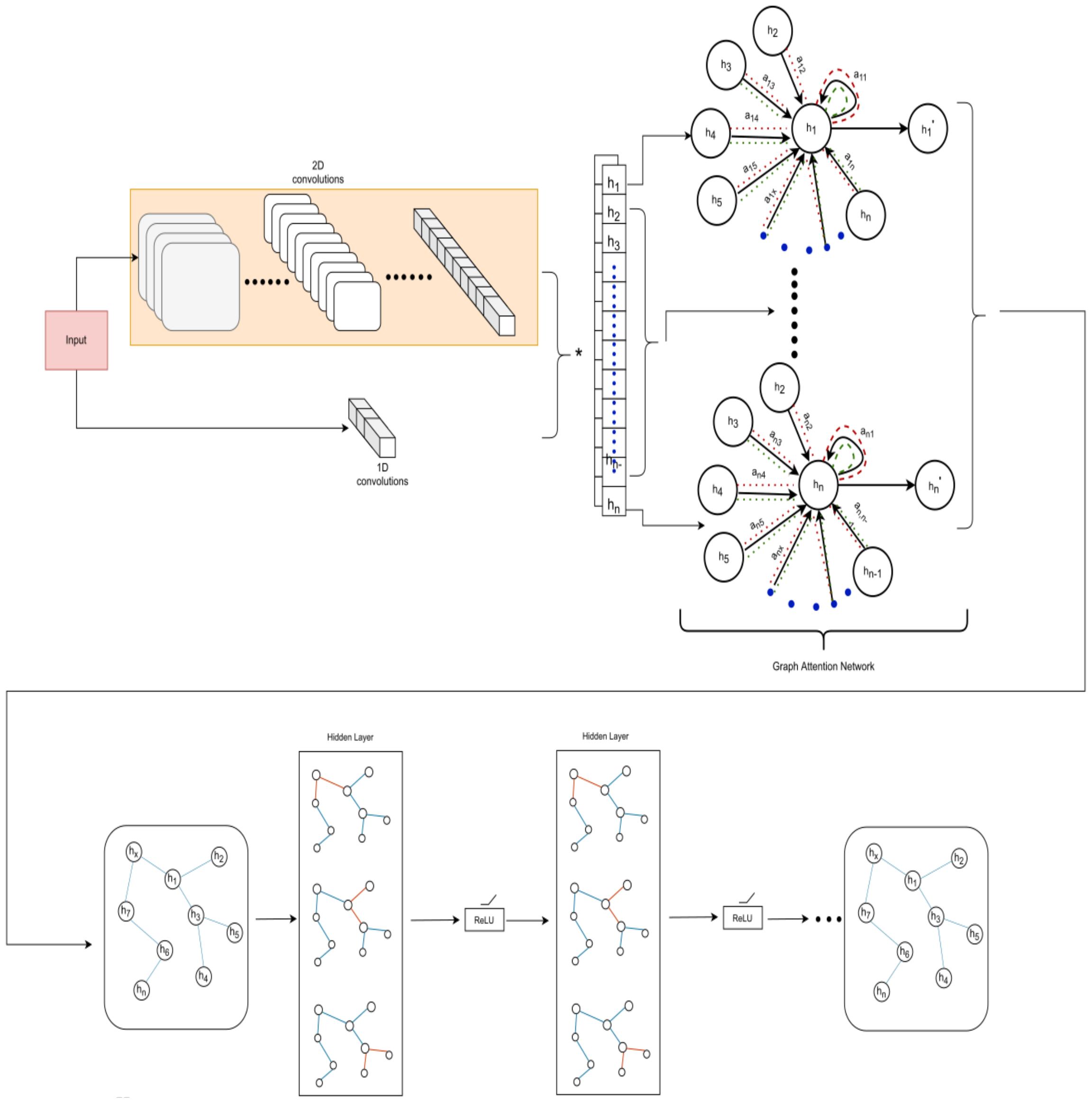


Figure 2.1: GNN diagram

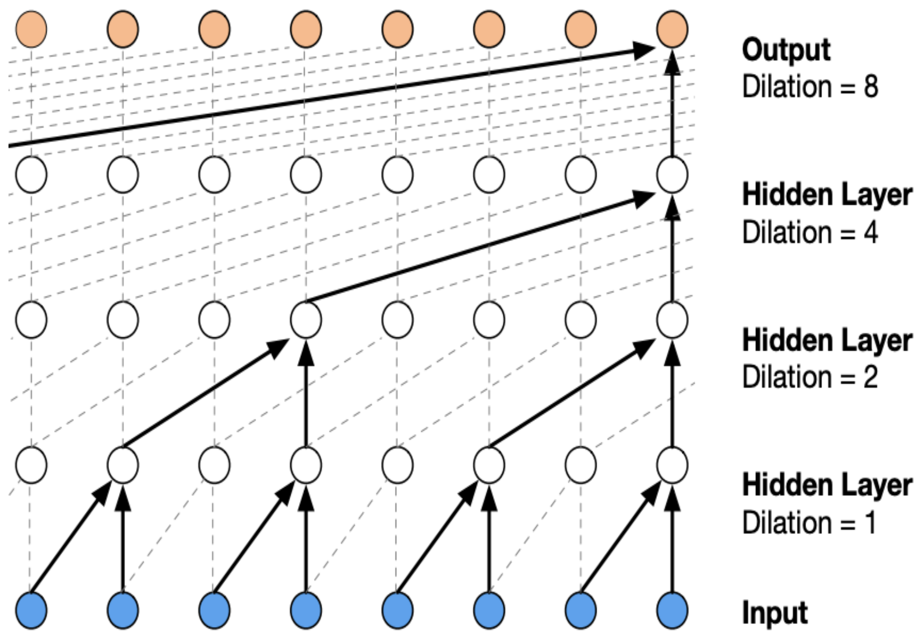


Figure 2.2: "Dilated Causal Convolution, as described in Oord (2016)"

where X is the output of the last layer, Θ_1, Θ_2, b and c are model parameters. σ, which is the sigmoid function, and $\tanh$ are two activation functions that determine the ratio of information passed to the next layer.

Graph convolutional networks (GCNs) are powerful deep learning algorithms for non-Euclidean graphs. They let each node aggregate information from its neighbors to model information flow. Adding an attention mechanism to a GCN enables us to assign different weights to the information coming from every other node. In order to learn spatial dependencies, we combine these two layers to build a Graph attention convolution block. For the graph attention part, suppose $h = h_1, h_2, ..., h_N, h_i \in R^F$ is a set of node features. First, a shared linear transformation, parameterized by the weight matrix $W \in R^{F' \times F}$, is applied to every node. Then a shared attention mechanism $a : R^{F'} \times R^{F'} \rightarrow R$ is performed on the nodes:

$$e_{i,j} = a(Wh_i, Wh_j) \tag{3}$$

These attention coefficients are then normalized using the softmax function:

$$\alpha_{i,j} = softmax_j(e_{i,j}) \tag{4}$$

As the final part, the new node feature h' for each node is computed as follows:

$$h'_i = \sigma(\sum_{j \in N_i} \alpha_{i,j} W h_j) \tag{5}$$

The GCN used here is formulated as:

$$Z = \sum_{k=0}^{K} P_f^k X W_{k1} + P_b^k X W_{k2} + \tilde{A_{adp}}^k X W_{k3} \tag{6}$$

Here $X \in \mathbb{R}^{N \times D}$ is the input to the GCN, where D is the input dimension, $W \in \mathbb{R}^{D \times M}$ represents the learnable parameters, and $Z \in \mathbb{R}^{N \times M}$ denotes the output, where M is the output dimension. Also P_f and P_b represent forward and backward transition matrices, defined as $P_f = A/rowsum(A)$ and $P_b = A^T/rowsum(A^T)$ respectively. If $A \in \mathbb{R}^{a \times b}$ we define each element of $A/rowsum(A)$ as:

$$(A/rowsum(A))_{i,j} = \frac{A_{i,j}}{\sum_{t=1}^{b} A_{i,t}} \tag{7}$$

Variational Graph Autoencoder (VGAE)

Models of transit networks typically address graph structure and spatial correlations using GNN architectures, which learn deterministic node features. In contrast, passenger flow data are complex and dynamic, presenting a challenge to deterministic node feature models. It is difficult for such deterministic models to capture the diverse patterns inherent in passenger flow dynamics.

This challenge is addressed first by encoding both graph structure and passenger flow data into a latent variable, followed by decoding this variable to reconstruct the input data. The network parameters are then learned by optimizing a variational lower bound on the graph data. Figure 2.3 provides a simplified overview of the model. The input x_t is processed through the GNN model,

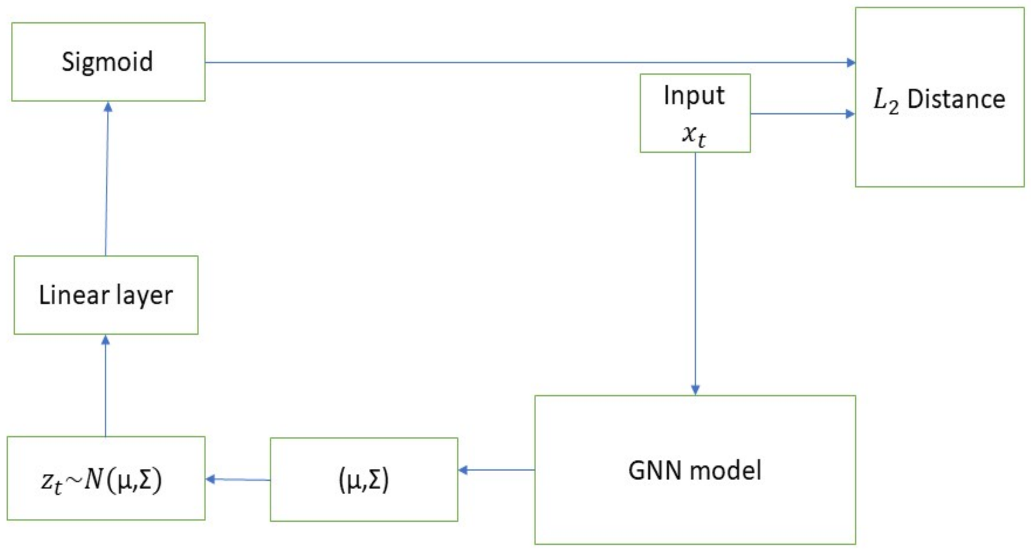


Figure 2.3: In this model architecture, a GNN (Graph Neural Network) is integrated with a variational autoencoder.

generating a latent representation $z_t \sim \mathcal{N}(\mu, \Sigma)$, which undergoes a linear transformation and sigmoid activation for further tasks. Finally, the L_2 distance block attempts to minimize the difference between the input signal and the one produced by the model $P(x_t|z_t)$.

The objective of using the VGAE is to learn a distribution over input data points x_t. To do so, suppose that each x_t is generated by the latent variable z_t. We try to maximize the probability of observing input data x_t under the generative model $P_\theta(x_t) = \int_{z_t} P_\theta(x_t|z_t)P_\theta(z_t)dz_t$; where $P_\theta(z_t)$ is the prior distribution of latent variable z_t which in our case is an isotropic multi-variate normal distribution $P_\theta(z_t) = \mathcal{N}(0, I)$, $P_\theta(x_t|z_t)$ is the generative model and θ represents the parameters of the model.

We aim to compute the posterior distribution of the latent variable z_t, but the posterior $P_\theta(z_t|x_t)$

is often intractable. Therefore, instead of learning the true posterior, we use a neural network $q_\phi(z_t|x_t)$ to approximate it, where ϕ represents the parameters that we try to infer. This method is advantageous because first, both the variation in sensor readings and the graph structure are encoded in the latent variable. Second, the encoding is done in a non-deterministic manner, which is crucial in the case of highly non-linear passenger flow data.

Here we approximate the posterior distribution of the latent factor by a mean field Gaussian:

$$q_\phi(z_t|x_t, A) = \prod_{i=1}^{N} q_{\phi_i}(z_t^i|x_t, A) \tag{8}$$

$$q_{\phi_i}(z_t^i|x_t, A) = \mathcal{N}(z_t^i|\mu_t^i, (\sigma_t^i)^2) \tag{9}$$

The parameters $\phi = \{\phi_i\}_{i=1}^N$ of the $q_{\phi_i}(z_t^i|x_t, A)$ are modeled with the Graph Neural Network explained in the last section. Specifically in our model $\mu_t = GNN_\mu(x_t, A)$ and $\sigma_t = GNN_\sigma(x_t, A)$ are vectors of mean and standard deviation variables for different nodes at each time step. In this way, not only is the graph structure and input variation encoded in the latent factor, but also the factor is a random variable.

To decode the latent variable z_t into sensor readings x_t we use the following model:

$$P_\theta(x_t|z_t) = \prod_{i=1}^{N} P_\theta(x_t^i|z_t^i) \tag{10}$$

$$P_\theta(x_t^i|z_t^i) = sigmoid(W_{dec} z_t^i + b_{dec}) \tag{11}$$

where W_{dec} and b_{dec} are parameters of the model. In equation 10 we assume that the input at each node is generated by its corresponding latent variable, independently of the latent variables at other nodes. A linear and then sigmoid function is used to map the latent variable z_t^i to x_t^i as can be seen in equation 11.

We will maximize the variational lower bound with respect to the variational parameters ϕ and generative parameters θ:

$$L_G = \mathbb{E}_{q_\phi}(\log P_\theta(x_t|z_t)) - KL(q_\phi(z_t|x_t, A)||P_\theta(z_t)) \tag{12}$$

where $KL(q(.)||p(.))$ is the Kullback-Leibler (KL) divergence between the distributions $q(.)$ and $p(.)$. Here we consider a Gaussian prior $P_\theta(z_t) = \prod_i P_\theta(z_t^i) = \prod_i \mathcal{N}(z_t^i|0, I)$.

In order to predict future time steps based on previous ones, most models use RNN based models as encoder and decoder. The shortcoming of this approach is its deterministic behaviour and lack of any stochastic variable, which is in contrast to the stochastic nature of passenger flow data. To handle this problem, we use a variational flow-to-flow architecture. There are two parts to this. The output variation is encoded in a latent variable in the inference part. Then the output is sampled from the appropriate posteriors in the generation phase. As a result of accounting for the stochastic nature of passenger flow data, our approach can handle the inherent variability in the data, which can lead to different next outputs based on the same input pattern. The increased flexibility and ability to capture the true nature of the data make flow-to-flow models more suitable for predicting passenger flow than RNNs.

Here we encode the graph structure and temporal dependencies of the passenger flow data in the latent variable u. To do so, the hidden states $h_{t'-1}$ and $y_{t'}$ are both incorporated in producing $u_{t'}$:

$$u_{t'}|y_{t'} \sim \mathcal{N}(\nu_{u,t'}, \delta_{u,t'}^2) \tag{13}$$

$$(\nu_{u,t'}, \delta_{u,t'}) = f_\psi^{enc}(y_{t'}, h_{t'-1}) \tag{14}$$

where $\nu_{u,t'}$ and $\delta_{u,t'}$ represent the mean and standard deviation of the posterior. Also f_ψ^{enc} can be any function, which in our case is a neural network with ψ as its parameters. The posterior distribution of u is defined as:

$$q_\varphi(u_{\leq T'}|y_{\leq T'}) = \prod_{t'=1}^{T'} q_\varphi(u_{t'}|y_{\leq t'}, u_{<t'}) \tag{15}$$

where ϕ represents the parameters of the inference model, $y_{\leq t'}$ is the passenger flow sequence up to the current moment and $u_{<t'}$ represents the corresponding latent variables before t'.

The latent variable is used to generate the output. $u_{t'}$ is sampled from the prior $P_\vartheta(u_{t'})$, and then passed through the neural network $P_\vartheta(y_{t'}|u_{t'})$ with ϑ as its parameters:

$$y_{t'} \sim P_\vartheta(y_{t'}|u_{t'}) = \prod_{t'=2}^{T'} P_\vartheta(y_{t'}|y_{1:t'-1}, u_{t'}) \tag{16}$$

At each time step, the prior $P_\vartheta(u_{t'})$ depends on $h_{t'-1}$ the last hidden state of GRU:

$$u_{t'} \sim \mathcal{N}(\nu'_{t'}, \delta^2_{t'}) \tag{17}$$

$$(\nu_{t'}, \delta_{t'}) = f_\psi^{prior}(h_{t'-1}) \tag{18}$$

where $\nu_{t'}$ and $\delta_{t'}$ are the mean and standard deviation of the prior. Also f_ψ^{prior} is a neural network. The generative distribution is conditioned on both the latent variable and the GRU hidden state:

$$y_{t'}|u_{t'} \sim \mathcal{N}(\nu_{y,t'}, \delta^2_{t'}(y, t')) \tag{19}$$

$$(\nu_{y,t'}, \delta_{y,t'}) = f_\psi^{dec}(u_{t'}, h_{t'-1}) \tag{20}$$

Here also $\nu_{y,t'}$ and $\delta_{y,t'}$ are the mean and standard deviation of the generative distribution and f_ψ^{dec} is another neural network. The hidden state of GRU is updated autoregressively:

$$h_{t'} = GRU(y_{t'}, u_{t'}, h_{t'-1}) \tag{21}$$

Now, having the distribution of $P_\vartheta(y_{t'}|u_{\leq t'}, y_{<t'})$ and $P_\vartheta(u_{t'}|y_{<t'}, u_{<t'})$, we can define the joint

distribution of generative model:

$$P_\vartheta(y_{\leq T'}, u_{\leq T'}) = \prod_{t'=1}^{T'} P_\vartheta(y_{t'}|u_{\leq t'}, y_{<t'}) P_\vartheta(u_{t'}|y_{<t'}, u_{<t'}) \tag{22}$$

Now, similar to the graph learning part, a variational lower bound is optimized to learn both the parameters of inference and generative model:

$$\begin{aligned} L_H = \mathbb{E}_{q_\varphi(u_{\leq T'}|y_{\leq T'})} \left[\sum_{t'=1}^{T'} \log P_\nu(u_{t'}|y_{\leq T'}, u_{<T'}) \right] \\ + \sum_{t'=1}^{T'} [-KL(q_\phi(u_{t'}|y_{<T'}, u_{<T'})||P_\nu(u_{t'}|y_{<T'}, u_{\leq T'}))] \end{aligned} \tag{23}$$

There's an important drawback with the above method: it doesn't learn a very expressive posterior distribution. We're going to use the method of normalizing flows, proposed by Kobyzev et al. (2021), to solve this. A normalizing flow describes the transformation of a probability density through a sequence of invertible mappings. By repeatedly applying the rule for change of variables, more complicated and expressive posterior distributions can be obtained.

Given a latent variable u_0 with a simple initial distribution $P_0(u_0)$, a latent variable u_k can be defined as:

$$u_k = f_k(v_{k-1}) = f_k(f_{k-1}(...f_1(v_0))) \tag{24}$$

where $f(.) = f_k(f_{k-1}(...(f_1(.))))$ is a series of invertible and differentiable functions. The distribution of the random variable u_k and can be computed as:

$$\begin{aligned} P_k(u_k) = P_{k-1}(u_{k-1})|\det(\frac{d_{u_{k-1}}}{d_{u_k}})| \\ = P_{k-1}(u_{k-1})|\det(\frac{f_k^{-1}(u_k)}{d_{u_k}})| \\ = P_{k-1}(u_{k-1})|\det(\frac{d_{f_k(u_{k-1})}}{d_{u_{k-1}}})^{-1}| \\ = P_0(u_0)|\det(\frac{d_{u_k}}{d_{u_0}})|^{-1} \end{aligned} \tag{25}$$

Here we use a series of planar flows, defined as:

$$f(u_{t'}) = u_{t'} + w_0 \tanh(w_1^T u_{t'} + b) \tag{26}$$

where w_0, w_1 and b are learnable parameters. The Jacobian of this transformation is:

$$|\det(\frac{df}{du_{t'}})| = |1 + w_0^T \tanh'(w_1^T u_{t'} + b) w_1| \tag{27}$$

Using this method, we are able to produce more complicated posterior distributions for the latent variable. A benefit of using this type of flow is that the additional terms that it provides can be computed in linear time. As a result, we will optimize the following variational lower bound instead of 23:

$$\begin{aligned} &L_F(y_{\leq T'}, \vartheta, \varphi) = \mathbb{E}_{q_\varphi(u_{\leq T'}|y_{\leq T'})}[\log P_\vartheta(y_{\leq T'}, u_{\leq T'}) - \log q_\varphi(u_{\leq T'}|y_{\leq T'})] \\ &= \mathbb{E}_{q_{u_0}}[\log P_\vartheta(y_{\leq T', u_k}) - \log q(u_k)] \\ &= \mathbb{E}_{q_{u_0}}[\log P_\vartheta(y_{\leq T', u_k})] - \mathbb{E}_{q_{u_0}}[\log q(u_0)] + \beta \mathbb{E}_{q_{u_0}}\left[\sum_{k=1}^{K} \log \det |\frac{du_k}{du_{k-1}}|^{-1}\right] \end{aligned} \tag{28}$$

Here, the identity in the second line comes from the fact that if $z_0 \sim q_0(z_0)$ passes through a series of transformations to produce new densities $z_k = f_k(z_{k-1})$, then $\mathbb{E}_{q_k}(h(z)) = \mathbb{E}_{q_0}(h(f_k(f_{k-1}(...(f_1(z_0))))))$. By using this identity, we are able to compute the expectation under a complicated distribution q_k without having to compute its density explicitly.

2.2.3 Experiments and Results

In this section we provide an overview of the data we used to evaluate our model. Afterwards, a series of benchmark methods are compared to the performance of the proposed model. These are the multi-layer perceptron (MLP) model, the convolutional neural network (CNN) model, the STGCN (S. Liu et al., 2021) model, and the VGRAN model (Kobyzev et al., 2021). In the past, these models have been used frequently in the literature for the analysis of spatio-temporal data. For the STGCN and VGRAN models, the dimensions of the hidden states are set as the number of nodes in the bus transportation network graph. The model is trained by minimizing the mean square error with a batch size of 32.

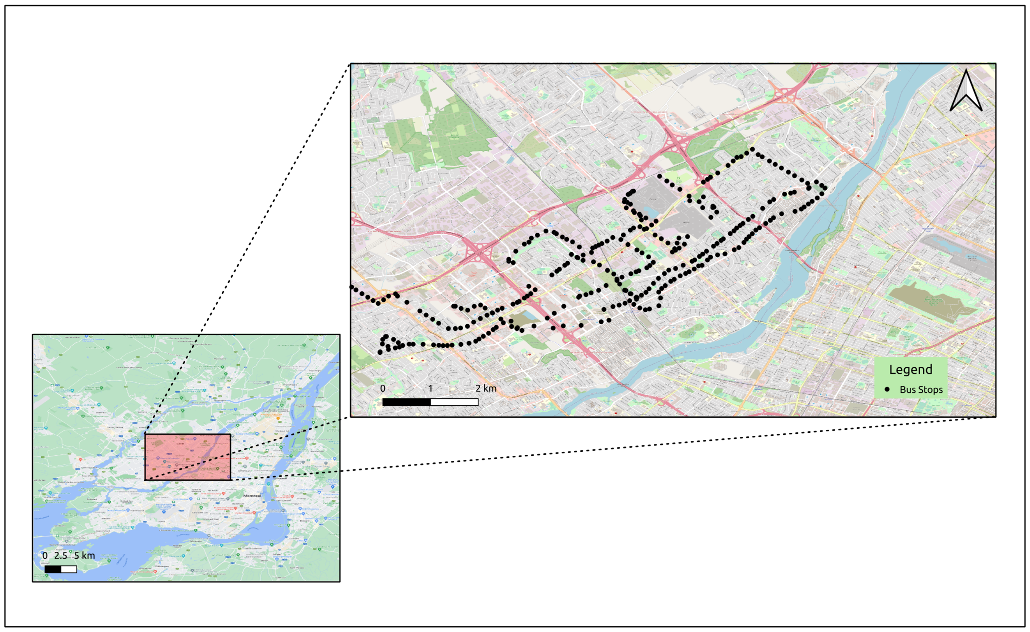


Figure 2.4: The network of bus stations and routes-Laval,Canada

Dateset

The data set used in this study was provided by the Société de transport de Laval (STL) in Laval, Canada, for the month of October 2021. The studied network can be seen in Figure 2.4.

Table 2.1 describes the dataset including the number of stops and the number of observations made on each bus route. In total, 410 stops are included in the dataset, with 325,484 passenger flow observations. It also describes the number of passengers, boardings and alightings on each route for the month.

Evaluation

To predict the passenger flow at each time step, data for 10 previous steps are fed as input to the model. Each interval is set to 10 minutes. Two commonly used metrics are considered to assess the performance of the model and the baseline methods in this study: (1) mean absolute error (MAE); and (2) root mean squared error (RMSE).

$$MAE = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n} \tag{29}$$

Table 2.1: Dataset statistics

Route No.	No. Of Stops	No. of Obs.	Boardings	Alightings off	Total Passengers (In all Buses)
20	56	29960	6774	6423	163255
24	48	51360	21679	20393	403593
26	62	138880	48681	47629	1101733
36	39	6903	3097	2846	80435
46	66	60258	14643	13928	441380
60	61	7596	7411	145318	3972000
66	52	35776	7240	6810	129330
730	12	1241	1937	1925	5136
744	14	1106	385	371	2720
Total	**410**	**325484**	**104436**	**100325**	**6299582**

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{30}$$

where $\hat{y}_i$ and y_i are the predicted and the true value, respectively. Estimating MAE and RMSE is based on a stop-level prediction and real-world values. According to the model, for each period in the test data set, the value of each stop will be predicted. Therefore, if n_{obs} is defined as the number of observations in the data, and n_{stops} is defined as the number of stops in the network, n will equal $n_{obs} \times n_{stops}$.

The accuracy of the model is another key factor to consider when comparing different methods. We classify the passenger flows into bins of size 5, and then determine the accuracy of the classification based on the results. This definition of accuracy considers deviations in the flow with a size smaller than 5 to be noise.

Experimental Results

Table 2.2 illustrates the results of our model in comparison to the baseline methods on the dataset, demonstrating the superiority of our model with regard to MAE, RMSE and both types of accuracy. The values in this table are calculated for all test observations based on the real and predicted values. As can be seen, the MAE and RMSE values for our model are respectively 0.372

and 1.301 which are less than other model values for these two metrics. The reason is likely related to the special way that the proposed model takes into account the spatial features, network topology, as well as temporal features.

Table 2.2: Performance comparison of different methods

Model	MAE	RMSE	Accuracy(%)	bins accuracy(%)
MLP	0.691	1.824	73.1	86.3
CNN	0.707	1.785	70.7	85.4
STGACN	0.474	1.427	78.7	94.1
VGRAN	0.410	1.393	80.2	94.8
VGConvNF	**0.372**	**1.301**	**83.4**	**96.2**

As can be seen in table 2.2 our model shows better accuracy compared to other models even with the bin size of 5 which is relatively small. Additionally, table 2.3 compares the performance of the models used in this article, only for peak hours. Compared to other models, the proposed model performs better on all three metrics.

Both tables 2.4 and 2.5 present the results for weekdays and weekends, demonstrating that the performance of various models is generally better on weekdays. It may be attributed to differences in the distribution of data between weekdays and weekends, including traffic volume, passenger demand, and bus schedules. Due to the higher number of bus trips on weekdays, the models may have access to a greater amount of training data, resulting in improved accuracy of predictions. In order to ensure accurate interpretation and practical application of the model, further investigation into the specific reasons for this performance difference is warranted.

Table 2.3: Performance comparison of different methods in peek hours

Model	MAE	RMSE	Accuracy(%)	bins accuracy(%)
MLP	0.734	2.203	68.1	81.1
CNN	0.769	2.347	66.9	79.9
STGACN	0.596	1.601	85.2	89.7
VGRAN	0.498	1.512	90.1	92.3
VGConvNF	**0.502**	**1.450**	**91.6**	**93.3**

The training efficiency of our model along with other deep learning methods in this study is

Table 2.4: Performance comparison of different methods on weekdays

Model	MAE	RMSE	Accuracy(%)	bins accuracy(%)
MLP	0.712	1.917	69.4	82.2
CNN	0.743	1.928	68.3	81.1
STGACN	0.531	1.507	81.2	87.4
VGRAN	0.417	1.410	83.4	89.8
VGConvNF	**0.398**	**1.317**	**92.6**	**93.9**

Table 2.5: Performance comparison of different methods on weekends

Model	MAE	RMSE	Accuracy(%)	bins accuracy(%)
MLP	0.747	2.104	67.2	79.2
CNN	0.754	1.921	66.3	78.9
STGACN	0.576	1.590	79.8	80.6
VGRAN	0.472	1.473	82.4	83.3
VGConvNF	**0.416**	**1.407**	**89.2**	**91.4**

also examined. Figure 2.5 shows the validation loss curves versus the number of training epochs for all models. In the training process, we use the early stopping mechanism, so that each model can have a different number of training epochs. As Figure 2.6 indicates, the loss function of our model decreases faster than the others. This model needs fewer epochs than STGACN to converge. However, the MLP and CNN models converged in fewer epochs.

Also, the comparison of computing time for all models is shown in Figure 2.6. We can see that the running time of each epoch for our model is longer than the others. The difference between running time of MLP model and CNN model is small. Since CNN convergence occurs in fewer epochs, the total running time of these two models is almost equal.

The same situation occurs with VGRAN and our model where our model converges faster than VGRAN but needs more training time for each epoch. For each epoch, the STGACN model needs half the time of ours. However, since it converges in almost twice as many epochs, the total running time of these two models is also close.

All of the models were implemented on the OMEN GT13-0090 30L Gaming PC with characteristics of NVIDIA GeForce RTXTM 3090, Intel Core i9-10850K, and HyperX 32 GB DDR4- 3200

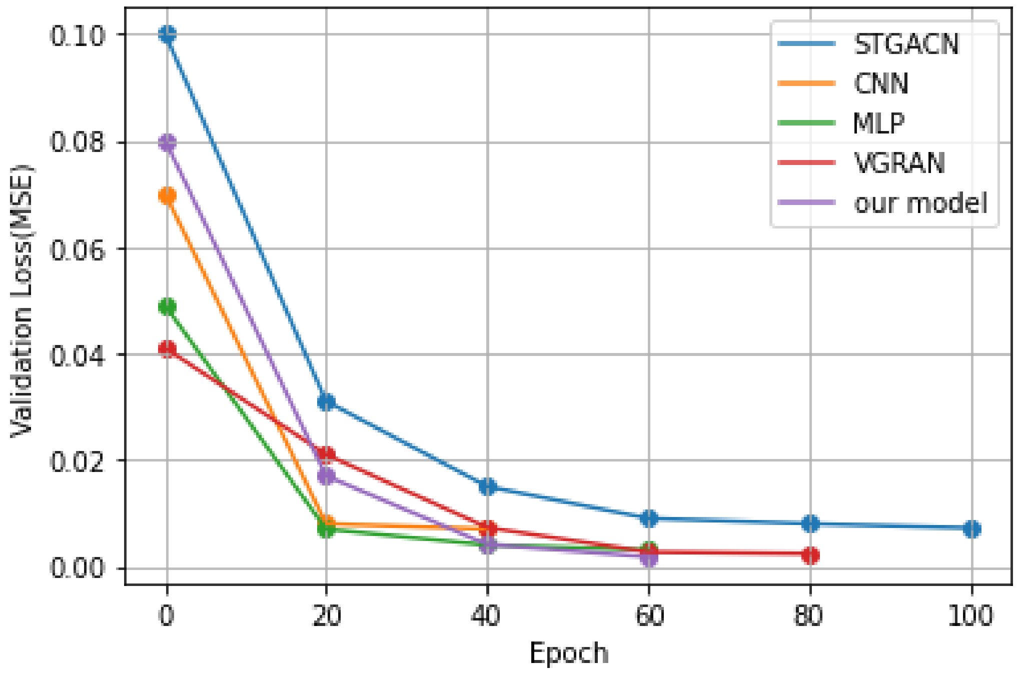


Figure 2.5: Validation loss versus training epoch

XMP MHz RAM (2 $\times$ 16 GB).

2.3 Conclusion

This chapter develops a novel deep learning approach for short-term prediction of passenger flow in a bus network. We achieve this by extending a GNN model to a variational autoencoder. We then use the procedure of normalizing flows to enhance the expressiveness of the posterior distribution of latent variables. In this way, not only are the nonlinear spatial and temporal correlations of passenger flow at different bus stops captured, but it is also done in a nondeterministic manner.

Moreover, based on actual data of the bus network of Laval, Canada, a case study is carried out to compare the proposed model with other prediction methods, that is, multi-layer perceptron (MLP) model, the convolutional neural network (CNN) model, Spatiotemporal Graph Convolution Network (STGACN) and variational graph neural network (VGRAN). The proposed model performs better than the other compared models on the measurement of mean absolute error (MAE) and root mean square error (RMSE). In addition we compare the accuracy for a scenario where

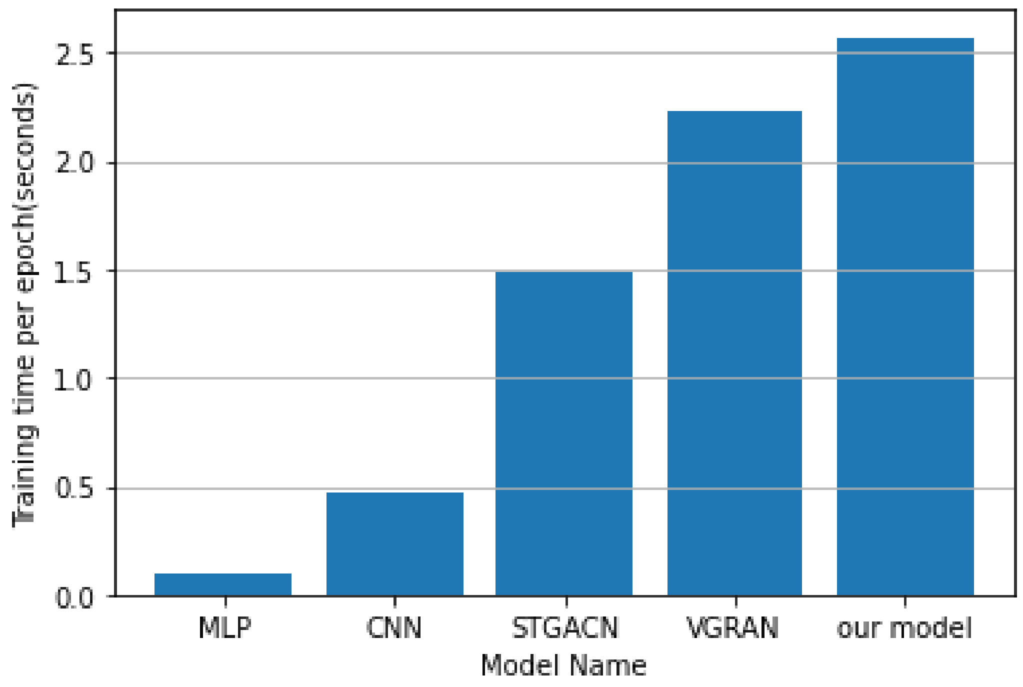


Figure 2.6: Comparing computation time between different models

passenger flow is classified into bins of size 5. Our model shows more accurate results than other models, even for a small bin size, demonstrating it can accurately predict many passenger flow variations.

Two conclusions were drawn as a result of this study. First, extending a GNN model to include stochastic node latent embeddings enhances its ability to capture spatio-temporal dependencies. Second, the accuracy of the prediction can be improved by using a more expressive GNN model as the core of the algorithm.

Chapter 3

A Reinforcement Learning Algorithm for ride-matching and vehicle routing [1]

On-demand transportation services offer a compelling alternative to traditional fixed-route and fixed-time transportation, particularly in suburban areas with sparse populations. These flexible systems have the potential to address transit challenges in such regions. A crucial component of these solutions are the ride-matching algorithms used to assign trip requests to available vehicles. In this study, we introduce a reinforcement learning (RL) model designed to optimize ride matching and vehicle routing in on-demand shared mobility services. Our approach, termed ROUTE-Ride, leverages a multi-agent rollout reinforcement learning algorithm to minimize waiting times, detour times, and total travel times, thereby enhancing both service quality and operational efficiency. The model was implemented and tested within the SUMO simulation environment using real-world Automatic Passenger Counting (APC) data from Laval, Canada, provided by the Société de transport de Laval (STL). Comparative analysis with traditional SUMO algorithms and a Q-network-based RL method demonstrated that ROUTE-Ride significantly outperforms these alternatives, achieving superior results across all key performance metrics. Specifically, ROUTE-Ride was able to reduce waiting times by over 20% compared to other methods. Additionally, distribution analyses showed ROUTE-Ride's effectiveness in reducing travel times, indicating higher efficiency. The findings

[1]This paper was submitted to the Transportation Research Board (TRB) and Transportation Research Record (TRR) in 2024.

suggest that ROUTE-Ride is particularly well-suited for applications in low population-density areas and for addressing first-mile transit challenges in both urban and suburban contexts.

3.1 Literature Review

The purpose of this section is to explore the literature surrounding on-demand transit in suburban areas, examining its usage and necessity. Our next step will be to explore various ride-matching algorithms, focusing specifically on RL. Moreover, we will discuss the gaps in current research that motivate our work.

3.1.1 On-demand transit for first-mile trips

Here we provide a concise review of the literature regarding the "first-mile problem," on-demand transit (ODT), and shared mobility services, emphasizing the critical role ride-matching algorithms play.

The first-mile problem in transportation refers to the connectivity challenges between transit stations and trip origins, which can discourage public transit use and impact access to essential services, particularly for underserved communities Kåresdotter, Page, Mörtberg, Näsström, and Kalantari (2022). This issue is especially pronounced in low-density suburban areas where traditional fixed-route transit systems are inefficient Alsaleh and Farooq (2023); Meshkani et al. (2024). To address first-mile barriers, various solutions have been proposed, including shared mobility services and on-demand transportation van Kuijk, de Almeida Correia, van Oort, and Van Arem (2022).

As a modern approach to public transportation, on-demand services eliminate fixed routes, schedules, or both Vansteenwegen et al. (2022). The flexibility comes at a price: the need to collect more detailed information from each passenger. Various methods can be used to collect this data, such as direct transportation requests via mobile devices Demissie et al. (2016), smart stops Meshkani et al. (2024), or predictive analytics based on historical and real-time informationVansteenwegen et al. (2022).

Various types of on-demand services have been extensively explored in scientific literature, each differing based on several criteria Vansteenwegen et al. (2022). Firstly, services are categorized by the number of pickup and drop-off locations: many-to-many systems consider multiple origins and destinations Alsaleh and Farooq (2023), while many-to-one systems focus on multiple origins converging to a single destination Bürstlein et al. (2021). Secondly, the degree of system flexibility varies, with fully flexible systems lacking a standard route or timetable Vallée, Oulamara, and Cherif-Khettaf (2017); Yoon et al. (2022), and semi-flexible systems adhering to a standard route or timetable but allowing dynamic deviations Pratelli, Lupi, Farina, and Pratelli (2018). Thirdly, different optimization objectives have been proposed, targeting either the transportation agency or drivers (e.g., maximizing driver profit), the passengers (e.g., minimizing waiting and detour times), or both Alonso-Mora, Samaranayake, Wallar, Frazzoli, and Rus (2017); Vansteenwegen et al. (2022). Lastly, these services differ in simulation details, such as the study network (grid-like Calabrò et al. (2022) or real-world networks Meshkani et al. (2024)) and the type of data used (simulated Meshkani et al. (2024) or real-world data Inturri et al. (2021)).

Numerous studies have investigated the efficiency of on-demand services compared to fixed-route and semi-flexible transit systems Alonso-Mora et al. (2017); Inturri et al. (2021); Yoon et al. (2022). This work has primarily focused on evaluating service quality for passengers, often measured in terms of waiting time, as well as the operational costs for transit agencies. While on-demand services can offer superior service quality by minimizing passenger waiting times through dynamic routing and scheduling, they are not necessarily better in all situationsAlsaleh and Farooq (2023). The efficiency of on-demand services depends on various factors, including demand density, operational strategies, and the ride-matching algorithm used. By comparing these transit models, researchers aim to identify the optimal balance between passenger satisfaction and operational efficiency, providing valuable insights for transit agencies to enhance their service offerings under different conditions.

On-demand transit frameworks are based on ride-matching algorithms, which dictate how vehicles are allocated to passengers in real time. In the context of the first-mile transit problem, where efficiency is paramount, the ride-matching algorithm becomes even more crucial. Traditional taxi services often rely on manual ride-matching methods or simplistic algorithms, which may suffice for steady demand patterns but falter when faced with dynamic scenarios. As a result of this discrepancy in efficiency, the ride-matching algorithm has been deemed one of the most crucial aspects of on-demand transit systems. Although Dynamic Real-Time Scheduling and Dispatching (DRST) models can meet the needs of high-traffic scenarios, their effectiveness diminishes under resource constraints or low demand. DRST systems must therefore be designed with an optimized ride-matching algorithm to bridge this efficiency gap and ensure their viability.

3.1.2 Ride-matching algorithm

Maintaining equilibrium between demand and supply is the key to a successful ride-matching strategy. Optimizing a platform's efficiency requires proactive deployment of vehicles to areas with large gaps in demand. In this regard, different approaches have been tested: Song and Earl (2008) formulated the problem as a multi-objective optimization problem considering uncertainty in vehicle arrival time. Nair and Miller-Hooks (2011) Developed a two-stage stochastic mixed-integer and then minimised the cost. He, Hu, and Zhang (2020) formulated the problem as a stochastic dynamic program, using distributionally robust optimization to account for temporal dependence.

Decisions regarding matching and environmental factors, such as supply-demand conditions at different spatial and temporal scales, exhibit extensive interrelationships. The application of a Markov Decision Process (MDP) framework would therefore be a viable solution in modeling these operations. As a result of employing RL techniques within this framework, effective decision-making policies can be developed and optimized. For the purpose of recognizing and adapting to these intricate interactions, existing research has used a variety of RL algorithms (Bertsekas (2021)).

Both model-based and model-free approaches have been used in the context of ride matching and vehicle routing Bertsekas (2021). To solve the MDP, model-based approaches tend to use value iteration techniques, helping to optimize decision-making processes based on structured representations of systems Shou and Di (2020); Zhou et al. (2019). In a similar manner, X. Yu and Shen (2019) uses matrix operations to accelerate computation. Other researchers have adopted model-free approaches. Rather than explicitly modelling behaviors like cruising, ride-matching, and matching, these studies learn directly from agent interactions. The Monte Carlo method is one example of this approach Verma, Varakantham, Kraus, and Lau (2017). Additionally, Q-learning algorithms and their variants, such as Deep Q-Networks (DQN), also exemplify model-free approaches Gao, Jiang, and Xu (2018); Wen, Zhao, and Jaillet (2017).

The rollout algorithm is another RL approach, initially tested in training agents to play chess Silver et al. (2017). Rollout is a policy iteration (PI) algorithm Bertsekas (2021). By using a base policy to guide the simulation, the rollout algorithm improves decision-making by iteratively simulating the outcomes of different actions. By refining the base policy at each step, the rollout algorithm enhances its performance, making it an effective tool in complex and dynamic environments. Rollout has been proposed in the context of the autonomous routing and pickup problem Garces, Bhattacharya, Gil, and Bertsekas (2023), however, to the best of our knowledge, this is the first time that it has been used in the context of vehicle matching and routing in an on-demand shared mobility environment.

3.1.3 Discussion

In summary, the scientific literature highlights two critical aspects for advancing RL research: the necessity of addressing time complexity and performance optimization, and the importance of platform flexibility.

Efficient time complexity and robust performance are essential for the practical implementation of RL models, ensuring that they can handle complex tasks within reasonable time frames. Equally important is the flexibility of the platforms used, as the rapid progression of RL models demands a testing environment that can easily accommodate and evaluate new algorithms as they emerge. This dual focus on computational efficiency and adaptability is fundamental to driving innovation and practical application in the field.

While previous research, such as Garces et al. (2023), has used an algorithm using the rollout approach in the context of automated vehicle routing and employed an explicit distribution for demand that is adjusted over time, we present a modified version of rollout in our algorithm for optimizing on-demand shared mobility services without assuming any demand distribution. Unlike their work, which tested the algorithm within a constrained $400 \times 400m^2$ area, we applied our approach using APC data for the entire city of Laval. Further The expansion of Rollout and Policy Iteration testing to more complex, real-world networks provides valuable insights and validates its effectiveness. Our model is uniquely designed for on-demand service contexts and has been rigorously tested using actual APC data from Laval. Moreover, our implementation in the SUMO simulation environment using TraCI ensures a unified approach to both algorithm development and simulation, enhancing the robustness and reliability of our results.

3.2 Methodology

In this study, we propose an RL approach to address the vehicle matching problem. Our model leverages the SUMO (Simulation of Urban MObility) framework for realistic simulation and testing. By integrating RL techniques with SUMO, we aim to optimize vehicle assignment in a dynamic transportation network. We begin by formulating the problem and then proceed to describe our RL-based solution. The methodology encompasses environment setup, agent design, and training procedures. Our approach aims to enhance efficiency, reduce congestion, and improve overall traffic flow through intelligent vehicle matching. We aim to enhance efficiency while maintaining low complexity, making it suitable for real-time systems.

3.2.1 The inspiration behind the rollout Idea

The rollout algorithm is a powerful technique that is used in RL in order to improve decision-making policies. As a core principle, rollout uses multiple simulated trajectories to evaluate and improve the current policy. It is possible to estimate the long-term benefits of different actions by simulating their outcomes in a given state in order to make more informed decisions.

Using rollout in the context of our on-demand shared mobility service allows us to manage the inherent uncertainty and dynamic nature of real-time transportation environments. Here is a step-by-step explanation of how the rollout process works:

- **Policy Evaluation:** At each decision point (e.g., dispatching a vehicle), the current policy suggests a set of actions based on the present state. In Figure 3.1, the central node represents the current agent, and the arrows indicate possible actions.

- **Simulation of Trajectories:** Several possible future trajectories are simulated for each action suggested by the current policy. By considering various possible future states and events, these simulations predict how the system might evolve if a particular action were taken. As can be seen in Figure 3.1, each action leads to a simulated final state.

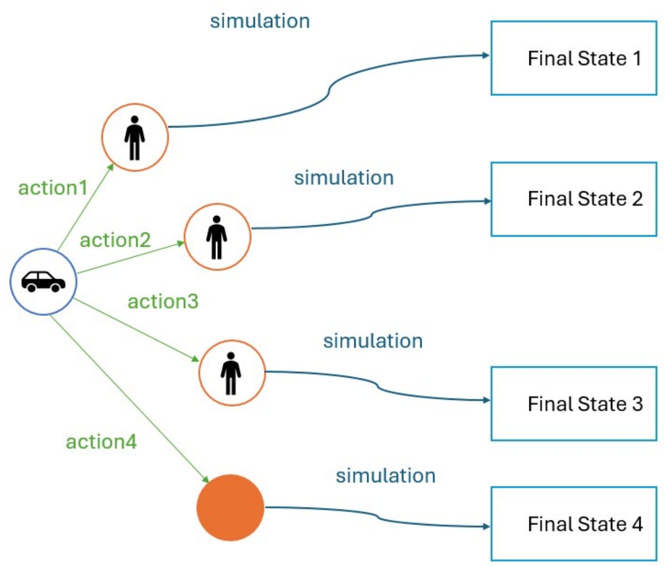


Figure 3.1: Illustration of the Rollout Process in On-Demand Shared Mobility. The central node represents the current state with potential actions leading to simulated final states. Each action is evaluated through simulations to estimate the cumulative rewards, guiding the selection of the optimal action to improve the policy iteratively. This process helps manage real-time transportation environments by optimizing decisions based on predicted outcomes. The orange circle indicates that there is no action.

- **Comparison of Rewards:** Based on the simulated trajectory, a cumulative reward estimate can be derived for each action. The comparison of these rewards allows us to determine which actions are likely to yield the best long-term results. As shown in Figure 3.1, the final states represent potential outcomes of each action, which are analyzed based upon their rewards.

- **Policy Improvement:** The action with the highest estimated reward is selected as the optimal decision at the present time. On the basis of the simulated experiences, the policy is then updated and refined. A new policy is calculated by looking ahead at potential outcomes, a process known as lookahead minimization.

- **Policy Iteration:** Iteratively, this process of policy evaluation and improvement allows the policy to gradually become more effective at making decisions that optimize the overall performance of the mobility service. As a result of policy iteration, each new policy performs better than the previous one.

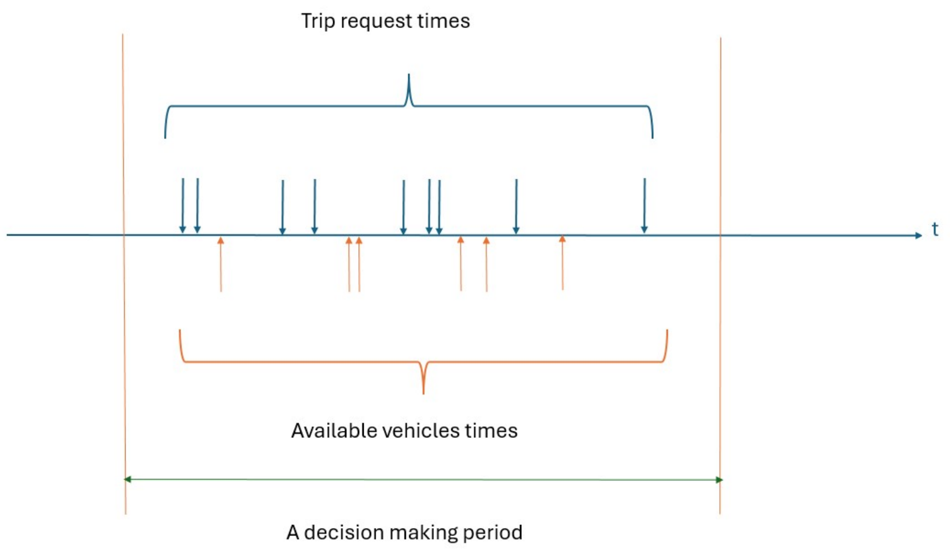


Figure 3.2: Timeline illustrating trip request times, available vehicle times, and a decision-making period.

Our primary training phase is conducted offline. A neural network is trained during this phase in order to approximate the cumulative cost from each state. The trained neural network is then used in real-time to determine the action with the lowest cost at each decision point for each vehicle. In the online simulation, this network is used to estimate the cost associated with each action.

3.2.2 Problem formulation

This study presents a formulation for addressing the problem of vehicle-to-request matching. The problem is characterized by discrete time and finite horizon stochastic dynamic programming (DP). In the following subsections, we will discuss the environment, the network, requests, agents, state space and action space.

Simulation setup

For modeling and testing the proposed RL model, we use the open-source, microscopic, and continuous traffic simulation package SUMO (Simulation of Urban MObility *SUMO* (2023)).

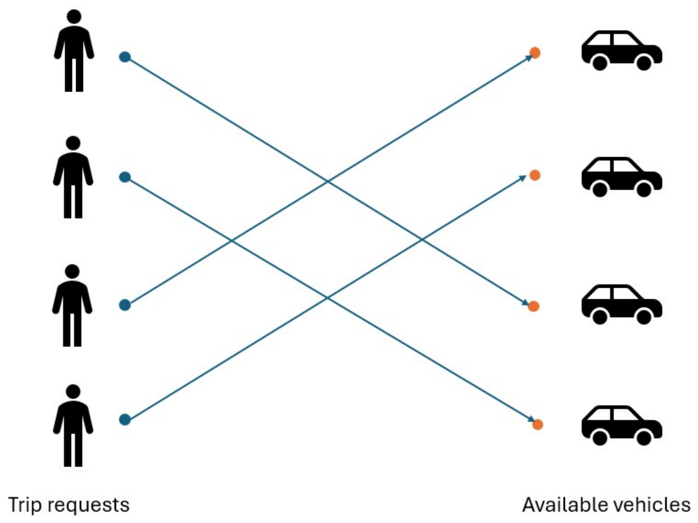


Figure 3.3: Illustration of the assignment of trip requests to available vehicles.

Study area

The city of Laval is located in southwest Quebec, north of Montreal, in the province of Quebec. It is the largest suburb of Montreal and the third-largest city in Quebec, following Montreal and Quebec City. According to the 2023 estimates, Laval had a population of 451,986, covering an area of 246.1 km^2, and a population density of 1,836 residents per km^2. The Société de transport de Laval (STL) is the primary public transportation provider, operating an extensive network of bus routes and shared taxi services throughout the city. Figure 3.6 shows map of Laval highlighting bus stops and the Cartier Metro Station, a primary public transit hub for commuters traveling to Montreal. In this study, Cartier is used as the sole destination for all ride requests.

Network

We assume that vehicles and trip requests operate within urban environments characterized by a fixed street topology, represented as a directed graph $G = (V, E)$. In this graph, $V = \{1, \ldots, n\}$ denotes the set of indices for street intersections within the city map, encompassing a total of n intersections. The set $E \subseteq \{(i, j)|i, j \in V\}$ represents the directed streets connecting intersection i

to intersection j.

Requests

A request is represented as tuple $r = (o_r, d_r, t_r, s_r)$, where $o_r \in V$ and $d_r \in V$ corresponds to the pick-off and drop-off locations of the trip request. t_r is time at which the request is made; s_r is the request's status, which can be unassigned, assigned, picked up, or fulfilled.

As shown in Figures 3.2 and 3.3, requests arrive at random times and in each fixed time period the requests and available vehicles are aggregated. Figure 3.2 illustrates the timeline of trip request times, available vehicle times, and the decision-making period. The black arrows represent the times at which trip requests are made, while the orange arrows indicate when vehicles are available. The decision-making period encompasses the interval during which decisions about trip assignments are made. Figure 3.3 shows the process of matching trip requests to available vehicles. The left side depicts individuals making trip requests, while the right side shows available vehicles. The lines connecting the individuals to the vehicles represent the matching process. This aggregation and matching process ensures that trip requests are efficiently assigned to available vehicles within each decision-making period.

Agents, state space and action space

As part of this environment, a transit user initiates a trip request, typically through a smartphone app, specifying their departure and destination. Upon receiving the request, the system processes this information and coordinates with the available vehicles to meet the user's travel needs. As an agent, each vehicle navigates through the environment to pick up and drop off users, aiming to minimize travel time and maximize occupancy. Users and operators interact seamlessly, with real-time updates on vehicle locations and request statuses, ensuring that users are informed and can rely on the system to manage their travel efficiently. As a result of this dynamic interaction, the simulation's agents, state space, and action space are defined.

We define an agent for each vehicle operating within the environment. We assume every agent has the ability to perfectly observe requests, every agent's location, and their occupancy status at all times which means all the agents share the the same state at each time. The state at time t

is $x_t = (v_t, \tau_t)$ where $v_t = [(v_{1t}, c_{1t}), (v_{2t}, c_{2t}), ..., (v_{mt}, c_{mt})]$ is a list of all vehicle locations in the graph and their occupancy; and $\tau_t = [r_{1t}, r_{2t}, ..., r_{nt}]$ is a list of all the requests in the simulation. Each two consecutive decisions are separated by 15 seconds. The action space includes 21 possible actions: remaining idle or picking up one of up to 20 available trip requests. We employ a standard one-step lookahead rollout algorithm, a policy iteration method where the base policy π guides decision-making. Starting from the initial state x_0, the agent generates a trajectory by selecting actions $\tilde{a}_t$ through one-step lookahead minimization, optimizing for the immediate and future expected costs. Since this formulation represents a separable action constraint for each agent, the controls available to all agents at time t, $A_t(x_t)$, are expressed as the Cartesian product of local control sets $A_t^1(x_t) \times \cdots \times A_t^m(x_t)$. Figure 3.4 depicts the environment with requests and vehicles and how the cloud receives and sends information.

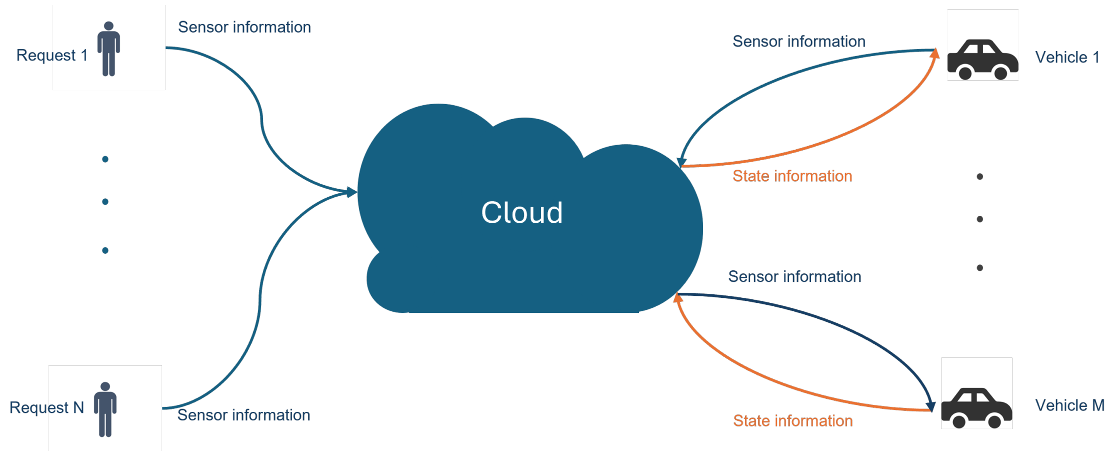


Figure 3.4: Agents (vehicles) and trip requests connected to the cloud for real-time decision-making.

Stochastic dynamic programming framework

RL is based on the Markov Decision Process (MDP) framework, whereby an agent, the decision-making entity, operates within a defined state space S. In the stochastic case, $\pi(a|s)$ denotes the probability of selecting action a given state s. We consider policies of the form $\pi = \{\mu_1, \ldots, \mu_T\}$, where μ_t maps states x_t into actions $a_t = \mu_t(x_t)$, and satisfies a constraint of the form $\mu_k(x_t) \in U_t(x_t)$ for all x_t. Given an initial state x_1 and a policy $\pi = \{\mu_1, \ldots, \mu_T\}$, the expected cost of π starting from x_1 is:

$$J^{\pi}(x_1) = \mathbb{E}\left\{ g_T(x_T) + \sum_{t=1}^{T} g_t(x_t, \mu_t(x_t), w_t) \right\} \tag{31}$$

where the expected value operation $\mathbb{E}\{\cdot\}$ is with respect to the joint distribution of all the random variables w_t and x_t. The optimal cost starting from x_1 is defined by:

$$J^*(x_1) = \min_{\pi \in \Pi} J^{\pi}(x_1) \tag{32}$$

where Π is the set of all policies. An optimal policy π^* is one that attains the minimal cost for every x_1; i.e.,

$$J^{\pi^*}(x_1) = \min_{\pi \in \Pi} J^{\pi}(x_1), \quad \text{for all } x_1. \tag{33}$$

The size of the state space at a time t, with m available vehicles, is $O(|V|^m(|V| \times |V|)^{|\tau|})$ since each available vehicle can be located at any of the $|V|$ locations and there may be $|V| \times |V|$ possible pickup-dropoff location pairs for each request. Given the complexity, finding an optimal policy for such a large multiagent vehicle routing problem is intractable. Therefore, we focus on finding suboptimal solutions that can be computed more efficiently.

As shown in Figure 3.5 , the state transition process is illustrated. For our rollout RL algorithm the cost function g_k should effectively capture the objectives of minimizing total waiting time and the number of unanswered requests.

The formulation for the cost function is as follows:

$$g_k(x_t, \mu_k(x_t), w_t) = \alpha \cdot \text{waiting\_time}(x_t, \mu_k(x_t), w_t) \quad + \beta \cdot \text{unanswered\_requests}(x_t, \mu_k(x_t), w_t) \tag{34}$$

Where waiting_time$(x_t, \mu_k(x_t), w_t)$ represents the waiting time of the requests at step t. This is computed as the sum of the waiting times for all requests during the step; specifically, the waiting time for each request is the time difference between when the request is made and when it is picked up. unanswered_requests$(x_t, \mu_k(x_t), w_t)$ represents the number of unanswered requests at step t. This can be a binary indicator that takes the value 1 if a request is not assigned yet, and 0 otherwise. α and β are weighting parameters that balance the trade-off between minimizing waiting time and minimizing the number of unanswered requests. As a result, this function penalizes both served requests and unanswered requests, aligning the goal of minimizing total waiting time and unanswered requests.

Simulation in the SUMO environment is used to transition between states, as shown in Figure 3.5. By using this setup, traffic dynamics and vehicle interactions can be modeled in detail, providing a realistic context in which to evaluate the performance of the different ride-matching algorithms and policies. The implementation uses the Traffic Control Interface (TraCI), which "allows retrieving values of simulated objects and manipulating their behavior online." *TraCI - SUMO Documentation* (2024). This capability enables dynamic interaction with the simulation, allowing for real-time adjustments and more accurate assessments of the algorithm's effectiveness. Each step's cost will be calculated at the end of the process.

3.3 Experiments and Results

In this section we provide an overview of the data we used to evaluate our model. Afterwards, a series of benchmark methods are compared to the performance of the proposed model. The dataset used in this study is Automatic Passenger Counting (APC) data provided by the Société de Transport de Laval (STL) in Laval, Canada, for the months of October and November of 2022. In Figure 3.6, a map of Laval is presented, highlighting its bus stops and the Cartier Metro Station, which serves as a key public transit hub for commuters traveling to Montreal. In this paper, we use Cartier as the single destination for all requests.

In order to measure the performance of our model we compared it with several other methods. Different ride-matching algorithms are available in SUMO, tailored to different spatial and

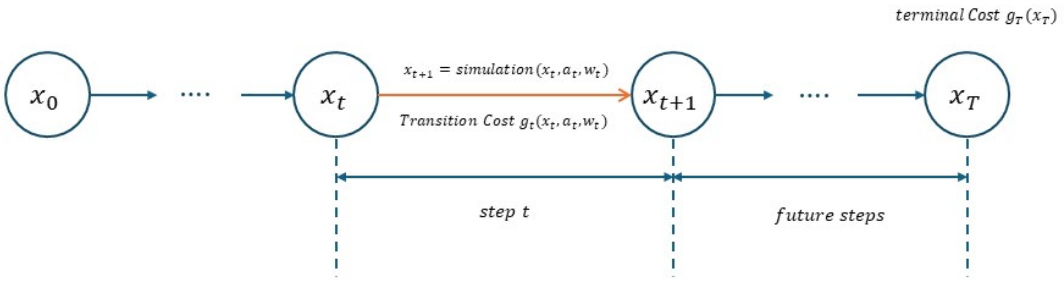

terminal Cost $g_T(x_T)$
x_0
x_t
$x_{t+1} = simulation(x_t, a_t, w_t)$
Transition Cost $g_t(x_t, a_t, w_t)$
x_{t+1}
x_T
step t
future steps

Figure 3.5: Illustration of the T-step stochastic optimal control problem: From an initial state x_t, the system evolves to the next state under the influence of action a_t

temporal scenarios. Among the ride-matching algorithms we tested were greedy, greedyClosest, greedyShared and routeExtension. As a result of the ‘greedy‘ algorithm, taxis are assigned to customers in the order of their reservations, choosing the taxi that is closest to them in terms of travel time; if the reservation date is too far in the future, the reservation is postponed *SUMO Taxi Simulation* (2023). On the other hand, the ‘greedyClosest‘ algorithm assigns a taxi to the closest customer, delaying reservations made in the future *SUMO Taxi Simulation* (2023). The ‘greedyShared‘ algorithm operates similarly to ‘greedy‘, except that it allows taxis to pick up additional passengers while delivering the first passenger to their destination *SUMO Taxi Simulation* (2023). Finally, the ‘routeExtension‘ algorithm is similar to greedy, but allows taxis to pick up any passenger along their route as long as the vehicle has unused capacity *SUMO Taxi Simulation* (2023). We also included an RL method without the rollout part, called Q-network.

Tables 3.2 and 3.1 provides a comparison of various algorithms in terms of waiting time, detour time, and total travel time during peak hours (6am to 8am) and a 24 hour period respectively. The “greedy“ algorithm shows the highest waiting time, detour time, and total travel time. “greedyClosest“, “greedyShared“, and “routeExtension“ algorithms perform slightly better, which is in line with our expectations since these methods allow for a single vehicle to have more than one passenger at

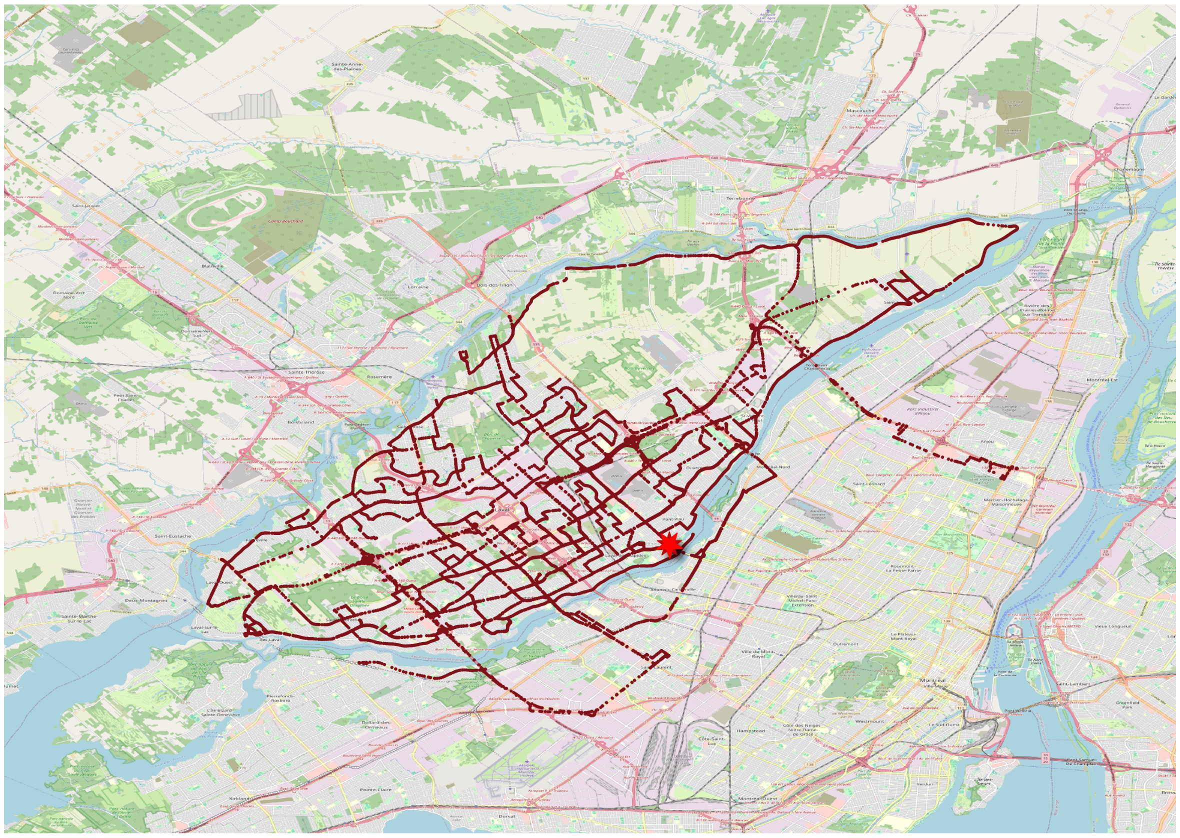


Figure 3.6: Map of Laval highlighting bus stops and the Cartier Metro Station, which serves as a primary public transit hub for commuters to Montreal. In this study, Cartier is used as the single destination for all ride requests.

a time.

Thanks to the RL foundations, Q-network and ROUTE-Ride demonstrate superior performance compared to the other algorithms. As a result of the RL agent's ability to capture more complex interrelationships in the data, these two perform better. Detour time and total travel time are significantly reduced as a result of the ability to learn from and adapt to dynamic environments. ROUTE-Ride further enhances performance by incorporating the rollout idea, which simulates different actions iteratively to improve decision-making. Using this approach, ROUTE-Ride is able to reduce wait times and travel times even further, underlining its effectiveness in optimizing ride-matching and vehicle routing.

Figures 3.7a and 3.7b comparing the total travel time distributions further highlight the advantages of ROUTE-Ride. In Figure 3.7a, we compare the distribution of total travel times between

Table 3.1: Comparison of Algorithms in Terms of Waiting Time, Detour Time, and Total Travel Time for 24 hours

Algorithm	**Waiting Time (min)**	**Detour Time (min)**	**Total Travel Time (min)**
greedy	9.01	25.02	32.65
greedyClosest	8.78	24.92	32.14
greedyShared	8.32	23.29	27.87
routeExtension	8.26	23.32	27.83
Q-network	8.97	16.35	25.14
ROUTE-Ride	**7.37**	**14.11**	**21.12**

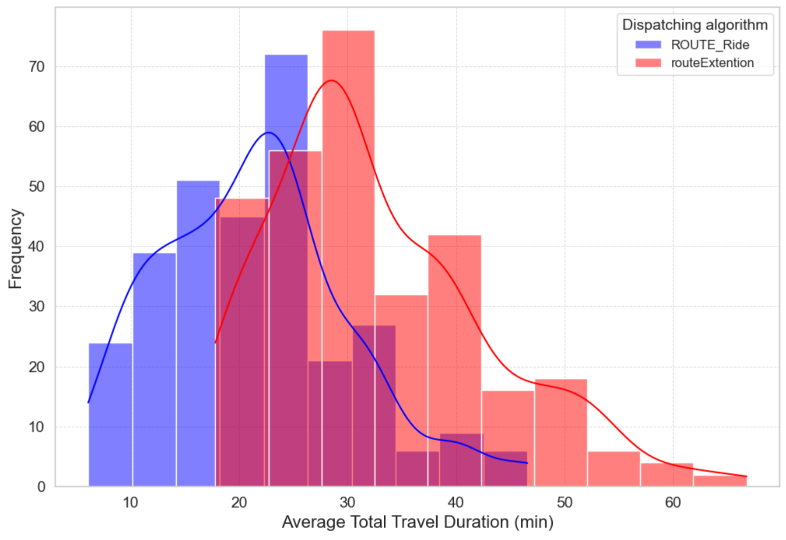
Table 3.2: Comparison of Algorithms in Terms of Waiting Time, Detour Time, and Total Travel Time for peak hours (6am to 8am)

Algorithm	**Waiting Time (min)**	**Detour Time (min)**	**Total Travel Time (min)**
greedy	9.70	26.12	33.63
greedyClosest	9.12	25.12	32.14
greedyShared	8.93	24.34	31.87
routeExtension	9.33	23.98	31.23
Q-network	9.16	17.23	26.47
ROUTE-Ride	**7.47**	**14.11**	**23.1**

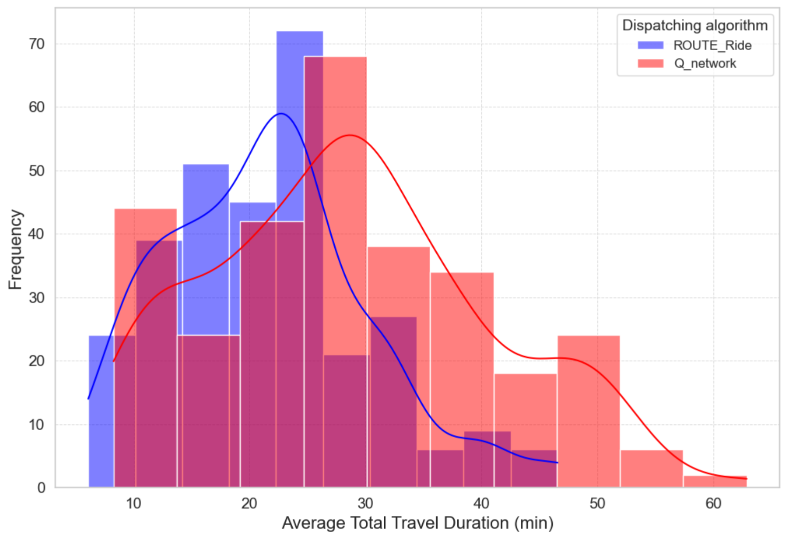
ROUTE-Ride and routeExtension, the best built-in algorithm from SUMO. ROUTE-Ride's distribution is more skewed to the left, indicating more efficient travel times. Figure 3.7b compares ROUTE-Ride with the Q-network, demonstrating again that ROUTE-Ride's distribution is more skewed to the left, aligning with our previous discussions about its superior performance. The visual comparisons show that the ROUTE-Ride model minimizes travel times better than other methods, validating its effectiveness.

3.4 Conclusion

This study introduced a new RL model to optimize service quality and operational efficiency in an on-demand shared mobility service that minimizes waiting time. Our model is based on the idea of policy iteration, specifically utilizing a multi-agent rollout RL algorithm. This approach not only captures complex interrelationships within the data but also iteratively refines decision-making through simulated outcomes, leading to enhanced performance in dynamic and real-time environments. The novelty of our algorithm lies in its innovative application of rollout in the context



(a) Comparison of total travel time for ROUTERide and routeExtension algorithms



(b) Comparison of total travel time for ROUTERide and Q-network algorithms

Figure 3.7: Comparisons of total travel time for different algorithms

of on-demand shared mobility to address the first-mile problem. By carefully designing the state-space and action-space, we were able to use rollout in a computationally efficient manner.

To simulate the proposed model, we implemented it in the SUMO simulation environment using

TraCI. Laval, a suburb of Montreal, Canada, was chosen as the study area due to its representative suburban characteristics and availability of detailed APC data. To assess the performance of the proposed matching and routing model, it was compared with the built-in SUMO algorithms as well as the Q-network, a non-rollout RL method. For training and testing the model, real-world data from the Société de transport de Laval (STL) was used, covering the months of October and November 2022. Different indicators were measured, including passenger wait time, total travel time, and detour time.

This comprehensive comparison allowed us to evaluate the effectiveness of the ROUTE-Ride model in reducing waiting times, detour times, and total travel times, demonstrating its superiority in optimizing service quality and operational efficiency. The results showed that the ROUTE-Ride model consistently outperformed both the built-in SUMO algorithms and the Q-network method. By reducing average waiting times by 20%, ROUTE-Ride demonstrated its ability to provide quicker service to passengers.

Additionally, the model minimized detour times, ensuring more direct routes and reducing overall travel durations. The histogram plots of total travel time distributions further reinforced these findings, with ROUTE-Ride displaying a more left-skewed distribution, indicating a higher frequency of shorter travel times. These improvements not only enhance passenger satisfaction but also suggest potential cost savings for transit agencies through more efficient routing and vehicle utilization. More specifically, ROUTE-Ride allows for quicker trips by reducing unnecessary detours. It is achieved by rolling out multiple actions during the online learning phase and using a cost approximation function that has been trained separately. The result is a shorter and more predictable travel time, which enhances passenger satisfaction. Furthermore, more efficient routing and vehicle utilization can reduce travel durations and fuel consumption for transit agencies, resulting in savings for the agency.

We think this study will significantly contribute to the design and operation of on-demand transit services in low population-density areas. Furthermore, the proposed service model holds potential for addressing the first-mile problem in both urban and suburban settings, particularly where the number of destinations is limited. By optimizing waiting times, detour times, and overall travel times, our approach can enhance the efficiency and user satisfaction of transit systems, providing a practical solution for improving accessibility and connectivity in various geographic contexts.

Future research could take several directions. To further improve the model's accuracy and responsiveness, additional real-time data sources, such as traffic conditions and passenger demand patterns, could be incorporated. It would also be worth exploring the generalizability and scalability of the ROUTE-Ride model in different urban environments and transportation networks. Additionally, investigating the impact of varying fleet sizes and vehicle types on system performance could provide valuable insights for optimizing resource allocation. Finally, combining the rollout algorithm with other advanced ML techniques may yield even greater improvements in service quality and operational efficiency.

Chapter 4

Conclusion and Future Direction

This thesis has explored innovative AI-driven approaches to enhance the efficiency and sustainability of public transportation within the context of smart cities. By leveraging deep learning and reinforcement learning techniques, this research addresses critical challenges in passenger flow prediction and on-demand ride-matching and vehicle routing, specifically within the public transportation systems of suburban areas like Laval, Quebec.

The first part of this research introduces a novel Graph Neural Network (GNN)-based model with probabilistic node embeddings for passenger flow prediction, addressing the limitations of the static node embeddings commonly employed in existing models. This innovative approach captures the inherent uncertainties and dynamic behaviors associated with passenger flows at various bus stops, leading to a significant improvement in prediction accuracy. The incorporation of normalizing flows further refines the model's posterior distribution, enhancing its precision and adaptability. This advancement optimizes resource allocation for transit agencies and contributes to a more balanced and responsive transit system, better equipped to meet the evolving demands of urban mobility.

The second part of the thesis presents a reinforcement learning algorithm tailored for ride-matching and vehicle routing in on-demand shared mobility services. This RL approach demonstrates superior performance over traditional methods by dynamically optimizing routes and matching strategies based on real-time conditions. Through rigorous testing using real-world APC data from Laval and implementation in the SUMO simulation environment, the proposed algorithm shows a significant potential to improve the operational efficiency and service quality of on-demand transit, particularly in low-density suburban areas. This research underscores the critical role of RL in maximizing long-term rewards in complex, real-world scenarios, highlighting its adaptability and effectiveness in optimizing public transportation networks.

Despite the promising results of the proposed RL algorithm, its primary weakness is its high computational resource requirement. As a result of its complexity and dependency on real-time data processing, the model can be limited in its scalability, especially when applied to larger networks or more densely populated areas. It is also challenging to optimize the model in order to make it less resource-intensive, since reducing the computational burden or simplifying the architecture may compromise the accuracy and performance of the algorithm. The model should be refined in the future in order to balance resource efficiency with the need for high-quality decision-making, ensuring that it can be used in a wider range of resource-constrained environments.

Together, these contributions provide a robust framework for advancing the role of AI in smart public transportation systems. By addressing key challenges in passenger flow prediction and on-demand service optimization, this thesis lays the groundwork for future research and development aimed at making urban mobility more efficient, sustainable, and user-friendly. The findings underscore the transformative potential of AI technologies in public transportation and support the broader vision of smart cities where innovative, data-driven solutions enhance the quality of life for residents and visitors alike.

Future research can build on these results by exploring additional deep learning architectures, expanding the application of RL algorithms to other aspects of public transit systems, and integrating more diverse datasets to further refine and validate the models presented. By continuing to push the boundaries of AI in public transportation, there is significant potential to drive further improvements in efficiency, accessibility, and environmental sustainability, ultimately contributing to the realization of smarter, more connected cities.

Future research could also explore the integration of the passenger flow prediction model and the ride-matching and vehicle routing algorithm into a unified framework. This combined approach would enable a more comprehensive and cohesive optimization of public transportation systems, where predicted passenger flows directly inform the dynamic decision-making processes of on-demand services. By aligning resource allocation with real-time demand forecasts, such a unified system could further enhance service responsiveness, reduce operational costs, and improve the overall passenger experience. This integrated framework would capitalize on the strengths of both predictive analytics and adaptive routing algorithms, creating a synergistic effect that addresses the complex interplay between passenger demand and service delivery. For instance, accurate passenger flow predictions could be used to proactively adjust vehicle deployment, match passengers more effectively, and refine routing strategies in real time, thereby maximizing operational efficiency and minimizing passenger wait times.

Developing such a unified model poses exciting challenges and opportunities, including the need to handle large-scale, high-dimensional data, and the requirement for advanced algorithms that can process information swiftly and make decisions under uncertainty. Future work could involve exploring new deep learning architectures, enhancing RL algorithms to better incorporate predictive data, and testing the integrated framework in diverse urban and suburban environments. This research direction not only promises to advance the state of AI in public transportation but also represents a significant step towards realizing fully autonomous and intelligent transit systems that are at the core of the smart city vision.

References

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (icet)* (p. 1-6). doi: 10.1109/ICEngTechnol.2017.8308186

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., & Rus, D. (2017). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, *114*(3), 462–467.

Alsaleh, N., & Farooq, B. (2023). Sustainability analysis framework for on-demand public transit systems. *arXiv preprint arXiv:2303.06007*.

Banerjee, N., Morton, A., & Akartunalı, K. (2020). Passenger demand forecasting in scheduled transportation. *European Journal of Operational Research*, *286*(3), 797–810.

Bertsekas, D. (2021). Multiagent reinforcement learning: Rollout and policy iteration. *IEEE/CAA Journal of Automatica Sinica*, *8*(2), 249–272.

Bürstlein, J., López, D., & Farooq, B. (2021). Exploring first-mile on-demand transit solutions for north american suburbia: A case study of markham, canada. *Transportation Research Part A: Policy and Practice*, *153*, 261–283.

Calabrò, G., Le Pira, M., Giuffrida, N., Inturri, G., Ignaccolo, M., & Correia, G. H. d. A. (2022). Fixed-route vs. demand-responsive transport feeder services: An exploratory study using an agent-based model. *Journal of Advanced Transportation*, *2022*, 1–20.

Cats, O., & Glück, S. (2019). Frequency and vehicle capacity determination using a dynamic transit assignment model. *Transportation Research Record*, *2673*(3), 574–585.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio,

Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Demissie, M. G., Phithakkitnukoon, S., Sukhvibul, T., Antunes, F., Gomes, R., & Bento, C. (2016). Inferring passenger travel demand to improve urban mobility in developing countries using cell phone data: a case study of senegal. *IEEE Transactions on intelligent transportation systems*, *17*(9), 2466–2478.

Diana, M., Quadrifoglio, L., & Pronello, C. (2009). A methodology for comparing distances traveled by performance-equivalent fixed-route and demand responsive transit services. *Transportation planning and technology*, *32*(4), 377–399.

Edwards, D., & Watkins, K. (2013). Comparing fixed-route and demand-responsive feeder transit systems in real-world settings. *Transportation research record*, *2352*(1), 128–135.

Gao, Y., Jiang, D., & Xu, Y. (2018). Optimize taxi driving strategies based on reinforcement learning. *International Journal of Geographical Information Science*, *32*(8), 1677–1696.

Garces, D., Bhattacharya, S., Gil, S., & Bertsekas, D. (2023). Multiagent reinforcement learning for autonomous routing and pickup problem with adaptation to variable demand. In *2023 ieee international conference on robotics and automation (icra)* (pp. 3524–3531).

Guo, Y., Tang, Z., & Guo, J. (2020). Could a smart city ameliorate urban traffic congestion? a quasi-natural experiment based on a smart city pilot program in china. *Sustainability*, *12*(6), 2291.

Han, Y., Wang, S., Ren, Y., Wang, C., Gao, P., & Chen, G. (2019). Predicting station-level short-term passenger flow in a citywide metro network using spatiotemporal graph convolutional neural networks. *ISPRS International Journal of Geo-Information*, *8*(6), 243.

He, L., Hu, Z., & Zhang, M. (2020). Robust repositioning for vehicle sharing. *Manufacturing & Service Operations Management*, *22*(2), 241–256.

Inturri, G., Giuffrida, N., Ignaccolo, M., Le Pira, M., Pluchino, A., Rapisarda, A., & D'Angelo, R. (2021). Taxi vs. demand responsive shared transport systems: An agent-based simulation approach. *Transport Policy*, *103*, 116–126.

Kåresdotter, E., Page, J., Mörtberg, U., Näsström, H., & Kalantari, Z. (2022). First mile/last mile problems in smart and sustainable cities: A case study in stockholm county. *Journal of Urban*

Technology, *29*(2), 115–137.

Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Kobyzev, I., Prince, S. J., & Brubaker, M. A. (2021). Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(11), 3964-3979. doi: 10.1109/TPAMI.2020.2992934

Li, H., Wang, Y., Xu, X., Qin, L., & Zhang, H. (2019, oct). Short-term passenger flow prediction under passenger flow control using a dynamic radial basis function network. *Appl. Soft Comput.*, *83*(C). Retrieved from https://doi.org/10.1016/j.asoc.2019.105620 doi: 10.1016/j.asoc.2019.105620

Liu, G., Yin, Z., Jia, Y., & Xie, Y. (2017). Passenger flow estimation based on convolutional neural network in public transportation system. *Knowledge-Based Systems*, *123*, 102–115.

Liu, S., Dai, S., Sun, J., Mao, T., Zhao, J., Zhang, H., et al. (2021). Multicomponent spatial-temporal graph attention convolution networks for traffic prediction with spatially sparse data. *Computational intelligence and neuroscience*, *2021*.

Ma, X., Zhang, J., Du, B., Ding, C., & Sun, L. (2019). Parallel architecture of convolutional bi-directional lstm neural networks for network-wide metro ridership prediction. *IEEE Transactions on Intelligent Transportation Systems*, *20*(6), 2278-2288. doi: 10.1109/TITS.2018.2867042

Meshkani, S. M., Farazmand, S., Bouguila, N., & Patterson, Z. (2024). Innovative on-demand transit for first-mile trips: A cutting-edge approach. *Transportation Research Record*, 03611981241239970.

Nair, R., & Miller-Hooks, E. (2011). Fleet management for vehicle sharing operations. *Transportation Science*, *45*(4), 524–540.

Oord, A. v. d. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Pratelli, A., Lupi, M., Farina, A., & Pratelli, C. (2018). Comparing route deviation bus operation with respect to dial-a-ride service for a low-demand residential area. *DATA ANALYTICS 2018*, 151.

Rahmani, S., Baghbani, A., Bouguila, N., & Patterson, Z. (2023). Graph neural networks for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*.

Ramaprasad, A., Sánchez-Ortiz, A., & Syn, T. (2017). A unified definition of a smart city. In *Electronic government: 16th ifip wg 8.5 international conference, egov 2017, st. petersburg, russia, september 4-7, 2017, proceedings 16* (pp. 13–24).

Shou, Z., & Di, X. (2020). Multi-agent reinforcement learning for dynamic routing games: A unified paradigm. *arXiv preprint arXiv:2011.10915*.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... others (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Song, D.-P., & Earl, C. F. (2008). Optimal empty vehicle repositioning and fleet-sizing for two-depot service systems. *European Journal of Operational Research*, *185*(2), 760–777.

SUMO. (2023). https://sumo.dlr.de/docs.

SUMO Taxi Simulation. (2023). https://sumo.dlr.de/docs/Simulation/Taxi.html.

Sun, Y., Leng, B., & Guan, W. (2015). A novel wavelet-svm short-time passenger flow prediction in beijing subway system. *Neurocomputing*, *166*, 109–121.

Traci - sumo documentation. (2024). Retrieved 2024-07-19, from https://sumo.dlr.de/docs/TraCI.html

Vallée, S., Oulamara, A., & Cherif-Khettaf, W. R. (2017). Maximizing the number of served requests in an online shared transport system by solving a dynamic darp. In *Computational logistics: 8th international conference, iccl 2017, southampton, uk, october 18-20, 2017, proceedings 8* (pp. 64–78).

van Kuijk, R. J., de Almeida Correia, G. H., van Oort, N., & Van Arem, B. (2022). Preferences for first and last mile shared mobility between stops and activity locations: A case study of local public transport users in utrecht, the netherlands. *Transportation Research Part A: Policy and Practice*, *166*, 285–306.

Vansteenwegen, P., Melis, L., Aktaş, D., Montenegro, B. D. G., Vieira, F. S., & Sörensen, K.

(2022). A survey on demand-responsive public bus systems. *Transportation Research Part C: Emerging Technologies*, *137*, 103573.

Verma, T., Varakantham, P., Kraus, S., & Lau, H. C. (2017). Augmenting decisions of taxi drivers through reinforcement learning for improving revenues. In *Proceedings of the international conference on automated planning and scheduling* (Vol. 27, pp. 409–417).

Wang, J., Zhang, Y., Wei, Y., Hu, Y., Piao, X., & Yin, B. (2021). Metro passenger flow prediction via dynamic hypergraph convolution networks. *IEEE Transactions on Intelligent Transportation Systems*, *22*(12), 7891-7903. doi: 10.1109/TITS.2021.3072743

Wen, J., Zhao, J., & Jaillet, P. (2017). Rebalancing shared mobility-on-demand systems: A reinforcement learning approach. In *2017 ieee 20th international conference on intelligent transportation systems (itsc)* (pp. 220–225).

Xiao, Z.-S., Mao, B.-H., & Zhang, T. (2018). Integrated predicting model for daily passenger volume of rail transit station based on neural network and markov chain. In *2018 ieee 3rd international conference on cloud computing and big data analysis (icccbda)* (p. 578-583). doi: 10.1109/ICCCBDA.2018.8386582

Yang, L., Yang, Q., Li, Y., & Feng, Y. (2019). K-nearest neighbor model based short-term traffic flow prediction method. In *2019 18th international symposium on distributed computing and applications for business engineering and science (dcabes)* (p. 27-30). doi: 10.1109/DCABES48411.2019.00014

Yang, S. (2011). Kalman filter-based short-term passenger flow forecasting on bus stop. *Journal of Transportation Systems Engineering and Information Technology*.

Ye, Y., Chen, L., & Xue, F. (2019). Passenger flow prediction in bus transportation system using arima models with big data. In *2019 international conference on cyber-enabled distributed computing and knowledge discovery (cyberc)* (p. 436-443). doi: 10.1109/CyberC.2019.00081

Yin, X., Wu, G., Wei, J., Shen, Y., Qi, H., & Yin, B. (2022). Deep learning on traffic prediction: Methods, analysis, and future directions. *IEEE Transactions on Intelligent Transportation Systems*, *23*(6), 4927-4943. doi: 10.1109/TITS.2021.3054840

Yoon, G., Chow, J. Y., & Rath, S. (2022). A simulation sandbox to compare fixed-route, semi-flexible transit, and on-demand microtransit system designs. *KSCE Journal of Civil Engineering*, *26*(7), 3043–3062.

Yu, W., Zhifei, W., Hongye, W., Junfeng, Z., & Ruilong, F. (2019). Prediction of passenger flow based on cnn-lstm hybrid model. In *2019 12th international symposium on computational intelligence and design (iscid)* (Vol. 2, p. 132-135). doi: 10.1109/ISCID.2019.10113

Yu, X., & Shen, S. (2019). An integrated decomposition and approximate dynamic programming approach for on-demand ride pooling. *IEEE Transactions on Intelligent Transportation Systems*, *21*(9), 3811–3820.

Zhang, J., Chen, F., Cui, Z., Guo, Y., & Zhu, Y. (2021). Deep learning architecture for short-term passenger flow forecasting in urban rail transit. *IEEE Transactions on Intelligent Transportation Systems*, *22*(11), 7004-7014. doi: 10.1109/TITS.2020.3000761

Zhang, S., Liu, Z., Shen, F., Wang, S., & Yang, X. (2020). A prediction model of buses passenger flow based on neural networks. In *Journal of physics: Conference series* (Vol. 1656, p. 012002).

Zhang, S., Tong, H., Xu, J., & Maciejewski, R. (2019). Graph convolutional networks: a comprehensive review. *Computational Social Networks*, *6*(1), 1–23.

Zhou, M., Jin, J., Zhang, W., Qin, Z., Jiao, Y., Wang, C., … Ye, J. (2019). Multi-agent reinforcement learning for order-dispatching via order-vehicle distribution matching. In *Proceedings of the 28th acm international conference on information and knowledge management* (pp. 2645–2653).