Algorithms for Context-Sensitive Prediction, Optimization and Anomaly Detection in Urban Mobility

By

Fangzhou Sun

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

May 11, 2018

Nashville, Tennessee

Approved:

Jules White, Ph.D.

Abhishek Dubey, Ph.D.

Douglas Schmidt, Ph.D.

Aniruddha Gokhale, Ph.D.

Hiba Baroud, Ph.D.

ACKNOWLEDGMENTS

First of all, I would like to express my deepest respect and gratitude to my advisors, Prof. Abhishek Dubey and Prof. Jules White, for their guidance throughout my Ph.D. research. They have patiently provided me with creative advice and comprehensive supervision on numerous research projects and publications.

I would like to thank my committee members, Prof. Douglas C. Schmidt, Prof. Aniruddha Gokhale and Prof. Hiba Baroud, for serving on the committee of my dissertation. They have offered helpful guidance on my research in classes and projects in the past few years. I also wish to thank Mr. Dan Freudberg from Nashville Metro Transportation Authority (NMTA), Mr. Jacob Staples and Mr. Lee Krause from Securboration Inc, Mr. Martin Lehofer and Dr. Monika Sturm from Siemens Corp. for their suggestions and insights.

I would like to express my special thanks to my parents, Mr. Jianhong Sun and Prof. Yancong Shi for their sacrificial love and best wishes. I would also like to thank my fiancée, Miss. Yunge Tong, for her companionship and selfless support. Special thanks also go to Prof. Yu Sun for giving me insights and inspiring me to follow my dreams. This research could not have been possible without the support of my friends, Chinmaya Samal, Saideep Nannapaneni, Shashank Shekhar, Subhav Pradhan, Yao Pan, Oruganti Aparna, Sanchita Basak, Shweta Khare, Peng Zhang, Qishen Zhang, Xiaochen Yang, Dongqing Zhang, Shunxing Bao, Amin Ghafouri, Geoffrey Pettet and Scott Eisele, who directly encouraged and supported me in research and writing.

Finally, I would like to acknowledge the research funding from National Science Foundation (NSF), Office of Naval Research (ONR), and Siemens Corporation for supporting the various research projects in which I have participated.

ABSTRACT

Transportation infrastructure is a complex human cyber-physical system that is currently facing significant challenges in many communities around the world. The problem emanates from increased congestion, which results in large-scale inefficiencies, including significant personal, health and environmental costs. The human integrated nature of this resource constrained system allows communities to go beyond the traditional mechanisms of adding infrastructure which is often expensive and difficult to build and embrace data-driven smart solutions that focuses on providing a robust decision support system, which can enable humans to use and optimize the system more efficiently. However, there are several challenges that arise due to the heterogeneity, sparsity, and noise in the data collected in an urban environment.

This dissertation examines a unique application platform called transit-hub that enables (1) integration of spatially and temporally distributed sensor streams, (2) integration of simulation-based decision support systems, and (3) development of experiments to understand how advanced decision support tools improve the utilization of the transportation infrastructure. We designed data mining and machine learning techniques for context-sensitive prediction of long-term, short-term and real-time delays in sparse public transit networks. In order to solve data sparsity issues, shared route segment networks and multi-task neural networks were developed. Further, we integrated algorithms for analyzing the performance of public transit networks and developed mechanisms to optimize the on-time performance under uncertainty of traffic and weather conditions. Heuristic search algorithms as well as sensitivity analyses of the hyper-parameters were also developed. Robust detection of anomalous operations of transit networks over a large metropolitan area was enabled by using deep learning techniques within the platform. A specific innovation is to set up the problem of identifying the non-recurring traffic congestion as an image classification task and to use convolutional neural networks to explain the congestion.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In 2007, for the first time in world history, the global urban population exceeded the number of people living in rural areas [1]. From 2010 to 2050, this population is expected to grow from 3.5 billion to 6.3 billion, which is an 80% percent growth [2]. With population booming and growing urbanization, urban mobility, which involves the movement of people, vehicles, and data in the city, faces increasing pressure regarding accessibility and sustainability. Urban mobility has become plagued with issues including traffic congestion and transit delays. Because of traffic congestion, people in the United States traveled an extra 6.9 billion hours and purchased an additional 3.1 billion gallons of fuel in 2014, resulting in extra time and fuel cost of $160 billion [3].

Transportation infrastructure is a complex human cyber-physical system that is currently facing significant challenges in many around the world. The problem emanates from the increased congestion, which results in large-scale inefficiencies, including significant personal, health and environmental costs. The traditional solution for cities to improve urban mobility is to build more infrastructure (e.g., roads, subways, etc.) and add more transportation services (e.g., bus routes, service hours, etc.), which can be very expensive. For example, it is estimated that constructing a two-lane undivided road in urban areas will cost between $2 and $5 million per mile [4]. In 2017, Nashville unveiled a mass transit plan that calls for $5.4 billion to build an ambitious light system that includes new and expanded light rail, commuter rail and bus rapid transit. However, the proposal offers a 25-year plan, which is a long period for the completion of construction. It is expected that the renewal of old infrastructure and expansion of new infrastructure will result in trillions of dollars in the investment over the coming decades [5]. Moreover, because of the horizontal growth of urban road networks and the increasing costs to urban core, residents are pushed

out of the suburban areas and need to travel longer distances to their destinations. For example, Figure 1.1 illustrates the daily travel demand distribution from home to work for employees of a major employer in Nashville. The steady development of urban infrastructure increases the complexity of multi-modal traveling, route planning, and the uncertainty of traffic accidents and travel delays on the roads.

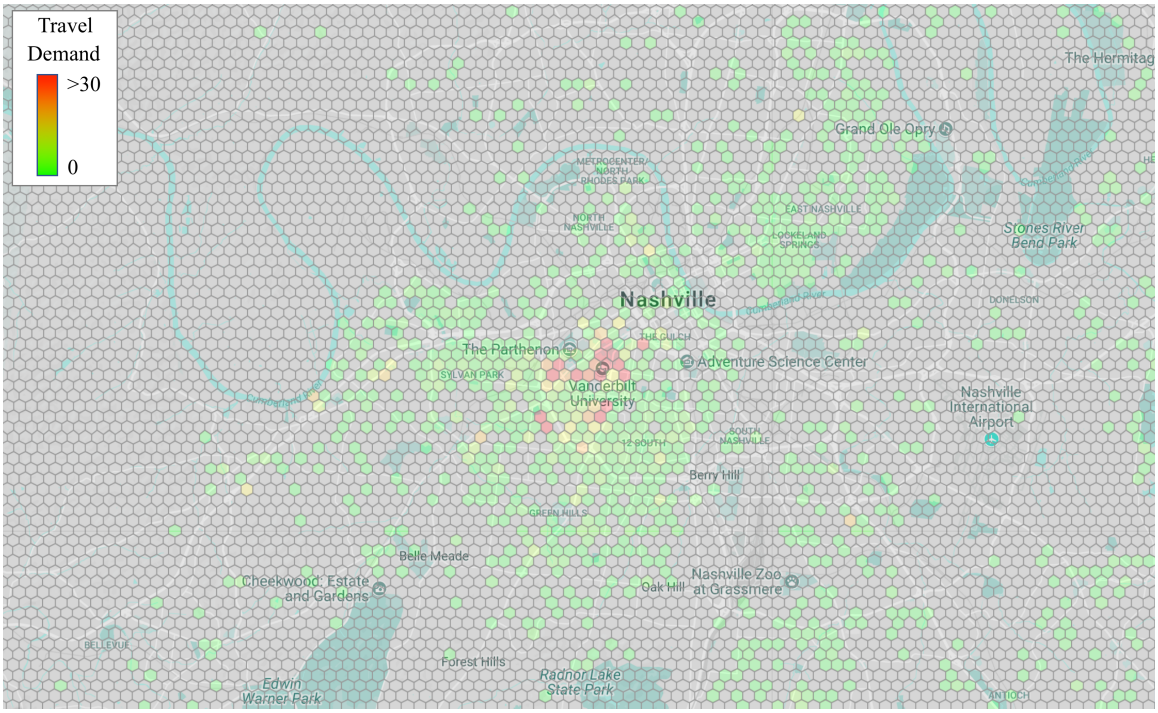


Figure 1.1: The daily travel demand distribution from home to work of a major employer's employees in Nashville.

There have been a number of transportation revolutions in history, mostly caused by major advances in transport technologies [6]. The most recent one is driven by the concepts of Smart City and Internet of Things (IoT) [1]. Various sensor devices are being developed and deployed with a wide range of research and operational objectives in the urban environment. It is estimated that 8.1 billion connected IoT units were in use in 2017, which is a 31% increase from 2016 [7]. As illustrated in Figure 1.2, these units generate an enormous

[1]The Internet of things (IoT) is the network of computing devices embedded in everyday objects, which enables these objects to connect and exchange data.

amount of data to serve different applications in the transportation domain, such as weather forecasting, traffic monitoring, vehicle locating, travel demand estimating, etc. The exponential growth of data provides opportunities for urban mobility systems. Instead of relying on individual or several limited data sources, city planners can gain in-depth knowledge of the mobility systems by aggregating multiple data sources and applying advanced analytic techniques such as machine learning and data mining. For example, traffic flow data and event information can be combined together to explain the root causes of anomalous congestion in the city [8, 9, 10]. City planners can also identify the bottlenecks from historical data and make robust plans about vehicle dispatching, route designing to reduce the uncertainty that operations in the real-world deviate from the schedule.

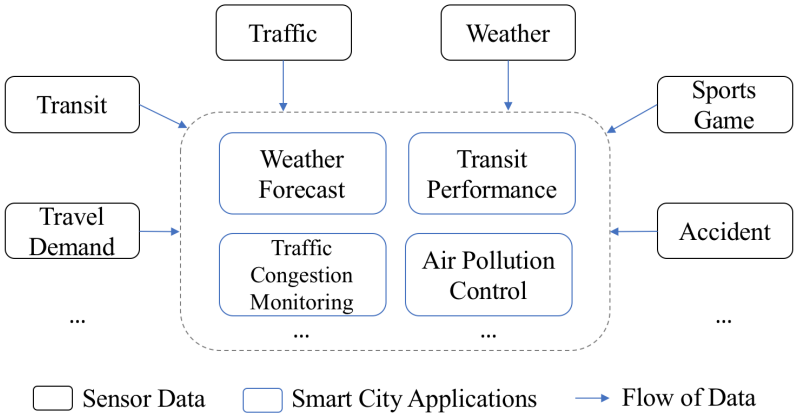


Figure 1.2: The sensor data that needs to be classified, structured and managed for various smart city applications are heterogeneous and in large scale.

The data also creates robustness challenges for urban mobility systems. Sensors are prone to failures. For example, a survey found that, in 2013, one-third of 27,000 traffic sensors on California Highways were off-line [11]. Dysfunctional sensors result in erroneous and missing data issues, which reduce the performance of services that rely on the data. Moreover, urban data can suffer from sparsity issue. There is a spatio-temporal uncertainty of when and where the data is available from the mobile sensors. For example, automatic vehicle location (AVL) and automatic passenger counter (APC) systems can pro-

vide real-time bus data for at-stop displays [12], bus time prediction [13, 14, 15], schedule planning optimization [16, 17], real-time control strategies [18, 19], etc. But in some low-density areas, the low frequency of bus trips makes the data too sparse to provide useful information. Sensors can be expensive for large-scale deployment. For example, the inductive loop surveillance, which is usually installed at intersections to track traffic condition, cost over $800 per unit [20] in 2013. The high cost makes it impossible for these types of units to be installed on all road intersections of a city.

Open Problem ⇒ Building a data-driven transit system to improve the perception of public transit by integrating data mining and machine learning techniques. Cities around the world are working on various smart city projects to establish replicable, scalable and sustainable solutions to the urban mobility issues. For example, led by National Institute of Standards and Technology (NIST), the Global City Teams Challenge (GCTC) program attracts over 100 cities and communities globally to collaboratively work on smart city projects [21], and transportation is one of the most important topics. Since the traditional way of building infrastructure and services can be expensive and time-consuming, this research focuses on applying model-driven and data-driven algorithms and techniques to improve the efficiency, effectiveness, and robustness of the existing urban mobility systems. The human-integrated nature of the transit systems allows the communities to go beyond the traditional mechanisms of adding infrastructure which is often expensive and difficult to build, and to embrace data-driven smart solutions that focus on providing a robust decision support system, which can enable humans to use the system more efficiently. Furthermore, these smart solutions can help the stakeholders adjust the parameters of the transportation system, making it more robust and efficient.

1.1 Understanding the Challenges

In this section, we discuss the key challenges for building a data-driven transit system, which arise due to the heterogeneity, sparsity, and noise in the data collected in the urban

environment. As discussed in the previous sections, the mobility in the urban environment is susceptible to many internal and external factors. The heterogeneity in problem domains, applications, and hardware introduces significant uncertainty and complexity in modeling, predicting and optimizing the mobility of people and vehicles.

1.1.1 Challenge 1: Handling Sparsity and Quality Issues in Data

Driven by the concepts of Smart City and IoT, various sensors of distributed systems are being developed and deployed on vehicles, such as Road-Side Units (RSUs) and stations. In modern cities, numerous Wireless Sensor Networks (WSNs) have appeared in different sub-domains of urban mobility systems, such as traffic monitoring, transit vehicle tracking, passenger counting, etc. Due to the domain- and device-specific constraints and requirements, these sensors have different features, accuracy, and energy consumption level, and they show considerable heterogeneity in data format and sampling rates. Some mapping and integration mechanisms are needed to synchronize the received data in terms of unit, time and location. Collecting data for the entire city makes the datasets very large and introduces more scalability difficulty.

Since the availability of sensors differs spatially and temporally, data sparsity is one of the biggest issues. Sensors can be expensive, and the process of static sensor deployment is time-consuming, which limits the number of sensors that are usable on the city scale. For low-density cities, deploying sensors all over the city is beyond budget. Instead of using static sensors, some novel analytic services rely on crowd sensing that collects data directly from vehicles and end users, e.g., using embedded sensors in mobile phones. The mobility of users introduces a significant uncertainty of when and where the user data is collected and uploaded. In the transit domain, the data from automatic vehicle locators (AVLs) on buses is a popular resource for tracking transit delay and traffic congestion. However, for cities where the frequency or coverage of bus routes is low, the data sparsity issue is severe. The data sparsity issue is one of the main challenges for data aggregation and analytics.

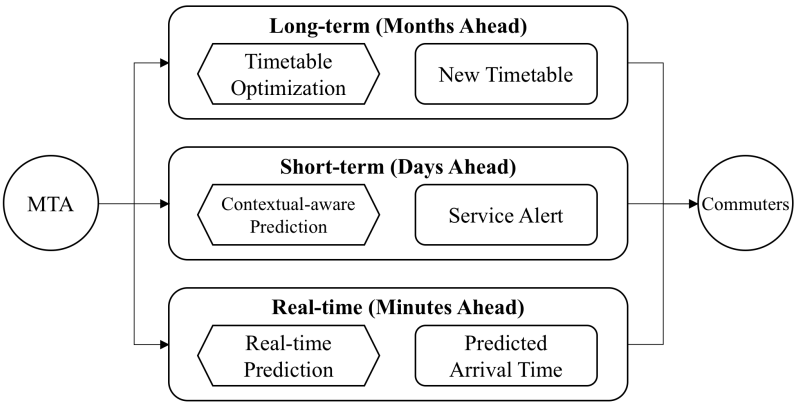


Figure 1.3: MTA provides transit schedule and status update to commuters in three terms (i.e., long-term, short-term and real-time) to help them plan transit trips.

Furthermore, the urban data suffers from integrity issues. Data can be missing, duplicated, and contains anomalies. For example, sensors used in traffic monitoring are prone to anomalies and failures [22, 23]. Faulty data without proper cleaning will lead to the inferior performance of systems using the data. Even though some techniques have been developed in the related area, it remains a challenge to detect faulty data quickly without high false positive (FP) and false negative (FN) rates.

1.1.2 Challenge 2: Analyzing and Predicting for Multiple Timescales

In transit systems, different stakeholders typically have various interests. From the perspective of city planners and MTA engineers, they need to analyze the long-term and short-term travel demand and performance patterns in order to better design and optimize the transit systems in various metrics, such as on-time rates, passenger waiting, and transfer time, etc. For residents and commuters, they need predictive information in advance for trip planning and real-time updates when taking transit trips. These requirements motivate advanced data-driven transit analytics in multiple timescales (illustrated in Figure 1.3).

The on-time performance of transit networks, which is mostly reflected in how much the actual travel time differs from the scheduled time, has a great impact on the choices

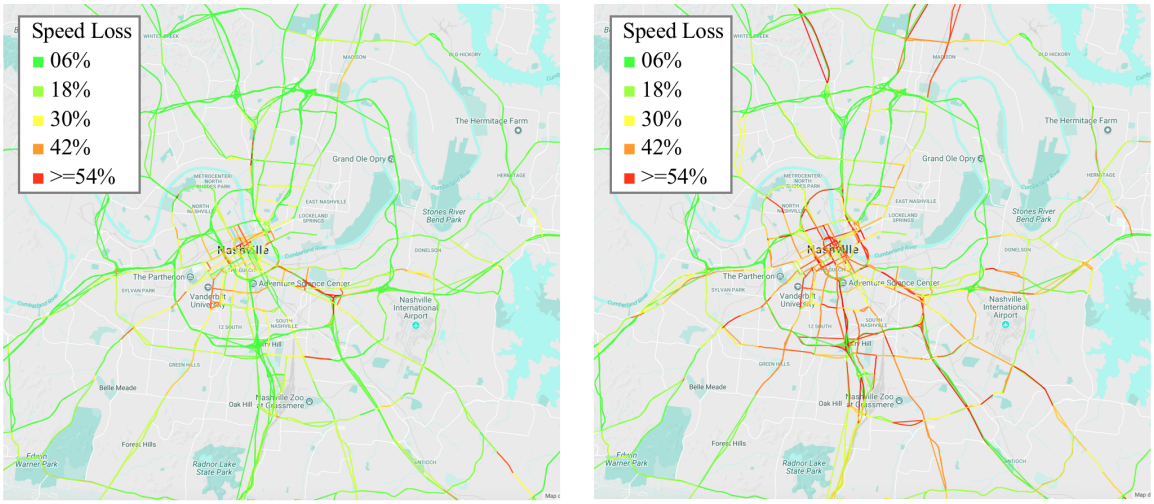
of travelers. For example, because of the low cost of deployment and operation, as well as relatively high capacity, the bus is usually the only transit system in many small and mid-size cities. However, unpredictability is a major issue that prevents people from taking buses [24]. Less public transportation use is expensive for society. Parking private cars wastes space, and private transportation increases green gas emissions and road congestion. The traditional way to improve mobility performance is to add new infrastructure. But it's not always possible because there are various infrastructure constraints, like limitations of population density and construction cost.

Urban mobility is based on a complex transportation system which consists of multiple modes, routes and trips. These modes interacting with each other introduce great internal uncertainty factors. For example, the delay of one trip often has a cascading effect on the next one. There are also many external factors affecting the actual mobility of vehicles and leading to delay: (1) traffic congestion, (2) weather condition, (3) special events like sports games, concerts, and (4) travel demand. The substantial impact of arrival delay (e.g., consequent bus trips in a block sometimes share the same vehicle, so the delay of the previous trip will cause departure delay of next trip) is a major internal uncertainty factor and was found to have a substantial impact on commuter satisfaction [25]. Systematically modeling the uncertainty sources in urban mobility, identifying the hidden patterns in historical data, and descriptively and predictively estimating the actual mobility for the future remain research challenges.

Additionally, developing an effective short-term prediction for transit systems is currently an open problem to solve. Route segments typically have different delay patterns and are affected by the contextual factors in varying degrees (see Section 3.4.3.1). Technically, multiple historical models can be built for different contextual features. However, since there are so many event types and their impacts on transit delay interact with each other and have varying spatio-temporal characteristics, advanced prediction models are needed to effectively integrate all contextual features. Additionally, due to the limited budget and

computation resources, it may not be feasible to create and train a single prediction model for each separate segment in practice because of the high computation time.

1.1.3 Challenge 3: Detecting Contextual Anomalous Operations in Transit Networks



(a) (b)

Figure 1.4: Nashville traffic congestion patterns on two time windows: (a) morning peak hours on Sept. 1, 2017 between 7 AM and 9 AM, (b) evening peak hours on Sept. 1, 2017 between 4 PM and 6 PM.

The anomalies that we consider are traffic congestion and severe transit delay. Traffic is probably the most important data in urban mobility systems. Wrong traffic information will result in low-performance of many mobility services, such as arrival delay prediction and on-time performance optimization. Traffic congestion in urban areas has been a significant issue in recent years. Studies have been conducted to explore the different patterns of congestion [9, 26, 27]. Congestion can be classified as recurring congestion and non-recurring congestion [28] according to their causes and frequencies. Recurring congestion is closely related to peak hours, time in the day, weekdays, and seasons [9, 29, 26], while non-recurring congestion is caused by accidents, roadwork, special events, or adverse weather [27, 30].

Figure 1.4 illustrates the traffic congestion patterns in two time windows (the speed loss of a road segment is defined as the percentage of speed decreased from the speed limit). Existing work usually treats traffic congestion as an independent spatio-temporal fragment for one road segment in one time period [31, 32]. A congestion pattern will be more meaningful if investigated among several congested road segments that are spatially and temporally related. Many studies focus on recurring congestion, but very few investigated the non-recurring congestion, even though non-recurring congestion has a severe impact [33, 27, 30]. Moreover, traffic patterns are actually aggregated results of many contextual and environmental factors (e.g., time, weather, events, etc.). It still remains an open challenge to utilize the rich urban sensor data to explore the root causes of traffic congestion, especially the non-recurring ones. Furthermore, modeling the historical traffic patterns is not enough — it will be more helpful if the traffic congestion for future events can be accurately predicted.

1.2 Overview of the Research

Given the background, emerging trends and key research challenges, we are going to present an overview of the proposed research of how algorithms and techniques can be utilized to solve the challenges identified in Section 1.1 to improve the robustness of urban mobility systems. We develop a unique application platform called transit-hub that enables (1) integration of spatially and temporally distributed sensor streams, (2) integration of simulation-based decision support systems, and (3) development and execution of experiments to understand how advanced decision support tools improve the utilization of the transportation infrastructure.

Table 1.1 shows the challenges and corresponding solutions with section locations in the research. An integrated view of this research is illustrated in Figure 1.5, which includes three main modules: (1) context-sensitive predictive mechanisms for long-term, short-term and real-time delays, (2) optimization algorithms for public transit schedules considering

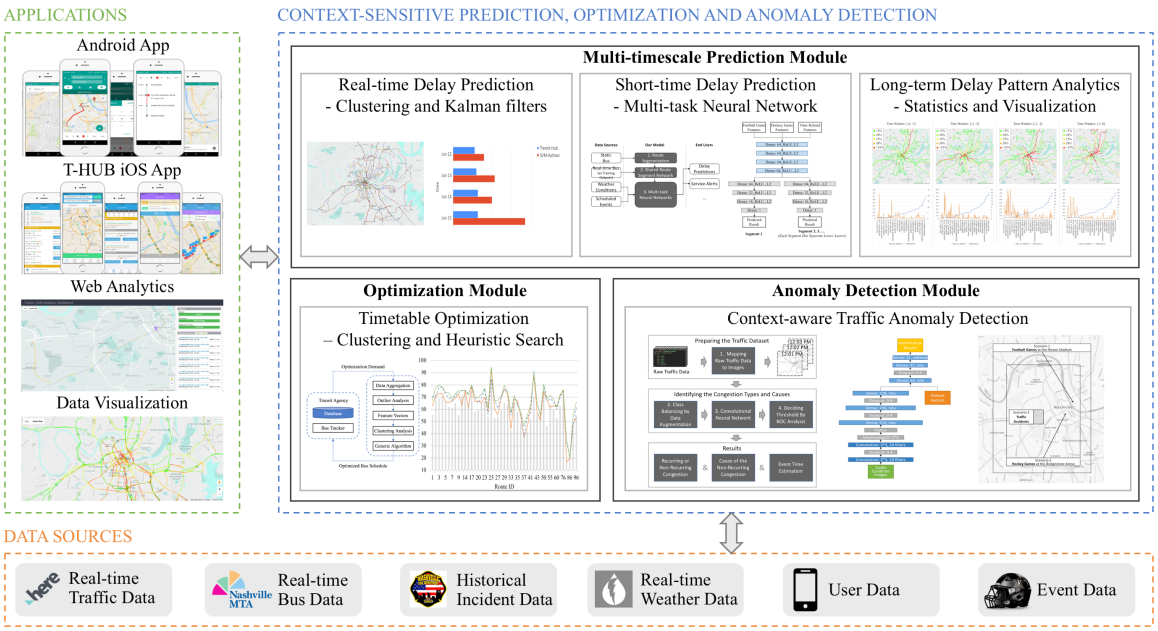


Figure 1.5: An integrated view of the research

seasonal delays, and (3) deep neural networks for context-aware anomaly detection.

Contribution 1: Robust arrival delay prediction models that solve data sparsity issue, and a multivariate predictive model to explore the significance of contextual predictors

To address the heterogeneity and sparsity issues of sensor data, we propose a robust urban data sensing mechanism that collects sensor data from multiple sources, stores the data in a central MongoDB database, and processes the raw sensor data to solve the data sparsity issue. The data store enables continuous data collection and aggregation, spatio-temporal mapping, surrogating data sensing, and accurate anomalous data identification. The data store provides a robust foundation for improving the urban mobility.

We provide an algorithm that generates shared bus route segment networks from standard General Transit Feed Specification (GTFS) datasets. Using shared route segment networks, we present a better real-time delay prediction model that combines clustering analysis and Kalman filters and uses real-time data from shared route segments. We show

Table 1.1: Key research challenges and corresponding solutions with section locations in the research.

Challenge	**Approach**	**Section**
Handling Data Sparsity & Quality Issues	Shared Route Segment Networks	Section 3.4.2.1
	Multi-task Neural Networks	Section 3.4.3.3
	Data Augmentation by Crossover Operator	Section 5.5.2
Analyzing and Predicting for Multi-timescales	Seasonal Variation Mitigation by Clustering	Section 4.4.1
	Timetable Optimization by Greedy and Genetic Algorithms	Section 4.4.2
	Real-time Prediction by Clustering and Kalman filters	Section 3.4.2
	Microservice Architecture for Back-end Analytics	Section 3.5
Detecting Contextual Anomalies	Feature Extraction by Image Representation	Section 5.5.1
	Classification by Convolutional Neural Networks	Section 5.5.3

the efficacy of our real-time delay prediction model. When predicting the travel time delay of segments 15 minutes ahead of scheduled time, our model reduced the root-mean-square deviation (RMSD) by about 30% to 65% compared with a SVM-Kalman model [15]. In order to predict the transit delay in the short-term when no real-time transit and traffic data is available, we propose a generic tool-chain that takes contextual information (e.g., scheduled events and forecasted weather conditions) as inputs and provides service alerts as outputs. Multi-task neural networks are trained using historical GTFS feeds as well as contextual information. Experimental evaluation shows that the proposed tool-chain is effective at predicting severe delay with a relatively high recall of 76% and an F1 score of 55%. The details are presented in Chapter 3.

We also propose an online architecture called DelayRadar. The novelty of DelayRadar lies in three aspects: (1) a data store that collects and integrates real-time and static data from multiple sources, (2) a predictive statistical model that analyzes the data to make pre-

dictions on transit travel time, and (3) a decision-making framework to develop an optimal transit schedule based on variable forecasts related to traffic, weather, and other impactful factors. This research focuses on identifying the model with the best predictive accuracy to be used in DelayRadar. According to the preliminary study results, we are able to explain more than 70% of the variance in the bus travel time, and we can make future travel predictions with an out-of-sample error of 4.8 minutes with information on the bus schedule, traffic, and weather.

Contribution 2: Optimizing the performance of transit networks under uncertainty that comes from many internal and external factors

To address the challenges of reducing the uncertainty that the actual performance of the transit network differs from the schedules, an unsupervised mechanism is proposed to improve the on-time performance of bus services with fixed schedules at the re-planning stage. The mechanism learns how timetables can be divided into seasonal schedule by applying outlier analysis and clustering analysis on bus travel times. The feature vectors we use include mean, median, and standard deviation of the historical travel time aggregated by route, trip, direction, timepoint segment, and month. In order to optimize timetables for a month group, we present a greedy algorithm, a genetic algorithm as well as a particle swarm optimization algorithm to optimize the scheduled arrival and departure time at timepoints to maximize the probability of bus trips that reach the desired on-time range. Sensitivity analysis on hyper-parameters is also presented to choose the best settings. Simulations show that using the proposed genetic algorithm with clustering improved the original performance from 57.79% to 68.93%. Details can be found in Chapter 4.

Contribution 3: Modeling traffic patterns and transit delay by Considering Contextual Information to identify anomalies in the system

The third contribution of this research includes an image-based deep learning model using contextual event information to identify non-occurring traffic congestion and classify the causes of traffic anomalies. Non-recurring traffic congestion is caused by temporary

disruptions, such as accidents, sports games, adverse weather, etc. We use data related to real-time traffic speed, jam factors (a traffic congestion indicator), and events collected over a year from Nashville, TN to train multi-layered deep neural networks. The traffic dataset contains over 900 million data records. The network is thereafter used to classify the real-time data and identify anomalous operations. Compared with traditional approaches of using statistical or machine learning techniques, our model reaches an accuracy of 98.73 percent when identifying traffic congestion caused by football games. Our approach first encodes the traffic across a region as a scaled image. After that the image data from different timestamps is fused with event- and time-related data. Then a crossover operator is used as a data augmentation method to generate training datasets with more balanced classes. Finally, we use the receiver operating characteristic (ROC) analysis to tune the sensitivity of the classifier. We present the analysis of the training time and the inference time separately.

Although the proposed solution will be demonstrated to solve the problem of identifying and predicting traffic congestion patterns, the core idea and mechanism can be applicable to other problem domains (such as smart grid, river water management, etc.), as long as the collected data has spatio-temporal connections and can be represented using single- or multi-dimensional images.

Dissertation Outline. The remainder of the dissertation is organized as follows: Chapter 2 introduces the basic terminology and concepts involved in data-driven transit systems; Chapter 3 presents real-time, short-term, and long-term delay prediction mechanisms that solve data sparsity issues as well as a multivariate predictive model that investigates the significance of bus delay predictors; Chapter 4 presents an unsupervised bus timetable optimization approach that integrates clustering analysis, genetic algorithm optimization, and sensitivity analysis; Chapter 5 proposes an image-based deep learning model for identifying non-recurring traffic congestion and inferring the causes; Chapter 6 presents concluding remarks and future work.

CHAPTER 2

BACKGROUND ABOUT DATA-DRIVEN TRANSIT SYSTEMS

In this chapter, we first introduce the basic terminology and concepts involved in transit systems. Readers who are familiar with the terminology and concepts may skip this chapter.

2.1 Terminology and Concepts

There are some basic terminology and concepts that define public transportation systems:

- **Route.** A bus route has multiple trips that depart at different times of the day. Each trip has the same scheduled time at different timepoints along the route.
- **Stop.** A stop is a bus station that is setup on a bus route. Multiple routes may use the same stop.
- **Timepoint.** A timepoint is a bus stop that is designed to accurately record the timestamps when buses arrive and leave the stop. Bus drivers use timepoints to synchronize with the scheduled time. If a bus arrives early, it will not depart until the scheduled time. If a bus arrives late, it will depart immediately. Not all stops are timepoints. Usually a bus route contains a handful of timepoints.
- **Trip.** A trip is a sequence of two or more stops that occurs along a route. Buses on a trip are always scheduled to depart at the same time on different service days.
- **Block.** A block consists of a single trip or many sequential trips made using the same vehicle. The group of trips share the same service day and block.

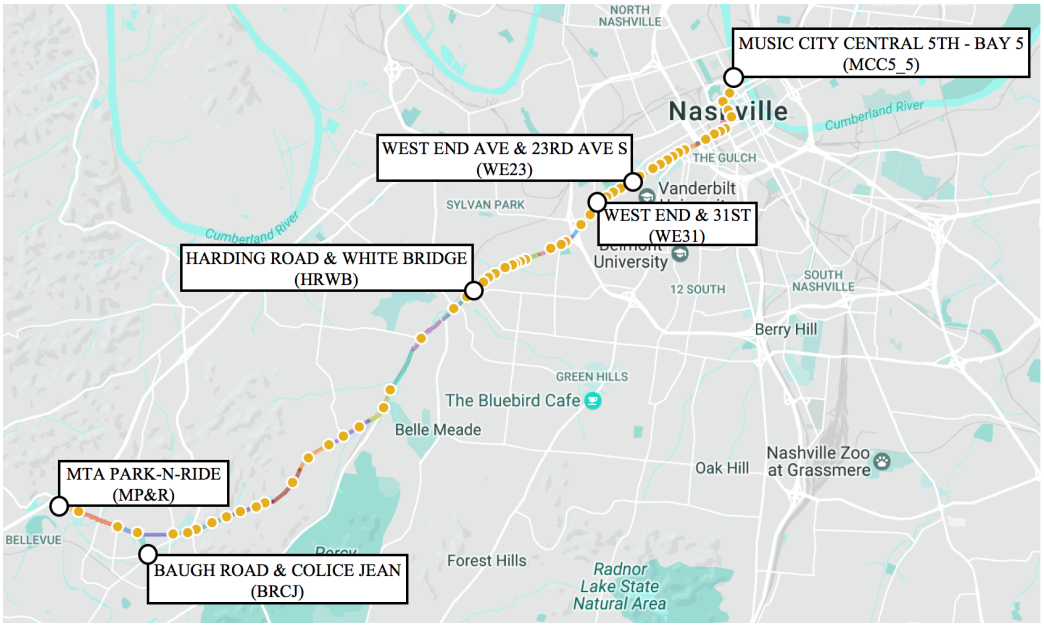


Figure 2.1: Timepoints on bus route 5 in Nashville

These concepts are illustrated in Figure 4.6. It shows route 5, a major bus route that connects downtown Nashville and the southwest communities in Nashville. The route contains 6 timepoints (i.e., MCC5_5, WE23, WE31, HRWB, BRCJ, and MP&R) and 5 segments between the 6 timepoints. A bus trip that departs at 11 AM from MCC5_5 and arrives at 11:46 AM to MP&R, and another bus trip that departs at 12:02 PM from MP&R and arrives at 12:50 PM to MCC5_5 belong to the same block, since they are designed to use the same bus.

2.2 Data Sources

We have been collaborating with the Nashville Metropolitan Transit Authority (MTA) for accessing the static and real-time transit data all across the Nashville city. Since October 2016, we have been continuously collecting and storing real-time traffic data from HERE API [34] for all major roads in the Nashville area. In order to explore the impact of contextual events on urban mobility, we also collect the data about incidents and sports games. We cooperate with the Nashville Fire Department [35] to access their incident datasets,

and manually collect the information about sports games from the web. As illustrated in Table 2.1, the details of the datasets that we have integrated into the system are as follows:

- *Static GTFS data sets:* Static bus schedules and associated geographic information in the General Transit Feed Specification (GTFS) [36] are collected. The data sets include routes, trips, stops, stop times and physical layout.
- *Real-time GTFS data feed:* Real-time transit fleet feed in GTFS real-time [37] format that contains three types of data: service alerts, trip updates and vehicle positions. The data source of the feed includes streaming Automatic Vehicle Location (AVL) data on operating buses.
- *Time point data sets:* Time point Datasets are the historical bus data at time points, including route ID, trip ID, drive ID, actual departure and arrival time, etc. This data is not available in real-time and is only made available at the end of the month.
- *Traffic data sets:* The traffic dataset provides the real-time traffic information on road segments, such as speed limit, real-time speed, jam factor (JF), etc. The dataset contains historical traffic data for 3049 TMC road segments in the Nashville area. The traffic data coverage is illustrated in Figure 2.2.
- *Sports game data sets:* The sports game dataset contains the operation information about sports games, such as game type, start and end time, attendance, location, etc.
- *Incident data sets:* The incident dataset provides the detailed records of incidents and the responding vehicles. For each incident, it provides the coordinates, incident type, alert time, vehicle arrival and departure time, weather condition, etc.

2.3 Data Management

Data Collection. We have to handle data from each source differently as they have different update rates and formats. For example, (i) Bus schedule data (static GTFS) is

Table 2.1: Real-time and static datasets collected in the research.

	Format	Source	Updating Interval	Size	Date Range
Bus Schedule	GTFS	Nashville MTA	Every public release	134 MB	09/2015 - present
Real-time Transit	GTFS-realtime	Nashville MTA	Every minute	723 GB	02/2016 - present
Timepoint	Excel sheet	Nashville MTA	Every month	300,000 entries/month	04/2016 - 07/2017
Weather	JSON	DarkSky API	Every 5 minutes	82.4 MB	03/2016 - present
Traffic	TMC	HERE API	Every minute	245 GB	10/2016 - present
Accident	JSON	Nashville Fire Department	Manually	387 MB	03/2014 - 03/2017
Sports Game	JSON	ESPN and others	Manually	28 Games	10/2016 - 01/2017

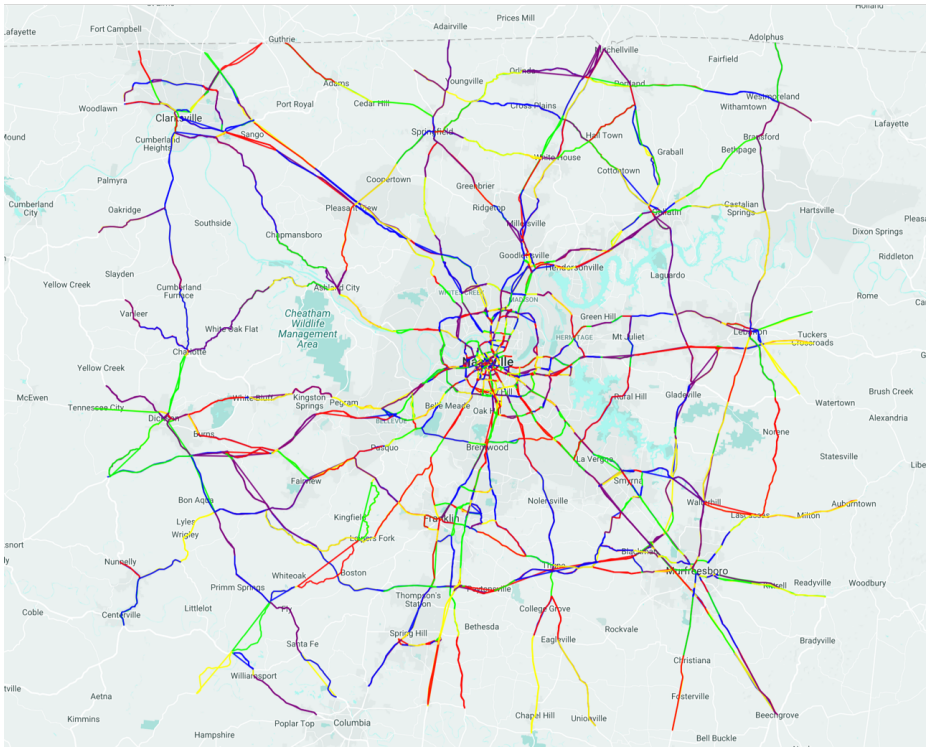


Figure 2.2: The colored road segments show coverage of the collected traffic data in Nashville.

updated only when MTA modifies its bus routes or schedules; (ii) Historical time point data set is collected by MTA at the end of the month and is then manually transferred and imported into our MongoDB database. On an average, we collect approximately three hundred thousand entries each month; and (iii) For the real-time transit data, our back-end server requests the data from these real-time feeds every minute and stores the responses in the database (see Table 2.1).

Data Cleaning. Data cleaning is a crucial step for data pre-processing to handle the following issues:

- *Duplicated data.* Detecting and eliminating duplicated data is one of the major tasks for data cleaning. We compare and remove data with the same time stamps and key-value pairs.

- *Data with logistic errors.* This type of data exists mainly in the real-time bus location data. To deal with it, for example, we remove the records where a bus' distance from a stop changes too fast, or if it moves in the wrong direction. This is done using some custom filters created by us.

- *Missing data.* This can happen for various reasons, which are: (a) operational disruptions due to service alerts, (b) hardware failures, or (c) data transmission issues. The missing data is filled in using linear interpolation on the sampled data.

Data Storage. The large scale of the historical and real-time transit data that are accumulated over time requires efficient storage and management methods. Also, the stored data must be accessible to multiple clients in the system at the same time. To meet this scale requirement, we employ JSON as the data structure and MongoDB [38] for data storage. MongoDB is a distributed NoSQL database that can efficiently store and query data on the scale of terabytes.

CHAPTER 3

CONTEXT-SENSITIVE PREDICTIVE MECHANISMS FOR LONG-TERM, SHORT-TERM, AND REAL-TIME DELAYS IN TRANSIT NETWORKS

Unpredictability is one of the top reasons that prevent people from using public transportation. In this chapter, we describe a system in use in Nashville and illustrate the analytic methods developed by our team. These methods use both historical as well as real-time streaming data for online bus arrival prediction. The historical data is used to build classifiers that enable us to create expected performance models as well as identify anomalies. These classifiers can be used to provide schedule adjustment feedback to the metro transit authority. We show how these analytic services can be packaged into modular, distributed and resilient micro-services that can deployed on both cloud back ends as well as edge computing re- sources. We also propose a generic tool-chain that takes standard General Transit Feed Specification (GTFS) transit feeds and contextual information (recurring delay patterns before and after big events in the city and the contextual information such as scheduled events and forecasted weather conditions) as inputs and provides service alerts as output. Particularly, we utilize shared route segment networks and multi-task deep neural networks to solve the data sparsity and generalization issues. Experimental evaluation shows that the proposed toolchain is effective at predicting severe delay with relatively high recall of 76% and F1 score of 55%. The content of this chapter has appeared in two conference papers [39, 40], a book chapter [41], and a journal paper [42].

3.1 Problem Overview

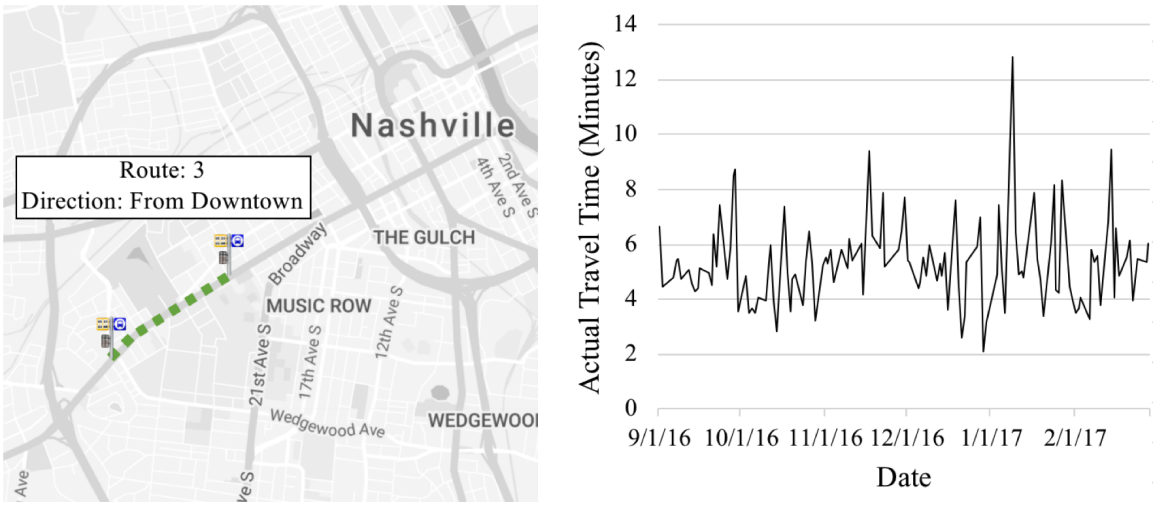
Emerging Trends. Public transit ridership in the United States increased by 37% between 1995-2015, which is roughly twice as much as the country's population growth

(21%) in the same years [43]. In 2013 alone, there were 10.7 billion trips taken on U.S. public transportation [44]. Meanwhile, people in the U.S. have been reducing the use of personal vehicles [45]. Public transportation has become an essential part of communities and cities.

Bus services, which is one of the most important segments of public transportation, are vulnerable to delays and congestion due to traffic congestion, weather conditions, special events, etc. Travel and arrival time variation was found to have a substantial impact on commuter satisfaction [25]. Moreover, people's tolerance to errors in bus time predictions is quite low [46]. Besides lack of cross-town routes and low service reliability, unpredictability is one of the top reasons that prevent people from using public transportation [1] [25, 46]. The variance of travel time on a bus route can very high due to a number of reasons, including passenger load and unload times, traffic congestions, weather conditions, events such as football and hockey games. For example, Figure 3.1(a) shows a bus route segment on route 3 departing from downtown; Figure 3.1(b) illustrates the travel times of a bus trip (departs at the same time of day) on the same bus route segment for six months. Since commuters want to arrive on time, they have to compromise with the unreliable transit service and accommodate some extra time in their schedule, which causes inconvenience and dissatisfaction to bus passengers.

Providing real-time bus schedules reduces this uncertainty and improves passenger experience and increases ridership. A direct benefit of increased ridership on public transport is the reduced use of personal vehicles and hence reduction in both traffic congestion and greenhouse emissions. Recently, transit agencies have been integrating real-time sensors into public transit systems. A number of technological systems have been developed by academic researchers and commercial companies to utilize this real-time data. For instance, AVLs (Automatic Vehicle Location) and APCs (Automatic Passenger Counter) can provide real-time data such as vehicle travel time, arrival and departure time, and passen-

[1] In 2015, a mass transit survey from Nashville, USA revealed that people who regularly use the transit system focus mostly on creating a system in which buses arrive on time [47].



(a) (b)

Figure 3.1: (a) A route segment on bus route 3 leaves downtown; (b) The variance of actual travel time on a bus route segment is very high in time period between Sept. 1, 2016 and Feb. 28, 2017.

ger boarding counts. This data can be used for at-stop displays [12], bus time prediction [13, 14, 15], schedule planning optimization [16, 17], real-time control strategies [18, 19], etc.

Research Gap The real-time prediction mechanisms have some open problems. Accurate real-time bus arrival and departure data that many prediction systems use is not always available. For example, in Nashville only special bus stops called timepoints are equipped with sensor devices that record exact times. There are over 2,700 bus stops all over the city but only 573 timepoints. In addition, the timepoint dataset is not real-time. It is available at the end of each month when the Nashville Metropolitan Transit Authority (MTA) summarizes and analyzes the historical data. APCs can help provide accurate timing of when a bus stops at a transit stop, which can be used in analysis. On the contrary, AVLs do not provide that. Many transit systems, including the city of Nashville, do not have APCs on buses and use automatic vehicle location (AVL) data to estimate the arrival and departure time at bus stops and use the estimated data for bus delay prediction in real-time. The issue with this approach is that the lack of quality data results in worse predictive analytic performance.

Even for systems with APCs, real-time sensor systems can have many problems in the real world [48, 49], due to reasons, such as low networking bandwidth and delays in uploads. As a result, often GPS position data is noisy. A typical mechanism for handling noise is to normalize the data. However, normalization requires large data sets, often clustered around transit routes. This is helpful because the transit data of preceding buses may be used to create the models for the current trip on that route. However, if a city does not have high frequency operations across its routes, then such data is not available.

A number of strategies have been used to improve transit vehicle performance in the long-term. For example, controlled time points [50] have been used to distribute the available slack time across the route. Periodic schedule update [51, 52, 53] and dedicated bus travel lanes [54] have also gained attention. Commuters typically rely on timetables that are scheduled according to long-term patterns, and delay status updated in real time to plan transit trips. However, for short-term transit scheduling (e.g., one day before the travel day) when there is no real-time data available yet, commuters have no clue how to choose a proper route and departure time, and it's even more challenging for those who do not take public transportation on regular basis. It has been observed that there are recurrent traffic congestion [55] and transit delay patterns associated with events happening in the city. For example, for a given day with a big football game, the bus routes that bypass the area around the stadium are more likely to have a more significant delay than usual. Such contextual information (e.g., scheduled events and weather forecast conditions) could be utilized to get a better estimation of the expected delay.

Most of the prior research work is focused on long-term delay pattern analytics [56, 57, 58, 59] and real-time delay prediction [60, 15, 15, 61, 62]. Long-term analytics provides statistical decision supports such as the mean and confidence interval of route segment delay, which are useful for city planners and metro transportation authority (MTA) engineers to gain a deep insight into the actual transit performance for scheduling and planning. On the other hand, real-time delay prediction utilizes real-time data such as trip updates and

vehicle locations to inform engineers and commuters on the expected arrival time at bus stops. However, there are still gaps in effective prediction mechanisms that work in the short-term phase (i.e., hours or days ahead of the scheduled travel time when real-time data is unavailable) to help commuters make decisions by informing of possible severe delay.

Developing an effective short-term prediction for transit systems is currently an open problem to solve. Firstly, route segments typically have different delay patterns and are affected by the contextual factors in varying degrees (see Section 3.4.3.1 and Figure 3.9). Technically, multiple historical models can be built for different contextual features. However, since there are so many event types and their impacts on transit delay interact with each other and have varying spatio-temporal characteristics, advanced prediction models are needed to effectively integrate all contextual features. Additionally, due to the limited budget and computation resources, it may not be feasible to create and train a single prediction model for each separate segment in practice because of the high computation time. Furthermore, the transit frequency in mid-sized cities like Nashville is relatively low compared to the large metropolitan areas. Therefore, there is never enough data to train the prediction models and results in data sparsity issues. Also, data samples with known contextual information are rarer compared with other samples[2] The biased datasets will produce challenges if we try to use regression or random forest based approaches.

Solution Approach. To address the lack of quality data for transit data analytics, yet make effective predictions for bus arrivals, we surmise that the Dynamic Data Driven Applications Systems (DDDAS) paradigm [63] holds promise as a solution approach. In DDDAS, both real-time and/or historical data is used to learn the model of the system that must be controlled, and subsequently a decision support system uses these learned models to make informed decisions and control the system in a feedback loop. This is the approach we utilize in this chapter. It integrates historical and streaming real-time bus location data from multiple routes for short-term delay prediction as well as long-term delay

[2]There are a limited number of football and hockey games for example.

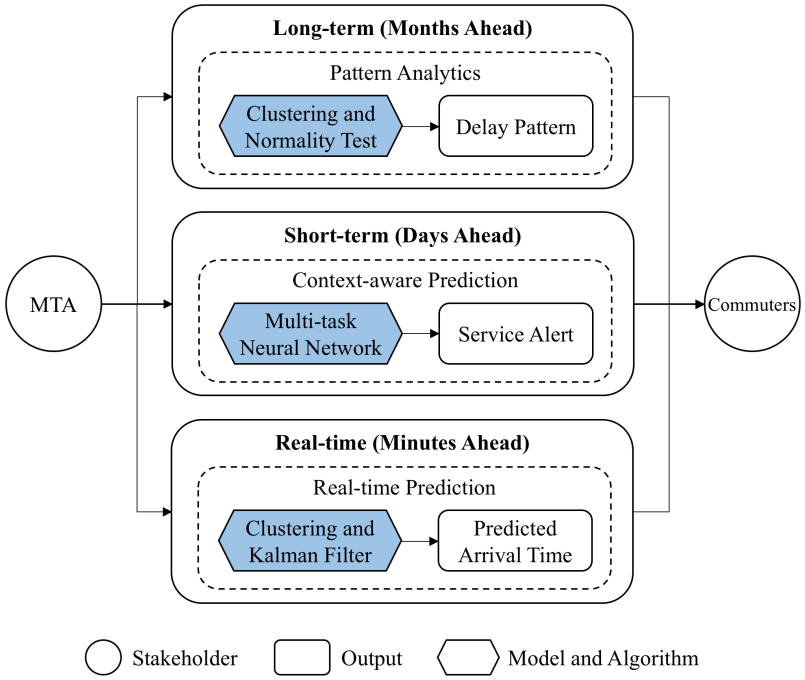


Figure 3.2: The proposed mechanisms provide context-sensitive analytics and predictions for three terms (i.e., long-term, short-term, and real-time) to help commuters plan transit trips.

pattern analytics. We also use the data feedback loop to provide results to city planners and end users.

Additionally, a short-term transit decision support system is being proposed that predicts severe delay days ahead to help users to schedule transit plans. Compared to providing just static schedules or historical patterns, the context-aware short-term delay prediction model can identify the severe delay that does not follow normal patterns days ahead of time and help commuters choose optimal routes, which gives them more confidence when choosing the public transportation. This system is being currently integrated into our transit decision support system called Transit-Hub [39, 41].

This chapter presents our research work on Transit-Hub and provides the following contributions to the study of real-time, short-term and long-term predictive analytics for public transportation (the mechanisms and solutions are illustrated in Figure 3.2):

- We provide an algorithm that generates shared bus route segment networks from standard General Transit Feed Specification (GTFS) datasets.

- We present a better real-time delay prediction model that combines clustering analysis and Kalman filters and uses real-time data from shared route segments. We show the efficacy of our short-term delay prediction model. When predicting the travel time delay of segments 15 minutes ahead of scheduled time, our model reduced the root-mean-square deviation (RMSD) by about 30% to 65% compared with a SVM-Kalman model [15]. The SVM-Kalman model that we used for comparison is a dynamic prediction model that combines SVM and Kalman filters, two of the most widely used models in bus delay prediction [64, 13, 60, 65].

- A generic tool-chain that takes transit feed (in standard and real-time GTFS format), forecasted weather condition, and time as input, is developed to provide expected delays and service alerts as output for short-term. A multi-task deep neural network architecture is presented that consumes contextual information in the augmented datasets and makes delay predictions for nearby segments in a bounding box all at once. A threshold-based mechanism is utilized to produce service alerts. Compared with single networks, the proposed multi-task learning architecture not only takes a shorter time to train but also reduces the risk of over-fitting to the limited training data. Utilizing the contextual event and weather features improves the performance of recall by 28% and F1 score by 13%.

- We illustrate how the analytical algorithms can be packaged into independently deployable and self-contained micro-services.

- We describe how the system's data feedback loop works to provide decision support to city planners by assisting Metro Transportation Authority (MTA) in identifying real-time outliers and optimizing bus timetables to improve bus services and availability.

Furthermore, we use the data collected over several months from one such transit system and study the effect of weather and other covariates such as traffic on the transit network delay. These models can later be used to understand the seasonal variations and design an adaptive and transient transit schedule as part of future work. Towards this goal, we also propose an online architecture called DelayRadar. The novelty of DelayRadar lies in three aspects: (1) a data store that collects and integrates real-time and static data from multiple data sources, (2) a predictive statistical model that analyzes the data to make predictions on transit travel time, and (3) a decision-making framework to develop an optimal transit schedule based on variable forecasts related to traffic, weather, and other impactful factors (not covered in this chapter). This chapter focuses on identifying the model with the best predictive accuracy to be used in DelayRadar. According to the preliminary study results, we are able to explain more than 70% of the variance in the bus travel time and we can make future travel predictions with an out-of-sample error of 4.8 minutes with information on bus schedule, traffic, and weather.

3.2 Related Work and Challenges

3.2.1 Related Work about Bus Delay Prediction

Statistical Models. The basic average models directly use the average delay from historical data as the estimated delay for future and are often constructed for performance comparison purposes. For example, Jeong et al. [66] developed a basic average model and found that the basic average model was outperformed by regression models and artificial neural network (ANN) models for bus arrival time prediction. The reason is that the basic average models only use historical data and perform simple average analysis, the model does not reflect real-time conditions and is limited by the consistency of route delay patterns.

Many researchers have conducted studies that utilize both historical and real-time bus

data. Weigang et al. [67] presented a model to estimate bus arrival time at bus stops using the real-time GTFS data. Their model contains two sub-algorithms to determine the bus speed using the historical average speed and the real-time speed information from GPS. Their main algorithm utilizes the calculated real-time speed to predict the arrival time. Sun et al. [68] proposed a prediction algorithm that combines real-time GPS data and average travel speeds of route segments.

Regression models are also used to explain the impact of variables for delay prediction. Since the variables in transit systems are correlated [69], regression models are typically limited to delay prediction. Patnaik et al. [70] presented a set of regression models that predict bus travel times on a route segment. The data they used is real-world data (number of passengers boarding, stops, dwell time and weather) collected by Automatic Passenger Counters (APC) installed on buses. They also found that weather did not have a significant effect on the prediction.

Kalman Filter Models. Kalman filters have been used widely for bus delay prediction because of their ability to filter noise and continuously estimate and update actual states from observed real-time data. Chien et al. [65] presented a dynamic travel time prediction model that used real-time and historical data collected on the New York State Thruway (NYST). Shalaby et al. [71] proposed a bus delay prediction model based on two Kalman filter algorithms: one for estimating the running time and another for estimating the dwell time at bus stops. Yang et al. [72] developed a discrete-time Kalman filter model to predict travel time using collected real-time Global Positioning System (GPS) data. Bai et al. [15] proposed a dynamic travel time prediction model that employed support vector machines to provide a base time estimate and a Kalman filter to adjust the prediction using the most recent bus trips on multiple routes.

Machine Learning Models. Artificial Neural Network (ANN) [73, 66, 60] and support vector machine (SVM) [64, 13, 60, 15] are two of the most popularly used machine learning techniques in bus delay prediction. For example, Jeong et al. [74] developed an

ANN model for bus arrival time prediction using Automatic Vehicle Location (AVL) data. Mazloumi et al. [75] used real-time traffic flow data to develop ANN models to predict bus travel times. Yu et al [60] proposed a machine learning model that used bus running times of multiple routes for predicting arrival times of each bus route and proposed bus arrival time prediction models that include Support Vector Machine (SVM), Artificial Neural Network (ANN), k-nearest neighbors algorithm (k-NN) and linear regression (LR).

3.2.2 Research Challenge 1: Handling Data Sparsity Issues

Prior work emphasized long-term and short-term transit data analysis and prediction. However, most of them, as mentioned above, focused on a single route and few noticed that many bus routes share segments with other routes. In 2011, Yu et al. [60] recognized that the data from multiple routes could help to improve the delay prediction. In 2015, Bai et l. [15] proposed a dynamic travel time prediction model that combines SVM and Kalman filter using multiple bus routes data. Even though these two works proposed the idea to solve sparsity issue, no automated mechanisms to identify shared route segments. On the contrary, our approach provided an efficient algorithm that generates shared bus route segment networks from standard General Transit Feed Specification (GTFS) datasets.

3.2.3 Research Challenge 2: Identifying the Temporal Aspects of Predictions

Many bus delay prediction techniques rely on recent travel time data to make predictions for the future. But they do not describe methods to identify differences in the transit delay data. When solving the shared-segment prediction problem, they only used the actual travel time of preceding buses and did not consider the scheduled time difference of separate bus routes. In fact, different bus routes usually have different travel demand and delay patterns. Also, the existing models that included the data of all recent preceding buses may contain outliers that should be excluded.

Furthermore, our experience tells us that transit delay models change over time. The recent delay data may not share the same delay patterns and trends as the predicted time.

Our solution combines clustering analysis, which is an unsupervised learning method, with Kalman filters. Travel time data instances are clustered into different groups according to travel time and time in the day. Then the cluster of data whose centroid is closed to now will be used and input to Kalman filter.

3.2.4 Research Challenge 3: Identifying the Significant Predictor Variables

The prior work in this area has been primarily focused on developing models for predicting delay as a short or long-term self-contained process. While some of the approaches have studied the effect of traffic on the travel delay, to the best of our knowledge, the effect of other environmental variables and effect of local events has not been extensively studied. Few studies explore and compare the significance of different factors on bus delay, such as visibility, temperature, traffic speed, etc. They arbitrarily select attributes according to the availability. Also, the selected attributes are assumed to be independent, but they are correlated with each other.

The prior work in this area has been primarily focused on developing models for predicting delay as a short or long term self-contained process. While some of the approaches have studied the effect of traffic on the travel delay, to the best of our knowledge the effect of other environmental variables and effect of local events has not been extensively studied. In this chapter, we present models to understand the effect of weather and traffic speed on travel delays.

3.2.5 Research Challenge 4: Predicting Transit Delay in Short-term Using Predictive Contextual Information

The number of available historical data samples for each transit route and segment depends on the service frequency. However, compared to the large metropolitans, mid-

sized cities like Nashville suffer from data sparsity issue since the frequency of bus services is relatively low. Furthermore, the availability of event information is also limited and constrained by the manual data collection. This limitation of contextual information (games for example) will result in biased training data, which makes the data sparsity issue worse.

Machine learning methods have demonstrated superior performance in the transit domain [15, 73, 66, 13, 60]. However, the machine learning models in those works were trained separately for a particular metric by training a single model or ensemble models, which may suffer from insufficient training data in reality. Training deep learning models will be more challenging since they have multiple hidden layers and there are much more parameters to optimize. This often results in overfitting issues and make the models difficult to generalize to new data.

To solve the challenges, we utilize a data augmentation structure called shared route segment network [42], and adopt the idea of multi-task learning to develop multi-task deep neural networks. Neural networks, which have been studied in many related work [66, 73, 60], are effective in prediction and have the potential to be trained online. Compared with single models, our proposed multi-task neural networks can not only produce more data for each task (i.e., predicting travel time and delay for a route segment) and are faster to train, but also reduces the overfitting issues for individual tasks. The details can be found in Section 3.4.3.

3.3 Data Feedback

In this section, we present the heterogeneous data sources that the system is using and then describe how we integrate and manage the collected data.

This section shows how data feedback in the transportation Decision Support System can be used to help metro transportation authority (MTA) to identify real-time outliers and perform long-term delay optimization to improve bus services and availability.

Figure 3.3 illustrates the data feedback cycle. We utilize the multi-source data from

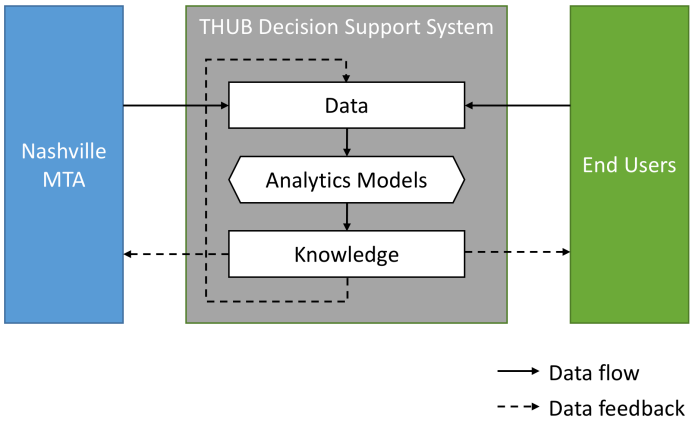


Figure 3.3: Proposed DDDAS loop in Transit-Hub transportation decision support system between MTA, Transit-Hub and end users.

Nashville MTA to conduct real-time and long-term data analytics, and the results can be sent back to them as feedback in different ways:

- *Metro Transportation Authority (MTA).* By doing long-term bus data analysis, our models can find the delay patterns that are associated with seasons, day of the week, and time of day. This feedback can be used by MTA to identify bottlenecks within routes and adjust the bus timetable or route layout accordingly. Also, by tracking the real-time bus data and comparing it with the historical delay patterns, we are able to find the outlier trips that deviate from the normal ones, which will be used to inform MTA to investigate and avoid these in the future.

- *End Users.* We are collecting anonymous usage and location data from application users. This data can be used to provide an alternative real-time data source for buses. If a user plans to take a bus that is full of people, the system can send notifications to advise him/her to take some other bus or routes. In addition, it can also help to optimize the bus route network and reduce rider walking distances as it shows the origins of users to the bus stops and helps MTA to identify areas with low/high transit service availability.

3.4 Our Approach

In this section, we present how we construct the long-term delay model, short-term travel-time model and arrival delay prediction model. In particular, we address Challenge 1 by creating a shared route segment network and utilize real-time data from multiple routes.

3.4.1 Building Model for Analyzing Long-term Delay Patterns

The section describes a long-term analytics model that constructs historical bus delay patterns at time points. In this model, clustering methods are applied to historical arrival delay and travel delay data.

3.4.1.1 Clustering Analysis

For each weekday, K-means clustering algorithm [76] is used to obtain the cluster of the delay data in accordance with the delay and time of the day by minimizing the within-cluster sum of squares (WCSS).

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 \tag{3.1}$$

where μ_i denotes the mean of all points in the cluster S_i.

Silhouete analysis [77] is an approach to measure how close each point is to others within one cluster.

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{3.2}$$

where for each data point i in the cluster, a_i is the average distance between i and the rest of data points in the same cluster, b_i is the smallest average distance between data point i and every other cluster, and $s(i)$ is the Silhouete score. We calculate the silhouette scores for 2 to 5 clusters derived from K-means algorithm to find the optimal number of clusters with the lowest silhouette score.

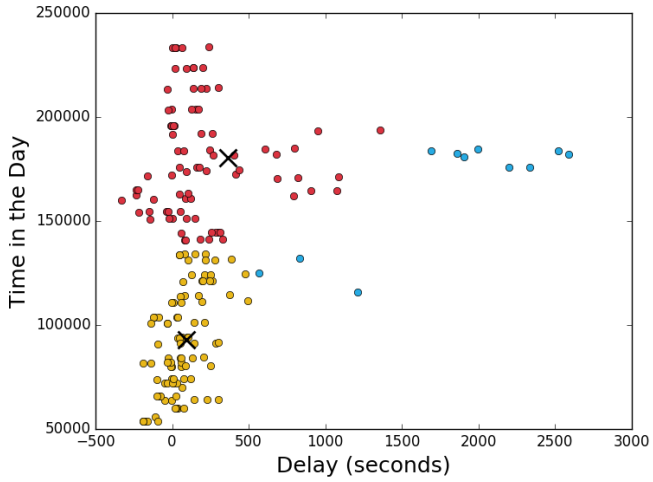


Figure 3.4: Cluster historical delay data according to the delay and time in the day at time point "HRWB" on route 3. The figure shows that there are two active delay patterns, one before and one after 2 PM. The blue dots are outliers identified by analysis in Section 3.4.1.3

The normal distribution of the clustered data helps to identify the typical delay patterns of previous buses, which can be given to users when they want an estimate for a future time, or if there is no real-time data available. The time point data is imported into the database at the end of each month. Then the data is stored according to weekday. We subsequently generate the clusters and normal distributions for all the route segments in each group. Meanwhile, the clustered data and normal distributions are cached and stored in the database. Thus, when we have to query the model, there is no need to run clustering analysis again.

Example Consider a time point 'HRWB' on route 3 in Nashville. The historical bus arrival delay data we select is for Wednesday, outbound direction, between June 1 2016 and June 30 2016 (for a total of 185 points). Figure 3.4 displays the delay data for a day during that month. In the figure, there are two obvious groups (yellow and red), one is between 5 AM - 2 PM and the other one is between 2 PM and 12 AM. The two groups reveal that there exist two different delay patterns which happen in the morning and in the afternoon separately. This information can be provided to end users to help them plan trips.

3.4.1.2 Normality Test and Analysis

The analytics is based on the assumption that historical delay data has a normal distribution [78]. In order to ensure this, we perform normality test on each cluster that we get in the previous step. We can calculate the confidence interval for long-term delay analysis from the distribution curve.

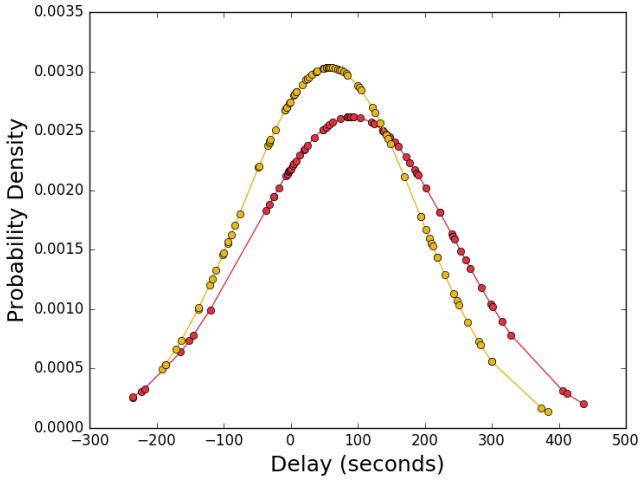


Figure 3.5: Distribution of the clustered historical delay data at time point "HRWB" on route 3

Example These are the two normal distributions in Figure 3.5 that we obtain after performing the normality test on the clusters generated from the data described in the previous example. The cluster for the delay in the afternoon has a higher mean value (92.0 seconds vs. 58.0 seconds) and a wider normal distribution curve, which indicates that buses on route 3 are more likely to be on time in the afternoon. In the afternoon, the 95% confidence interval of delay is between -60.4 seconds to 244.4 seconds while in the morning the 95% confidence interval of delay is between -73.5 seconds to 189.6 seconds (the negative seconds mean the buses are predicted to arrive earlier than scheduled time).

3.4.1.3 Outlier Analysis

In order to identify outliers from historical bus data, the first step is to generate the normal distribution for each of the clustered data groups described in the former sections. Since for a normal distribution where μ is the mean value and σ is the standard deviation, 95% of all data is within the confidence interval of [μ-2σ, μ+2σ], we define that the outliers are the historical data with delay greater than μ+2σ or less than μ-2σ in the distribution.

Example For the dataset mentioned in the previous two examples, there exist some outliers (blue points) in Figure 3.4. These outliers belong to the two clusters obtained from clustering analysis and are identified by outlier analysis. The outliers mostly emerged during rush hours in the morning and in the evening. One hypothesis is that during rush hours, there are more passengers and more traffic congestion on the route, which will increase the boarding time at stops and travel time on the road. Since our back-end server is monitoring the real-time transit feeds and in the meantime records real-time data, trips that have severe outliers and do not fit in the typical delay pattern can be easily detected and used for further investigation.

3.4.1.4 Bottleneck Identification

After mean delay patterns of all time points and all route segments are derived, we can then identify the bottlenecks along the routes by using those patterns. This also helps so that actions to optimize the route performance can be taken afterwards.

	Timepoints			
	WE23	WE31	HRWB	WHBG
Morning	116.90	127.71	93.14	443.52
Afternoon	121.03	146.28	114.48	545.49

Table 3.1: Mean value of the delay data distributions for 4 time points on route 3 in morning and afternoon in June.

There are 4 time points “WE23”, “WE31”, “HRWB” and “WHBG” on route 3 (travel-

ing away from downtown Nashville). Table 3.1 shows the findings that the typical arrival delay for "WHBG" is 443.52 seconds in the morning and 545.49 seconds in the afternoon. Considering the fact that the typical arrival delays for "WE23", "WE31" and "HRWB", timepoints before "WHBG", in the morning and afternoon are all below 150 seconds, we can draw the conclusion that the bus stops between "HRWB" and "WHBG" are the bottlenecks for route 3.

3.4.2 Real-time Data Integration

This section describes a short-term bus arrival delay prediction model that we have developed to address the challenges presented in Section 3.2. The model integrates real-time bus location data of shared route segments and combines clustering analysis and Kalman filters for delay prediction.

3.4.2.1 Utilizing Shared Route Segment Data

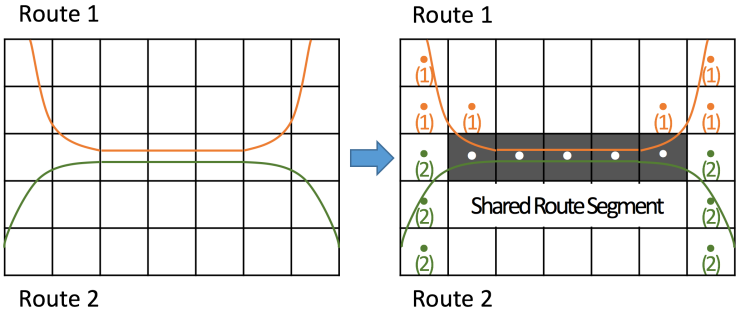


Figure 3.6: Finding shared route segments between two bus routes. The segment that contains the three center points is shared by route 1 and route 2.

Challenge 1 describes the issue that real-time bus data is not always available due to infrequency of buses. To address this challenge, the short-term delay prediction model in Transit-Hub creates a shared bus route segment network, and uses the real-time data from shared route segments for short-term predictive analysis.

Our prior work [39] was based on shared route segments, but at that time we used shared segments that were manually selected and we did not provide a solution to automatically identify shared route segments. In this chapter we present an algorithm to create a shared bus route segment network for all the existing routes in the city [79]. Also, the data that the algorithm uses is in standard GTFS format, so the algorithm can be easily applied to other cities that use the same data format.

A route segment is defined as a maximal part of bus route that is shared by a set of bus routes. In GTFS format, the physical path of bus routes is described using a sequence of coordinate points (in the *shapes.txt* file) on the map. If there are two segments from two bus routes that share the same sequence of coordinate points, then we can assume that the routes share that road segment. The outline of the algorithm to generate the share route segment network is described as below (The key steps are illustrated in Figure 3.6):

Input: *Static GTFS dataset.* Static bus and associated geographic information are loaded from database.

Output: *Shared route segment network.* Segment layout for each bus route is saved in the database.

Step 1: *Map grid initialization.* The Nashville map is divided into map grids of squares. The length of each square is about 8.97 meters, so each grid cell covers about 80.51 square meters on the map.

Step 2: *Route path re-sampling and smoothing.* The sequences of points in all bus routes are re-sampled to the centers of grid cells if the point is covered by the cell. Also, if the distance between adjacent points in the sequences is larger than the width of a grid cell, points will be interpolated to fill the cells that are missing points. The re-sampled points of each route are cached in the database for determining the shared route segments in the later step. As shown in Figure 3.6, the paths of route 1 and 2 are re-sampled to the center points of grid cells.

Step 3: *Calculating segments for bus routes.* Each cell is tagged by every route that

uses that cell. If a cell contains tags from multiple routes then it becomes part of a new shared segment. For example, the three-point segments in Figure 3.6 are shared by route 1 and 2, so this segment is marked as a shared route segment. New segments are checked to make sure no duplicated segments are generated.

Step 4: *Segment length limitation.* Any segment that has a length that is greater than 1 mile is divided into smaller segments because our model is based on the assumption that the travel delay within each segment is equally distributed, and hence the division of larger segments into smaller ones will satisfy this assumption and reduce prediction error.

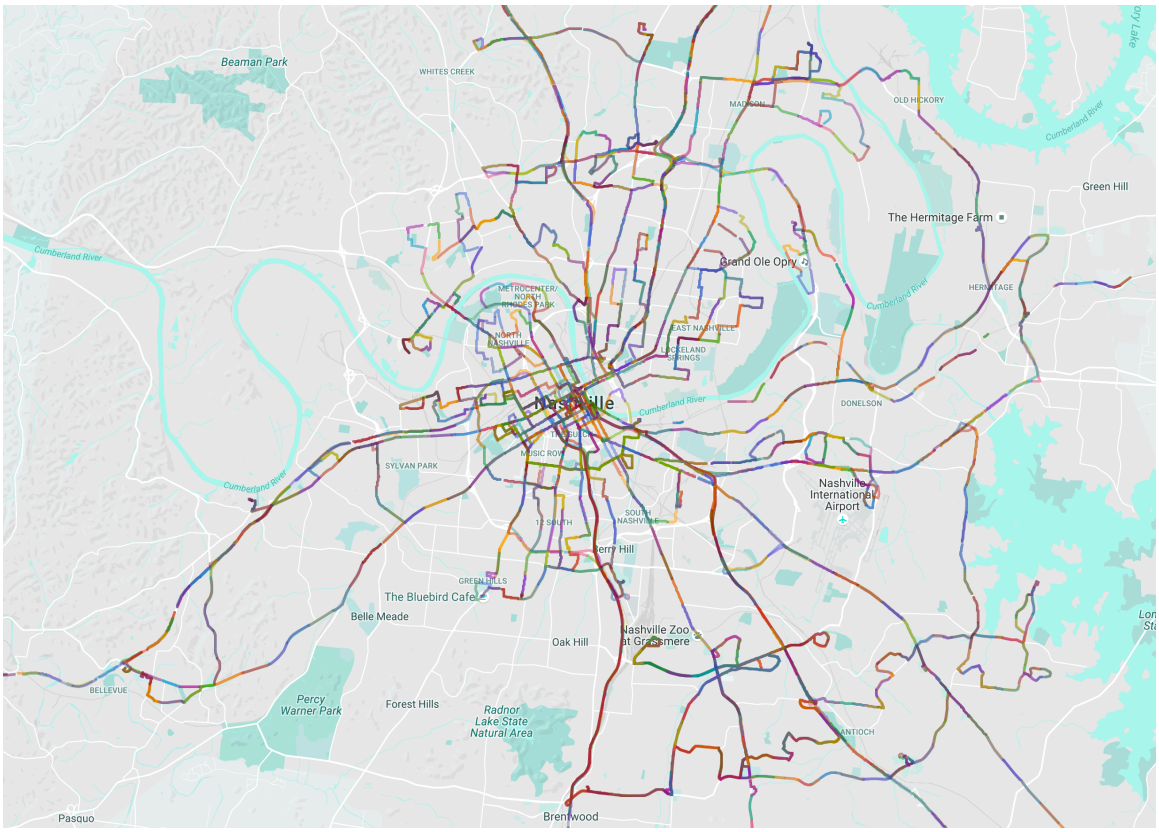


Figure 3.7: Generated shared bus route segment network in Nashville. The lines with different colors represent the 5139 shared route segments in all 57 bus routes in the network. The length of the segments are limited to less than 1 mile.

Using Nashville's static GTFS (version of March 9, 2016), we generated a shared route segment network shown in Figure 3.7. The 57 bus routes in Nashville city were divided

into 5139 segments. The lines in different colors show different route segments. Since the static bus schedules are updated regularly by MTA, the shared route segment network should be updated when new schedules are released.

There are many benefits to using real-time data of shared route segments, such as: (1) Utilizing the real-time data from other routes can greatly increase the volume of data that are available for short-term delay prediction analysis. For example, the route 3 in Nashville from White Bridge to Downtown has a schedule interval of 40 minutes at holiday and weekends. Only using route 3 data means the most recent data is at least 40 minutes old, which is not recent enough to predict the currently delay on route 3. (2) The length of each segment in the network can be controlled by the one-mile limitation mentioned in the last step of the algorithm Since the delay pattern varies along a bus route, segments with longer length are divided by the algorithm to produce more accurate analytics results. (3) By creating a shared route segment model, the divide and conquer design pattern is used. Individual and self-maintained microservice model can then run for each of the segments concurrently.

3.4.2.2 Estimating the Arrival Time at Bus Stops

Since the actual arrival time at bus stops are not included in the real-time GTFS feed in Nashville, we integrate the real-time bus location data and the static bus stop locations to estimate the arrival time of buses.

From the real-time bus location feed, we can get the bus location and timestamps in the following array format: $[(t_1, d_1), ..., (t_k, d_k), ...]$. Because the update rate of the original data varies from seconds to minutes, we first aggregate the collected data into 1-minute average data using sliding time windows. Then, we assume that bus speed is approximately the average of the two adjacent data points and apply the following equation to calculate the

bus arrival time at stops:

$$t_{stop} = t_{k-1} + (t_k - t_{k-1}) \frac{d_{stop} - d_{k-1}}{d_k - d_{k-1}} \tag{3.3}$$

where t_{stop} denotes the estimated arrival time, d_k is the bus's distance from the current location to the first bus stop of the route along the route path at time t_k. Also, $d_{k-1} <= d_{stop} < d_k$.

3.4.2.3 Updating the Travel Delay Prediction Using K-means Algorithm and Smoothing Filter

Excluding the outliers. If the travel time of a preceding bus differs greatly from other preceding buses, we consider this point an outlier and exclude it from the model computation.

To identify the outliers from the data, we employ K-means algorithm to cluster the preceding bus data according to travel time and time in the day. The Silhouette analysis that was introduced in equation 4.6 is also used here to find the optimal number of clusters. We choose the cluster whose time of day is closest to the current time. The data points from that cluster are smoothened through the filter described in the next section and used as an estimate for the current travel time on that segment.

Smoothing the preceding bus data. By comparing the travel time of preceding buses and the scheduled travel time within the route segment, we compute the travel delay of the preceding buses in the segment. The travel delay data is then through a filter to eliminate noise and predict the segment's current travel delay. The state transition equation is:

$$x_k = x_{k-1} + \omega_{k-1} \tag{3.4}$$

where the state variable x_k denotes the time step for which the travel delay needs to be predicted, ω_k denotes the zero mean normal distribution noise with covariance Q_k.

The observation equation used is:

$$z_k = x_k + v_k \tag{3.5}$$

where variable z_k represents the observation of delay at time step k. v_k represents the zero mean Gaussian distribution observation noise with covariance R_k. ω_k and v_k are assumed to be independent.

3.4.2.4 Example

In this section we use an example to explain the workflow of Transit-Hub multi-timescale analysis services. Figure 3.8 illustrate a common scenario where a bus b_1 is running along a bus route r_1 and the system needs to predict on request, the expected delay for a bus at stop s_i:

1. *Creating shared route segment network.* From the figure we can see that routes r_1 and r_2 are divided into 5 segments: seg_1, seg_2, seg_3, seg_4, seg_5. The segment seg_2 is shared by the routes.

2. *Getting preceding buses using static bus schedules.* From the static bus schedules we find that there are many buses (b_2, b_3, etc.) from route r_1 and r_2 that have passed through segment seg_3

3. *Estimating travel time of the buses in segments.* Preceding buses' travel time can be estimated using the collected real-time bus location data.

4. *Predict travel delay in segments.* The data from recent buses are clustered by travel time and time in the day. The group of data whose mean value (time in the day) is closed to the current time will be smoothed with a Kalman filter.

5. *Getting arrival delay at bus stop.* The sum of the delays for each segment between the current bus position and the target stop s_n is the model's prediction for arrival.

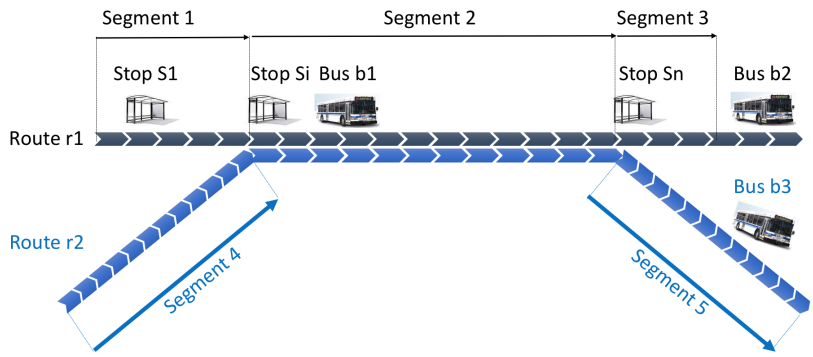


Figure 3.8: Use Case: Example of using shared route segments to predict a bus's delay at a bus stop

3.4.3 Short-term Context-aware Delay Prediction

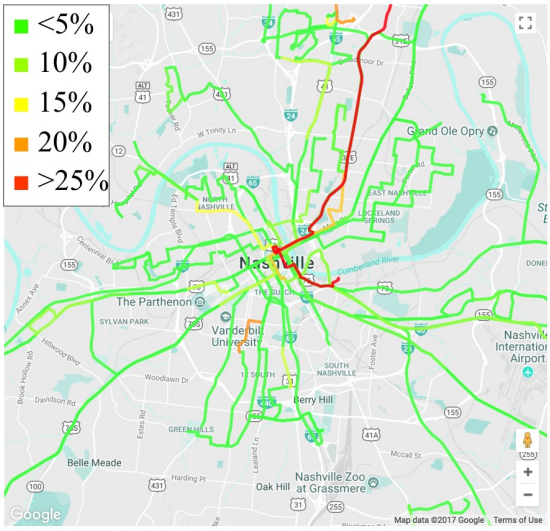
3.4.3.1 Motivating Example

To unveil the delay patterns associated with events, an analytics study using real transit and sports game data is conducted. The days between Sept. 1, 2016 and Jan. 1, 2017 are selected as the study period and the start time, end time, and attendance of eight football games occurred in the period are collected manually. We divide the time period before football games into four one-hour time windows ([-4, -3], [-3, -2], [-2, -1], [-1, 0]) and compare average bus delay on bus route segments between game days and non-game days using the following equation:

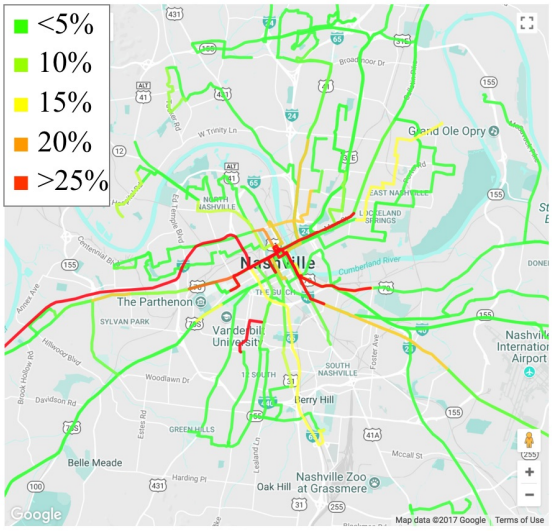
$$D_{PI} = max(avg(\frac{TT_{GD} - TT_S}{TT_S}) - avg(\frac{TT_{NGD} - TT_S}{TT_S}), 0) \tag{3.6}$$

where D_{PI} = delay impact, TT_{GD} = actual travel time on a game day, TT_S = scheduled travel time, TT_{NGD} = Actual travel time on a non-game day.

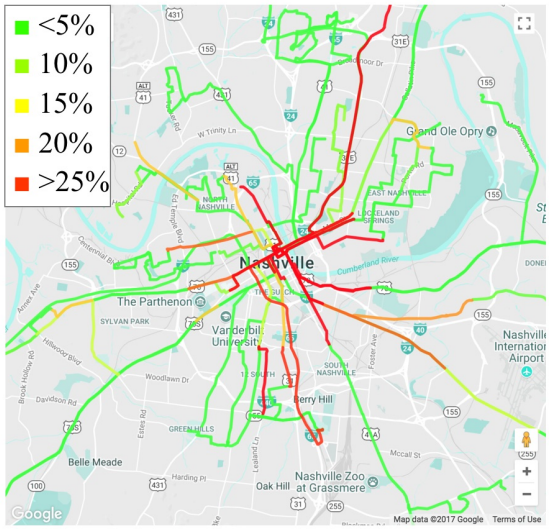
The results are visualized using heatmaps (see Figure 3.9). Generally, there are two patterns: (1) route segments have more delay as the time is closer to the game start time, (2) route segments that are closer to the stadium have more delay.



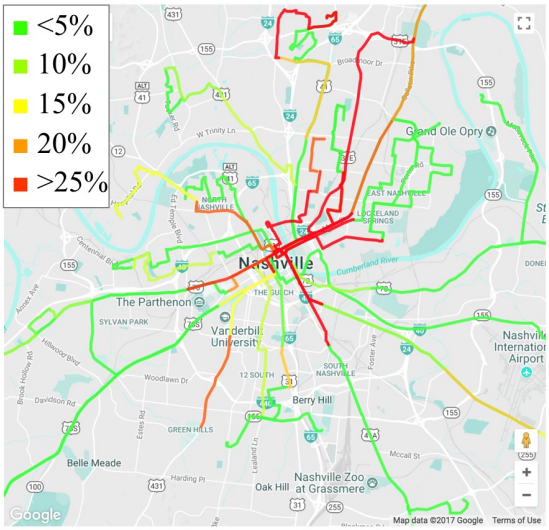
Time Window: [-4, -3]



Time Window: [-3, -2]



Time Window: [-2, -1]



Time Window: [-1, 0]

Figure 3.9: The impact of football games on travel delay of bus route segments in four one-hour time windows before 8 football games: (1) from 4 to 3 hours, (2) from 3 to 2 hours, (3) from 2 to 1 hour, (4) within 1 hour. The green colors are the baselines (i.e., average delay of bus route segments on non-game days). Other colors show the difference of average delay on games days compared to the baselines.

3.4.3.2 Feature Engineering

The features of different data sources are represented differently using numeric or one hot encoding according to their attributes as follows:

- Event Features: The information of scheduled events is represented by one-hot vectors. The dimensionality of the vectors is the size of effective time windows before and after events plus a numerical class for attendance and an additional class for no event. The effective time windows when events have impact on bus delay varies on different event types. For example, from the motivation study it was found that on average the impact of football games on bus delay starts as early as 4 hours before and as late as 4 hours after games, so the length of the feature vectors for football games is ten (i.e., no football game, attendance, and eight time windows). An example is illustrate in Figure 3.10.

- Weather Features: The forecasted weather conditions contains seven features: temperature, nearest storm distance, humidity, ozone, pressure, wind speed and visibility. Each weather condition sample is converted into a seven-dimensional vector of numerical features.

- Time Features: Time features contains two classes: time of day and day of week. The 24 hours in a day is divided into 48 half hours and time of day is represented using one hot encoding of 48 classes. Similarly, day of week is represented using a feature vector of 7 classes.

3.4.3.3 Multi-task Neural Networks

Deep learning techniques have gained great success in various fields, such as neutral language processing (NLP), image processing, information retrieval (IR), among others. Researchers start to develop deep learning techniques for transportation [80, 81, 82]. Compared with existing studies which focus on real-time delay prediction and rely on real-time

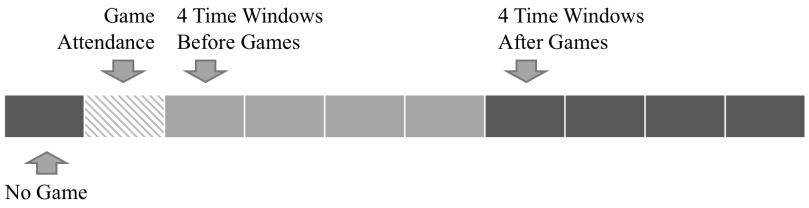


Figure 3.10: The one hot encoded feature vector for football games.

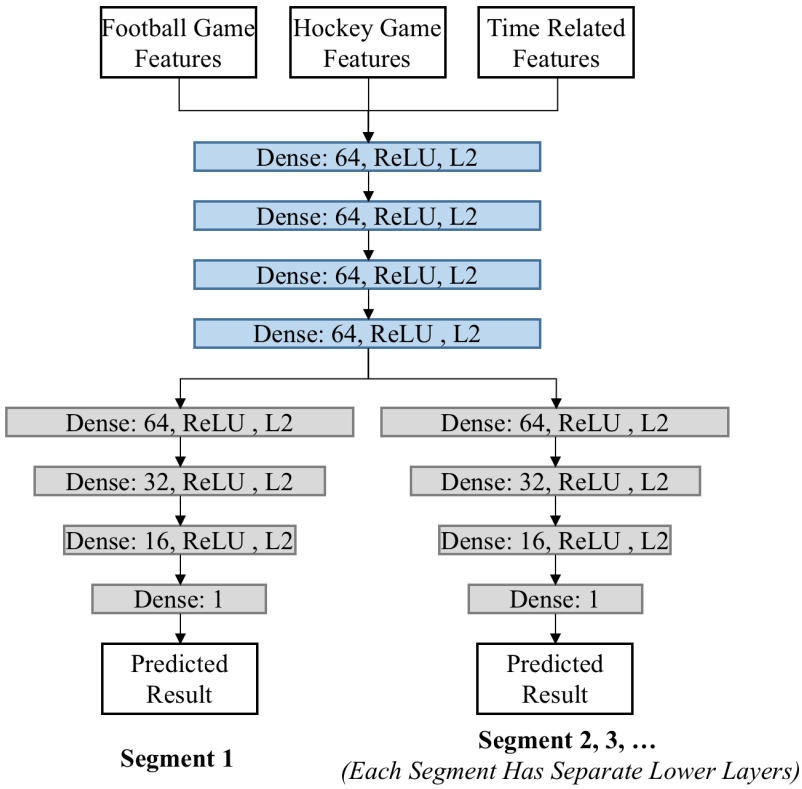


Figure 3.11: Proposed Multi-task Neural Network. Blue blocks are shared layers and gray blocks are independent layers for different segments.

data feeds (e.g., transit and traffic), our model only assumes the availability of forecasted weather conditions and information of scheduled events that have proved to have significant impact on transit delay.

Multi-task learning (MTL) share representations between related tasks. By leveraging the domain-specific information contained in the training signals of related tasks, MTL improves the generalization ability on original tasks [83]. In the transit domain, the travel delay on road segments in a nearby area is usually impacted by the same events at the same time and show similar patterns. Therefore, we aggregate the original tasks of predicting delay for nearby segments together in a multi-task learning architecture. We developed multi-task deep neural networks to get more data for training since they leverage supervised data from multiple nearby road segments. Furthermore, the use of multi-task networks can also reduces overfitting to specific tasks and better generalizes to new data [84].

The architecture of the proposed multi-task neural network is shown in Figure 3.11. We apply the approach of hard parameter sharing in the neural networks. Generally, it consists of upper layers that connect directly to input feature vectors and are shared across different segments, and lower layers that are specific to different segments. When training the models, Adam algorithm [85] is used for optimization and mean square errors are used as loss functions. The architecture greatly reduces the risk of overfitting and it is generally faster to train and can predict for multiple segments at the same time.

3.4.3.4 Service Alert Generation

The predicted delay can be utilized to decide whether a service alert can be generated for each route. Our system collects the historical travel times for each segment and sends out a service alert if the predicted delay is larger than 90th percentile in the data.

3.4.4 Significant Predictor Identification

Besides bus data, there are also many contextual and environmental variables impacting the bus delay. In order to identify the significant predictors in estimating bus delay, some joint work [62] has been done with Aparna Oruganti, who is a master student in Civil Engineering at Vanderbilt University. We consider regression and tree-based family of models. Regression models have been studied [86, 70] but to the best of our knowledge, there is no work using tree-based models to solve this problem. All models were trained and validated using the historical bus, traffic and weather datasets. In the end, the models with the highest fitness and predictive accuracy in the validation are analyzed to find the significant predictors.

Modeling Approach. We first build a series of multivariate linear regression models, in which the travel time is the outcome variable, and is modeled as a linear function of the predictors. Linear regression models can be extended to account for non-linear relationships between the predictors and the outcome variable through polynomial regressions [87], so we built several linear models that considered different combinations of the predictors. They can account for correlations among the predictors through the consideration of interactions between the predictors. We also train random forests using the same outcome variable and predictors because random forests are able to capture the non-linearity in the data by dividing the space into smaller sub-spaces.

Evaluation. We use two metrics to evaluate the model performance: (1) root mean squared error (RMSE) [88] for assessing the predictive accuracy, (2) R^2 [89] for assessing the goodness of fit. 10-fold cross-validation is used to validate the model ability in predicting using new data.

Results. We identify that the most significant predictors for bus delay prediction include the traffic speed, the visibility, the wind speed, the nearest storm distance, the speed limit, with an interaction term between the traffic speed and the scheduled travel time. Other predictors that are not included in the list can be expressed through these predictors.

3.5 Deployed Architecture

In this section, we describe the implementation architecture for the short term online delay prediction service, which addresses problem concerning scalability and availability.

Microservice deployment is an application architectural pattern where independently deployable and self-contained services can work together, which may be more suitable for complicated web applications [90, 91, 92] Microservices communicate with each other via lightweight network mechanisms, such as using REpresentational State Transfer (REST) API, message broker, etc. Figure 3.12 illustrates the overall architecture of the Transit-Hub analytics.

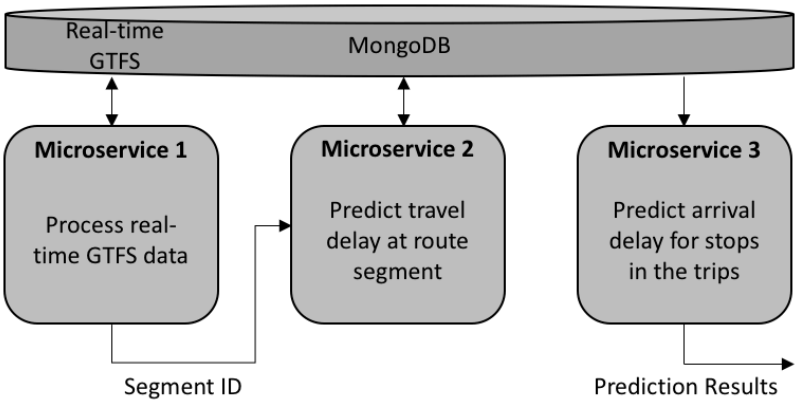


Figure 3.12: Microservice architecture of Transit Hub back-end analytics services

Microservice 1: *Smoothing the real-time GTFS data.* Microservice 1 first cleans the raw real-time GTFS data by removing the duplicate and missing data, and then re-sample it to estimate the bus arrival time on bus routes. This microservice tracks real-time bus location and when a new bus travels through a route segment, it will inform Microservice 2 which updates the travel time delay for this route segment. Microservice 1 is activated by a scheduler every 5 minutes.

Microservice 2: *Predicting arrival delay at segments.* Microservice 2 collects the data processed by Microservice 1 and employs short-term delay prediction (Section 3.4.2)

to update the estimated delay for the route segments. When Microservice 2 receives a prediction update request for a route segment, it wakes up and runs the prediction process to update the travel delay prediction for that route segment.

Microservice 3: *Predicting arrival time at bus stops.* Microservice 3 combines the current delay of all buses and the predicted travel delay for all route segments to produce the arrival delay prediction at all bus stops for all routes. This microservice is activated every minute and stores the prediction results in the database. Note that Microservice 2 runs per route segment whereas service 3 runs to update the arrival time for all routes.

Representational state transfer (REST) API and message broker are two of the popular approaches for providing a communication mechanism between microservices. The REST approach is synchronous by default and uses DNS or a registry for service discovery, and supports load balancing by using software like Ribbon [93]. The message broker is an asynchronous mechanism, which uses queues to manage message queues and can achieve load balancing very easily. Asynchronous message passing is a better choice for microservices because: (1) the individual microservice that sends a message will not be blocked before the other microservice responds; (2) using asynchronous communication can help to reduce unnecessary duplicate computation. For example, in our architecture, microservice 1 is continuously sending the IDs of route segments that need to update prediction to microservice 2. If we find that there are two identical segment IDs in the message queue, then the duplicate can be removed to avoid duplication of work. Based on these considerations, we use RabbitMQ [94], which is a

The microservices are deployed on an OpenStack [95] cloud operating system. We created a *m1.large* nova computing instance for the microservices which has 4 virtual CPUs, 8*GB* RAM and runs Ubuntu 14.04 (LTS). The microservices all together use 10.9% CPU resources and 28% RAM on average. The performance and resource consumption of the microservices will not be affected by user interactions. They run separately and repeatedly in the back end and store analysis results for later use. When an end user sends a prediction

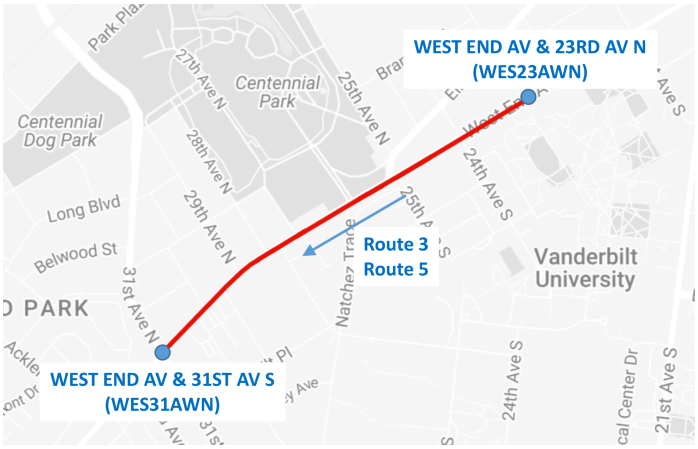


Figure 3.13: Studied road segment shared by route 3 and 5

request for a route, an independent service in the system will fetch the prediction results from the database and provide the information to the end user.

3.6 Prediction Performance Evaluation

This section presents experimental results from Transit-Hub's real-time delay prediction model. These results empirically evaluate Transit-Hub's bus travel time delay prediction ability against a SVM-Kalman model [15] using real-time data collected in Nashville. Compared to the SVM-Kalman model, our model takes the scheduled time of preceding buses into consideration, and since we are clustering the data of preceding buses according to time of day and delay, only clusters with an average time of day close to the current time of day will be used. We also evaluate how well our model predicts arrival delay comparing it against real-world data.

3.6.1 Experiment 1: Evaluating the Real-time Travel Time Delay Prediction

The first experiment is designed to evaluate Transit-Hub's ability to predict travel time delay, using its prediction model and comparing against other prediction models using the same real-world data.

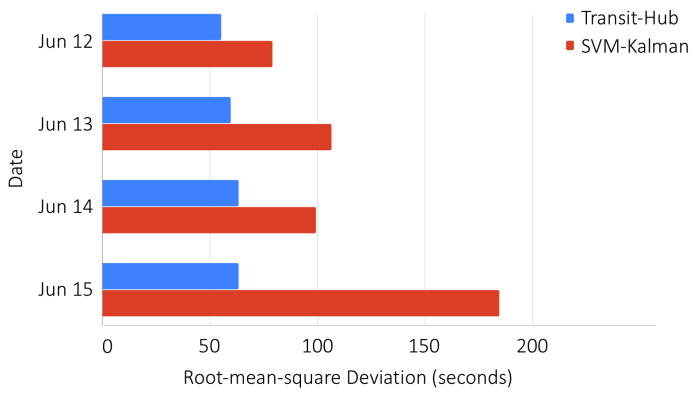


Figure 3.14: RMSD of travel time delay prediction for each day when comparing the Transit-Hub model with the SVM Kalman model proposed in 2015. Our model outperforms the SVM-Kalman model: (1) RMSD values are smaller (2) it shows less variation on different days.

Experiment Setup. Routes 3 and 5 are two of the major bus routes in Nashville. As shown in Figure 3.13, they share the same route segment between time point WES23AWN and time point WES31AWN along West End Avenue. We select this route segment of route 3 and 5 towards WHITE BRIDGE to test our proposed model.

The data used in this experiment is the real-time and static GTFS data for routes 3 and 5 that we collected from Nashville MTA in June 2016. We divide the data into two parts: a training dataset and a validation dataset. The training dataset contains bus data from June 6^{th} to June 12^{nd} and the validation dataset contains data from Jun 13^{rd} to Jun 15^{th}. Our model and the SVM-Kalman are evaluated using the same validation dataset. From our previous paper [39] we learned that only data 120 minutes old or newer is important for real-time delay prediction. Therefore, in this experiment we use the data for buses in the past 2 hours.

Comparing with a SVM-Kalman Model. In order to evaluate the performance of the proposed short-term delay prediction model, we chose and implemented a dynamic SVM-Kalman model that was proposed by Bai, et al. in 2015 [15]. The dynamic model consists of a support vector machines (SVM) model that uses historical data to estimate the current

travel time as a baseline prediction, and a Kalman filter model that uses real-time preceding bus data to adjust the base time. The features that they use in the SVM model include: (1) time of the day, (2) road segment ID, (3) weighted average bus travel time of preceding buses, and (4) the travel time of the preceding buses on the same route.

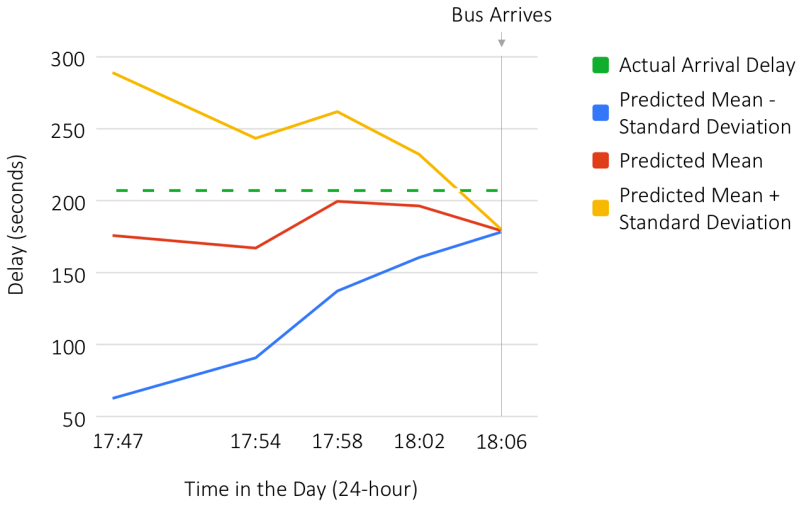


Figure 3.15: Arrival time delay prediction for a bus stop of a trip: (1) actual arrival delay, (2) predicted mean value - standard deviation, (3) predicted mean, (4) predicted mean value + standard deviation.

Results. Figure 3.14 shows the root-mean-square deviation (RMSD) of the travel time delay prediction results for three days in June. The RMSD of travel time delay is calculated using the following equation:

$$t_{ij}^{act\_tra} = t_j^{act\_arr} - t_i^{act\_dep} \tag{3.7}$$

$$RMSD = \sqrt{\frac{\sum_i^n (t_{ij}^{act\_tra} - t_{ij}^{pred\_tra})^2}{n}} \tag{3.8}$$

where i and j are indexes of the timepoints along the route, and $i < j$. Variable $t_i^{act\_arr}$ and $t_i^{act\_dep}$ represent the actual arrival and departure time at timepoint i, $t_{ij}^{act\_tra}$ and $t_{ij}^{pred\_tra}$ represent the actual and predicted travel time at the segment between timepoint i and j, respectively. n is the number of bus trips in the dataset.

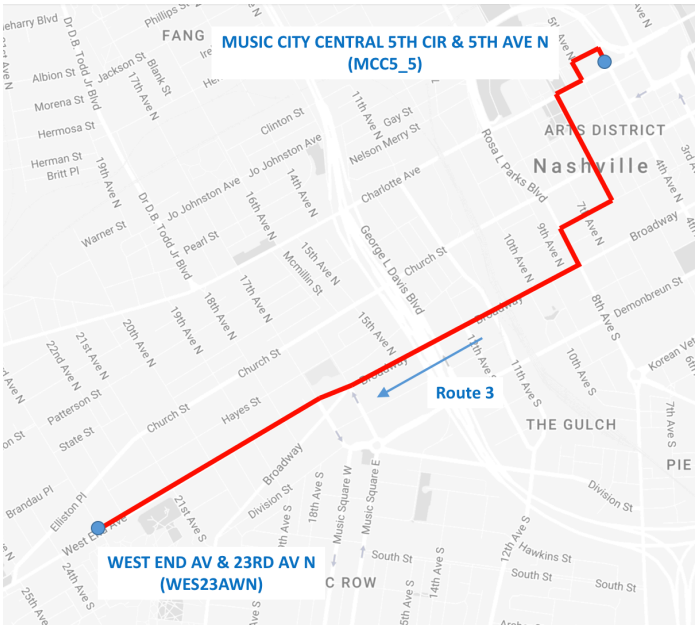


Figure 3.16: Studied segment of route 3 that starts from first bus stop (MCC5_5) to the 15^{th} bus stop (WES23AWN)

Since the SVM model ignores the differences that exist in the scheduled travel of preceding buses and the model does not exclude outliers, we expect our model to outperform the SVM model from [15]. The experimental results validate our hypothesis. When predicting the travel time delay 15 minutes ahead using collected data from Jun 12 to Jun 15, the RSMD of the our model is about 30% to 65% lower compared to the SVM-Kalman model.

3.6.2 Experiment 2: Evaluating the Real-time Arrival Time Delay Prediction

The second experiment is designed to evaluate the short-term prediction model's performance when the prediction horizon changes. For this experiment, we choose a trip from route 3 on June 14^{th} 2016. The studied segment is shown in Figure 3.16.

Results. Figure 3.15 shows the actual delay and the predicted arrival delay with confidence interval as the prediction horizon decreases from 19 minutes to 0 minutes (the time

just before the bus arrived).

Since our model integrates the predicted the travel delay in route segments and estimated arrival delay at the most recent bus stop that the bus passed, we expect the predicted confidence interval will become smaller and the error will decrease as the prediction interval reduces, i.e., as we make the prediction closer to the scheduled time of arrival.

This example shows that when predicting 19 minutes before the actual arrival time, the confidence interval is 226.5 seconds and the interval decreases to 2.1 seconds. We notice a 27.8 seconds difference between predicted delay and actual delay when the bus arrives, we attribute this to normal system variance.

3.6.3 Experiment 3: Evaluating the Short-term Arrival Time Delay Prediction

In this section, we evaluate the proposed model using two experiments by (1) comparing the multi-task neural networks with single models, and (2) comparing different feature vectors. Keras Python deep learning library with TensorFlow backend is used in the implementation [96].

Scenarios. The experiment scenario is illustrated in Figure 3.17. Between Oct. 1, 2016 and Jan. 1, 2017, 7 NFL football games held at the Nissan Stadium and 19 NHL hockey games at the Bridgestone Arena in Nashville. We selected the bounding box between coordinates of 36.175106, -86.760105 and 36.161903, -86.773335. Real-world bus, event and weather data within the time period and the bounding box is used in the experiments. The data between Oct. 1, 2016 and Dec. 11, 2016 is used for training. The data between Dec. 12, 2016 and Jan. 1, 2017 when there is at least an event happened is used for validation. Particularly, the following decisions are used to mark if a generated service alert is positive or not: (1) an output is considered to be positive if the delay is more than 90th percentile in the historical (training) dataset; (2) otherwise the output is negative. Recall performance, which indicates how many relevant items are selected in a classification task, is the key metric for evaluating our models since the metric was selected for this study to output noti-

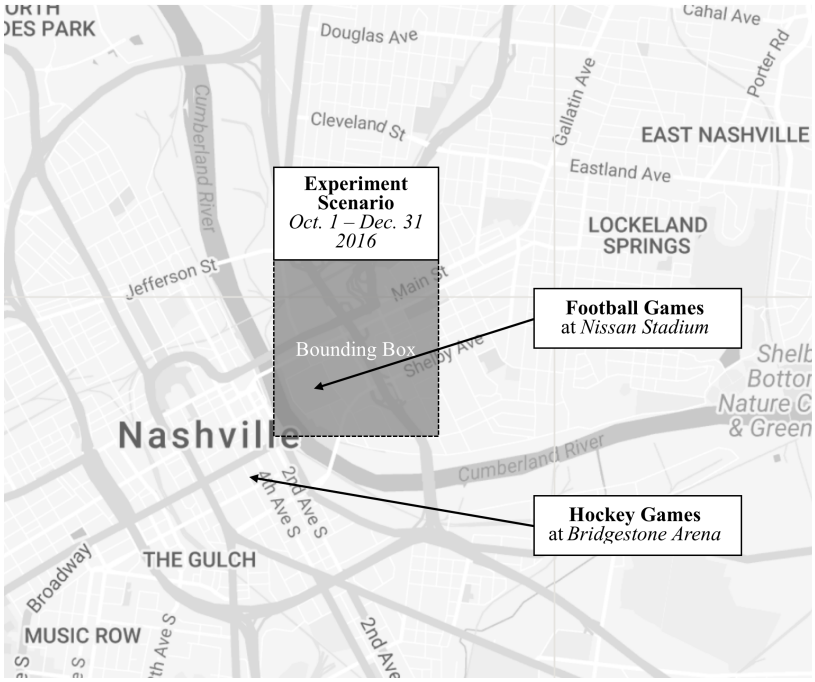


Figure 3.17: Experiment scenario. The selected bounding box is close to the Nissan Stadium where football games play and the Bridgestone Arena that hockey games play.

fies users to avoid severe delays as much as possible. The models are also evaluated using F1 score [97], which can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.

Comparing single models and multi-task learning models. We assume the multi-task learning models leverage supervised data by getting more training data for each segment and reduce overfitting for better generalization ability. To validate the assumption, we build the same neural network layers architecture for each segment and uses the same training and validating datasets to evaluate the single models. The difference is that the upper layers of the single models don't share parameters anymore and the parameters can only be optimized using data belong to the individual segments.

The root mean square error of the two models during training epochs are illustrated in Figure 3.19. The multi-task models are trained faster than single models. After 70 epochs, the single models' performance on the validating dataset becomes worse, which means it

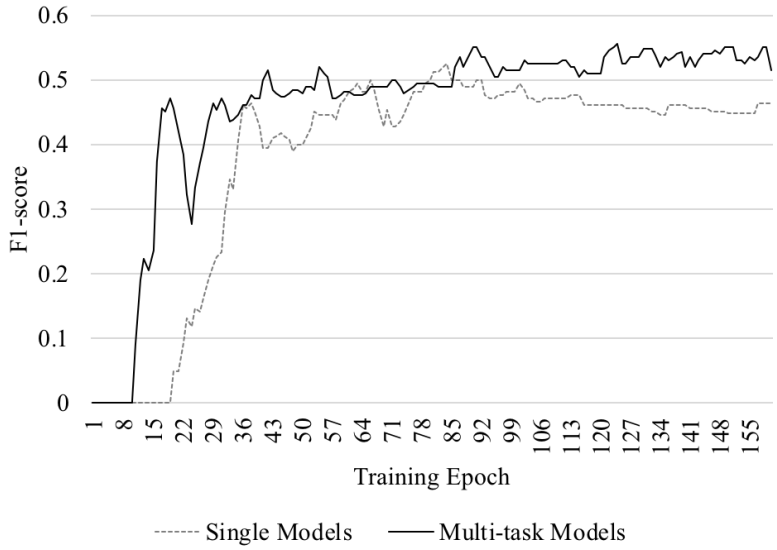


Figure 3.18: The F1 score between single models and multi-task models

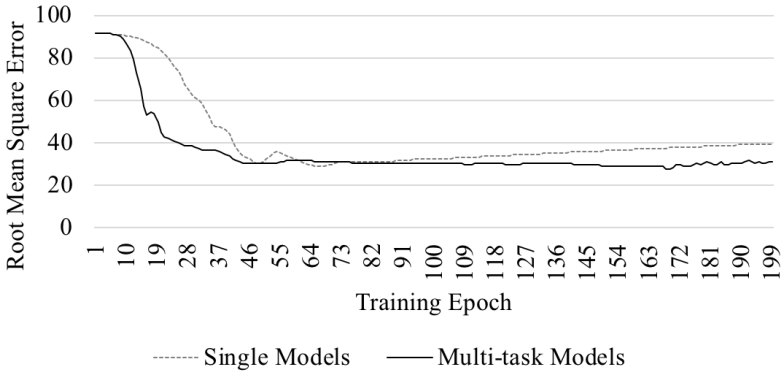


Figure 3.19: The mean square error between single models and multi-task models

starts to overfit the training dataset. On the contrary, our multi-task neural network doesn't have a significant overfitting problem. The F1 score of the multi-task neural network is also higher than the single models (see Figure 3.18).

Comparing different feature vectors using multi-task deep neural networks. In the second experiment, we compare the performance of the proposed multi-task deep neural networks using different feature vectors: (1) Time feature vectors: [day of week, time of day], (2) Contextual feature vectors: [football game time window, football game attendance, hockey game time window, weather conditions, day of week, time of day]. The recall and F1 score are calculated using the validation dataset. As shown in Figure 3.20 and Figure 3.21, compared with the time feature vectors, the contextual feature vector gets both higher recall (about 0.76) and F1 score (about 0.54), which means the model predicts more relevant (severe) delays and is more effective to warn commuters of real possible delays.

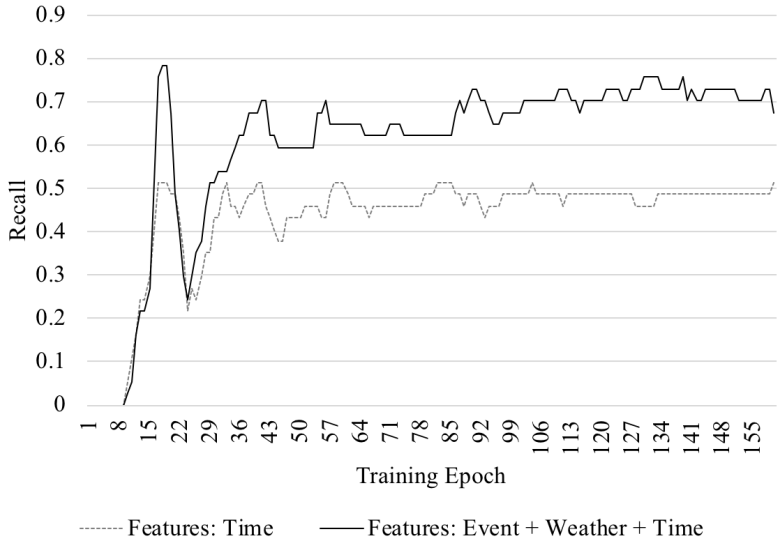


Figure 3.20: The recall using time feature vectors vs. using contextual feature vectors

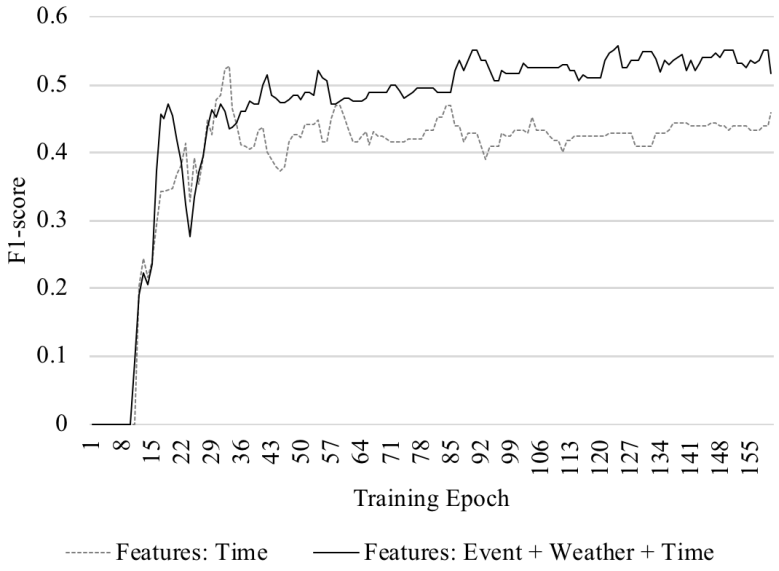


Figure 3.21: The F1 score using time feature vectors vs. using contextual feature vectors

3.7 Conclusion

In this chapter, we presented research on a DDDAS-enabled smart public transportation decision support system that illustrates and validates the mechanisms developed for long-term, short-term, and real-time predictive analytic services. Our long-term delay analysis service excludes the noise of outliers in the historical dataset and identifies the delay patterns of time points and route segments that are associated with different times of day, day of the week and seasons. The city planners can utilize the feedback data to optimize the bus schedules and improve rider satisfaction. Residents and travelers in cities like Nashville can also benefit from our short-term and real-time delay prediction services.

Publications. This work has been published in the following places:

- Sun, Fangzhou, Yao Pan, Jules White, and Abhishek Dubey. ”Real-time and predictive analytics for smart public transportation decision support system.” In Smart Computing (SMARTCOMP), 2016 IEEE International Conference on, pp. 1-8. IEEE, 2016. (34% acceptance rate)

- Sun, Fangzhou, Abhishek Dubey, Jules White, and Aniruddha Gokhale. ”Transit-hub: A smart public transportation decision support system with multi-timescale analytical services.”Cluster Computing(2017).

- Oruganti, Aparna, Fangzhou Sun, Hiba Baroud, and Abhishek Dubey. ”Delayradar: A multivariate predictive model for transit systems.” In Big Data (Big Data), 2016 IEEE International Conference on, pp. 1799-1806. IEEE, 2016.

- Shekhar, Shashank, Fangzhou Sun, Abhishek Dubey, Aniruddha Gokhale, Himanshu Neema, Martin Lehofer, and Dan Freudberg. ”Transit hub.”Internet of Things and Data Analytics Handbook: 597-612.

CHAPTER 4

ALGORITHMS FOR OPTIMIZING THE SCHEDULE OF PUBLIC TRANSIT CONSIDERING SEASONAL DELAYS

Bus systems are the backbone of public transportation in the US. An important indicator for the service quality of public transit is on-time performance at stops — the level of success of the service remaining on the published schedule. However, there are few stochastic optimization models that focus on optimizing bus timetables with the objective of maximizing the probability of bus arrivals at timepoint with delay within a desired on-time range. Furthermore, there lacks research work considering the monthly and seasonal variations of delay patterns. This chapter provides the following contributions to the study of transit on-time performance optimization: (1) we present an unsupervised clustering mechanism that groups months with similar seasonal delay patterns together, (2) we show how we formulate the problem as a single objective optimization task, and design greedy algorithms, genetic algorithms (GA) as well as a particle swarm optimization (PSO) algorithm to solve it, (3) we provide empirical results that compare the greedy, GA and PSO algorithms and present sensitivity analysis on the hyper-parameters of the heuristic algorithms, which could help other researchers and engineers as references. The content of this chapter has appeared in a conference paper [59].

4.1 Problem Overview

Emerging Trends. Bus systems are the backbone of public transportation in the US, carries over 47% of all public passenger trips and 19,380 million passenger miles in the US [98]. For the majority cities in the US who don't have enough urban forms or budget to build expensive transit infrastructures like subways, they rely on buses as the most im-

portant transit systems since bus systems have advantages of relatively low cost and large capacity. Nonetheless, bus is also one of the most unpredictable transit modes. Figure 3.1 illustrates the large variation of bus travel times (the bus trip departs at a specific time of the day on route 3 in Nashville). Our study found that the average on-time performance across all routes of Nashville bus system was only 57.79% (see Section 4.5.1). The unpredictability of delay has been selected as the top reason why people avoid bus systems in many cities [99].

Providing effective transit service is a critical but difficult task for all metropolis in the world. To evaluate service reliability, transit agencies have developed various indicators to quantify public transit systems through several key performance measurements from different perspectives [100]. In the past, a number of technological and sociological solutions have helped to evaluate and reduce bus delay. Common indicators of public transit system evaluation include schedule adherence, on-time performance, total trip travel time, etc. In order to track the transit service status, transit agencies have installed AVL on buses to track their real-time locations. However, the accuracy of AVL in urban areas is quite limited due to the low sampling rate (every minute) and the impact of high buildings on GPS devices. To have some basic controls during bus operation, public transit agencies often use time point strategies, where special timing bus stops (time points are special public transit stops where transit vehicles try to reach at scheduled times) are deployed in the middle of bus routes to provide better arrival and departure time synchronizations [101].

An effective approach for improving bus on-time performance is creating timetables that maximize the probability of on-time arrivals by examining the actual delay patterns. When designing schedules for real-world transport systems (e.g. buses, trains, container ships or airlines), transport planners typically adopt a tactical-planning approach [102, 103]. As such, scheduling decisions are usually performed a few weeks or months prior to the day-of-operations. Conventionally, metro transit engineers analyze the historical data and adjust the scheduled time from past experience, which is time consuming and error

prone. A number of studies have been conducted to improve bus on-time performance by reliable and automatic timetabling. Since timetable scheduling problem is recognized to be an NP-hard problem [104, 105], many researchers have employed heuristic algorithms to solve the problem. The most popular solutions include ad-hoc heuristic searching algorithms (e.g. greedy algorithms) [51, 52, 53, 56], neighborhood search (e.g. simulated annealing (SA) and tabu search (TS)), evolutionary search (e.g. genetic algorithm) and hybrid search [57, 58].

However, there are few stochastic optimization models that focus on optimizing bus timetables with the objective of maximizing the probability of bus arrivals at timepoint with delay within a desired on-time range (e.g. one minute early and five minutes late), which is widely used as a key indicator of bus service quality in the US [106]. For example, To quantity bus on-time arrival performance, many regional transit agencies use the range of [-1,+5] min compared to the scheduled bus stop time as the on-time standard to evaluate bus performance using historical data [106]. The actual operation of bus systems is vulnerable to many internal and external factors. The external factors include urban events (e.g., concerts, sporting events, etc.), severe weather conditions, road construction, passenger and bicycle loading/offloading, etc. One of the most common interval factors is the delay between two consecutive bus trips, where the arrival delay of previous trips causes departure delay of the next trip. Furthermore, there are monthly and seasonal variation in the actual delay patterns, but most transit agencies publish a uniform timetable for the next several months despite the variations. How to cluster the patterns and optimize timetables separately remains an open problem. Furthermore, heuristic optimization techniques have attracted considerable attention, but finding the optimal values of hyper-parameters are difficult, since they depend on problem nature and the specific implementation of the heuristic algorithms, and are generally problem specific. Furthermore, for cities that are expanding as people keep moving in, city planners have to invest more in public transit infrastructure and services, which results in longer and more complex bus routes. The long duration and

travel distance make effective and accurate schedule planning even more difficult.

Solution Approach. Heuristic searching algorithms have been studied in a wide range of the related work for various objectives (such as minimizing the cost of both user's and operator's, maximizing the simultaneous bus arrivals, etc.). However, to the best of the authors' knowledge, there are few stochastic optimizing models proposed focusing on optimizing the transit service quantified by on-time arrival range (e.g. one minute earlier and six minute later than advertised schedule) [106]. Furthermore, the importance of identifying monthly and seasonal delay patterns in helping timetabling is not recognized enough and integrated in the optimization processes.

In this chapter, we present research on bus on-time performance optimization that significantly extends our prior work [59]. To utilize the monthly and seasonal delay patterns, we apply outlier analysis and clustering analysis on bus travel times to group months with similar patterns together. The feature vectors used include mean, median and standard deviation of the historical travel times, which are aggregated by routes, trips, directions, timepoint segments and months. A greedy algorithm, a genetic algorithm, as well as a particle swarm optimization (PSO) algorithm are proposed to generate new timetables for month clusters that share similar delay patterns. Sensitivity analysis on choosing the optimal hyper-parameters for the proposed heuristic optimization algorithms are also presented. The overall workflow of the proposed optimization mechanisms is illustrated in Figure 4.1. Particularly, we provide the following contributions to the study of on-time performance optimization of transit systems:

- We describe an unsupervised analysis mechanism to find out how months with similar monthly and seasonal delay patterns can be clustered to generate new timetables.
- We present a greedy algorithm, a genetic algorithm, and a particle swarm optimization (PSO) algorithm to optimize the schedule time to maximize the probability of bus trips that reach the desired on-time range.

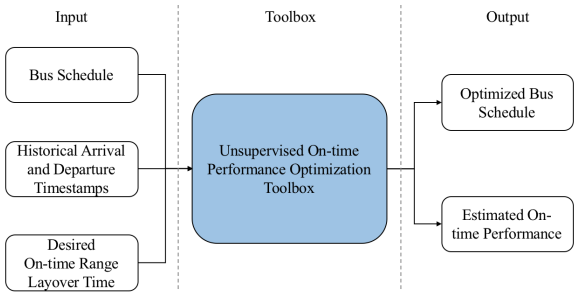


Figure 4.1: The proposed toolbox for bus on-time performance optimization. City planners use bus schedule, historical trip information and desired on-time range and layover time, and get outputs of optimized timetable as well as estimated on-time performance.

- Simulations of optimization performance as well as sensitivity analysis on the hyper-parameters of the GA algorithms and PSO are conducted. The results indicate the performance difference between the algorithms, and provide practical strategies to select optimal hyper-parameters.

4.2 Related Work and Challenges

4.2.1 Related Work about Transit Performance Optimization

This section compares our system with related work on transit timetable scheduling. A number of studies have been conducted to provide timetabling strategies for various objectives: (1) minimizing average waiting time [107, 108, 58, 109], (2) minimizing transfer time and cost [110, 111, 53, 57, 109], (3) minimizing total travel time [112, 56], (4) maximizing number of simultaneous bus arrivals [113, 114, 115, 116], (5) minimizing the cost of transit operation [117], (6) minimizing a mix of cost (both the user's and the operator's) [51, 52].

Friedman et al. [107] formulated a mathematical model of a general transportation network and presented a procedure to optimize bus departure times for minimizing the average waiting time of passengers by changing decision variables (i.e. bus departure time). Chakroborty et al. [110] first used genetic algorithms to develop optimal schedules for ur-

ban transit systems. The problem is formulated as a mathematical program that minimizes the sum of total time transferring from one route to another route for all transferring passengers and initial waiting time for all passengers at the origin. Pattnaik et al. [51] proposed a genetic algorithm for designing urban bus transit route network. Their research focuses on selecting a set of optimum route sets using a GA. Ceder et al. [113] proposed a mixed integer linear programming model to maximize the number of simultaneous bus arrivals at transfer nodes. Charkroborty et al. [52] developed genetic algorithm based procedures for route planning and scheduling. Eranki et al. [114] defined the synchronization problem by considering waiting times at transfer stops and developed a model to attain maximum synchronization. Ting et al. [117] applied a heuristic algorithm to minimize the total cost of transit network by optimizing the headways and slack times at transfer stops. Zhao et al. [111] presented a mathematical stochastic methodology to minimize transfer and user cost. Wong et al. [108] presented a mixed-integer-programming optimization model to minimize the interchange waiting times of all passengers. Yang et al. [53] proposed an improved genetic algorithm to optimize timetables that passenger transfer time is minimized using constraints of traffic demand and departure time and maximum headway. Szeto et al. [57] proposed a genetic algorithm for route design problem and a neighborhood search heuristic for bus frequency setting problem. Their goal is to reduce the number of transfers and the total travel time of the users. Ibarra-Rojas et al. [115] formulate the timetabling problem with the objective of maximizing the number of synchronizations of different bus lines and avoiding bus bunching along the bus network. Tilahun et al. [112] modeled single frequency route bus timetabling as a fuzzy multi-objective optimization problem using preference-based genetic algorithm. Nayeem et al. [56] presented a genetic algorithm based optimization model for maximizing the number of satisfied passengers, minimizing the total number of transfers and minimizing the total travel time of all served passengers. Hora et al. [58] applied a Mixed Integer Linear Programming (MILP) model to obtain robust bus schedules that minimize the differences between scheduled times and actual ar-

rival time. Their solution works on allocating the slack time of two subsequent stops. Wu et al. [109] developed a stochastic integer programming model to minimize the total waiting time cost of three types of passengers: transferring passengers, boarding passengers and through passengers. Later, Wu et al. [116] proposed a robust schedule coordination mechanism by combining planning and operating strategies. They addressed the stochastic travel time issue by adding slack time in scheduling and using a safety control margin in operation.

4.2.2 Research Challenge 1: Setting up the Transit Performance Optimization Problem

Transit performance optimization is well studied. However, to the best of the authors' knowledge, even though the on-time arrival range (e.g. one minute earlier and six minute later than advertised schedule) is widely used as a key transit reliability indicator by transit agencies for analyzing and timetabling in the United States [106], there no stochastic optimization models focusing on optimizing bus timetables to increase bus trips within the on-time performance range. In Section 4.3, we show how we formulate the problem as a single objective optimization task with constraints and set up the solution population for the genetic algorithm and particle swarm optimization (PSO).

4.2.3 Research Challenge 2: Clustering the Monthly and Seasonal Variations in Historical Arrival Data

Studying the historical travel time at segments can be an effective way to set bus timetables. However, existing work doesn't consider the monthly and seasonal variation in historical monthly data, and the variation can be utilized for better scheduling. Generating one timetable for all months may not be the best solution. The difference between existing approaches and our approach is that since traffic and delay patterns change over seasons and different times, we generate month clusters based on unsupervised learning and develop optimization models for these clusters. We evaluate the proposed mechanism via simula-

tion. The cluster-specific schedule is shown to further increase the on-time performance compared to generating one uniform timetable.

4.2.4 Research Challenge 3: Computing Efficiently and Accurately in the Optimization Solution Space

Transit performance optimization techniques rely on historical delay data to set up new timetables. However, the large amount of historical data makes it a challenge to compute efficiently. For example, Nashville MTA updates the bus schedule every 6 months but each time there are about 160,000 historical record entries to use. Moreover, the solution space is typically very large under constraints (e.g., sufficient dwelling time at bus stops, adequate layover time between trips, etc.). A suitable optimization algorithm is necessary for efficient and accurate computation. Since this is a discrete-variable optimization problem, gradient-based methods cannot be used and gradient-free methods need to be considered. A naive algorithm for discrete optimization is exhaustive search, i.e., every feasible time is evaluated and the optimum is chosen. Exhaustive search works for a small finite number of choices, and cannot be used for high-dimensional problems. Genetic algorithm [110, 51, 52, 111, 53], as well as particle swarm optimization [118] are used commonly in solving heuristic problems . Thus we consider applying genetic algorithm and particle swarm optimization (PSO) in the context. Section 4.4 describes the key steps of how we apply greedy, genetic and PSO algorithms to solve the timetable optimization problem.

4.3 System Model

In this section, we first describe the targeted optimization problem by using a concrete example, and then present how we formulate the problem as a single objective optimization task, and finally give necessary assumptions and primary notations.

Typically, transit delay are not only affected by external factors (such as traffic, weather, travel demand, etc.), but also caused by some internal factors. For example, the accumu-

Table 4.1: The scheduled time and recorded actual arrival and departure time of two sequential trips that use the same bus of route 4 on Aug. 8, 2016. The arrival delay at the last timepoint of the first trip accumulates at the first timepoint of the second trip.

		Timepoints			
		MCC4_14	SY19	PRGD	GRFSTATO
Trip 1	Scheduled Time	10:50 AM	11:02 AM	11:09 AM	11:18 AM
	Actual Arrival Time	10:36 AM	11:10 AM	11:18 AM	11:27 AM
	Actual Departure Time	10:50 AM	11:10 AM	11:18 AM	11:30 AM
Trip 2	Scheduled Time	11:57 AM	11:40 AM	11:25 AM	11:20 AM
	Actual Arrival Time	12:11 AM	11:51 AM	11:34 AM	11:27 AM
	Actual Departure Time	12:11 AM	11:51 AM	11:34 AM	11:30 AM

lated delay occurred on previous trips may cause a delay in consecutive trips by affecting the initial departure time of the next trip. In order to illustrate the problem context with simplicity and without generality, we take two sequential bus trips of route 4 in Nashville as an example (the scheduled time and the actual arrival and departure time recorded on Aug. 8, 2016 are shown in Table 4.1) to describe the optimization problem. On each service day, after a vehicle of the first trip arrives at the last stop (Timepoint GRFSTATO) with scheduled time of 11:18 AM, the second trip is scheduled to depart using the same vehicle from the same stop at 11:20 AM. On Aug. 8, 2016, the arrival time at the last stop (Timepoint GRFSTATO) of the first trip is exceptionally late for 9 minutes, which contributes to the 10-minute departure delay at the beginning of the second trip. Since the scheduled layover time between the two trips is only 2 minutes (between 11:18 AM and 11:20 AM), any large delay at the first trip is very likely to transfer to the next trip. Therefore, the optimization problem should involve a optimization process that considers not only the travel delay on segments, but also the improper lay over time between trips.

4.3.1 Problem Definition

Let $H = \{h_1, h_2, ..., h_m\}$ be a set of m historical trips of a given bus trip schedule b. Each trip passes a set of n timepoints $\{s_1, s_2, ..., s_n\}$. The *on-time performance* of the bus

trip schedule b can be expressed as :

$$P = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} I(h_i, s_j)}{m \times n} \tag{4.1}$$

where h_i denotes a historical trip and s_j denotes a timepoint on the trip. The indicator function $I(h_i, s_j)$ is defined as:

$$I(h_i, s_j) = \begin{cases} 1, & \text{if } d_{i,j} \in [t_{early}, t_{late}] \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

$$d_{i,j} = t_{h_i,s_j}^{arrival} - T_{h_i,s_j}^{arrival} \tag{4.3}$$

where $d_{i,j}$ is the actual delay that a bus from the historical trip h_i arrives at a timepoint s_j, t_{early} and t_{late} are two time parameters that transit authority has pre-defined to rate the schedule adherence of the bus at that timepoint. The goal of the schedule optimization problem is to generate new $T_{h,s}^{departure}$, such that the on-time performance is maximized. A list of primary symbols and definitions defined in the chapter can be found in Table 4.2.

4.3.2 Assumptions and Notations

To formulate the problem and facilitate the model development, some assumptions are made as follows:

1. *Assumption 1.* The proposed method assumes the availability of accurate arrival and departure timestamps in the dataset. Even though the proposed mechanisms use timepoint datasets for evaluation purpose in the case studies, for other cities where timepoints are not available, the mechanisms can easily adapt to other data sources, e.g, from buses equipped with automatic passenger counters (APC).

2. *Assumption 2.* The scheduled departure time at the first stop of each trip will not be changed. The scheduling of trip headways are constraint by budget and travel

Table 4.2: List of primary symbols and definitions used in the optimization problem

Symbol	Definition	Unit
$[t_{early}, t_{late}]$	the time window that the arrival delay on-time bus should satisfy within	min
h	a bus trip that departures at the same time in different days. In Table 4.1, there are two consecutive bus trips that scheduled to depart at 10:50 AM and 11:20 AM.	-
s	a timepoint	-
$t_{h,s}^{arrival}$	the actual arrival time at a timepoint s on trip h	min
$t_{h,s}^{departure}$	the actual departure time at a timepoint s on trip h	min
t_{h,s_i,s_j}^{travel}	the actual travel time between two adjacent timepoints s_i and s_j on trip h	min
$T_{h,s}^{arrival}$	the scheduled arrival time at timepoint s on trip h	min
$T_{h,s}^{departure}$	the scheduled departure time at timepoint s on trip h	min
$t_{s_j}^{dwell}$	the dwell time (in simulation) at timepoint s_j that caused by riders getting on/off	min
b	a bus schedule that defines the departure and arrival time at bus stops and timepoints	-
$v_{i,j}(t)$	the velocity of i^{th} particle in j^{th} dimension at iteration t in particle swarm optimization	-
$x_{i,j}(t)$	the position of i^{th} particle in j^{th} dimension at iteration t in particle swarm optimization	-
$p_{i,j}(t)$	the best position of i^{th} particle in j^{th} dimension till iteration t in particle swarm optimization	-
$p_{g,j}(t)$	the best position of the global best particle in j^{th} dimension till iteration t in particle swarm optimization	-
w	inertial weight component in a particle's velocity in particle swarm optimization	-
$c1$	cognition accelaration coefficient in a particle's velocity component in particle swarm optimization	-
$c2$	social accelaration coefficient in a particle's velocity component in particle swarm optimization	-
$r1$	random number between 0 to 1 used in particle swarm optimization	-
$r2$	random number between 0 to 1 used in particle swarm optimization	-

demand, which doesn't belong to the problem discussed in the chapter.

3. *Assumption 3.* The scheduled slack time between two adjacent bus trips that belong to the same block must be greater than or equal to zero minute i.e. $T_{s_1'} - T_{s_n} \geq 0$, where s_n is the last timepoint of the current trip and s_1' is the first timepoint of the next trip in the same block.

4. *Assumption 4.* The actual departure time at a timepoint should be greater than or equal to the scheduled departure time i.e. $t_s^{departure} \geq T_s^{departure}$.

5. *Assumption 5.* The scheduled departure time at a timepoint should be equal to the scheduled arrival time at the timepoint i.e. $T_s^{arrival} = T_s^{departure}$. How we handle dwell time at timepoints is described in Section 4.4.2.

4.4 Timetable Optimization Mechanisms

Optimizing bus timetables periodically to match the ever-changing travel delay patterns is an effective way to improve on-time performance. This section presents mechanisms to automatically optimize the arrival on-time performance at stops, which include: (1) a clustering analysis mechanism to group months with similar delay patterns together so that different timetables could be generated later for each month group, (2) a light evaluation mechanism to evaluate the expected on-time performance of bus schedules, (3) a greedy algorithm and two heuristic algorithms (i.e., genetic algorithm and particle swarm optimization algorithm) to update the scheduled arrival times of fixed schedule transit vehicles. The greedy algorithm chooses locally optimal travel time between timepoints and the optimized results are used as baselines. The heuristic algorithms are then compared, and their optimal hyper-parameters are explored by sensitivity analyses. Evaluation results are presented in Section 4.5.

4.4.1 Month Grouping by Clustering Analysis

When planning bus trips that depart at the same times on a specific weekday, MTA engineers often set the same schedule times along bus routes and assign the same identifier (e.g., $trip\_id$ in GTFS) to the trips. However, transit delays on road segments often show monthly and seasonal variations. These variations make it difficult for a trip's timetable to match the actual delay patterns over a long time. Thus planning a single timetable for all months is not an optimal solution. This section introduces a clustering analysis mechanism that groups months with similar transit delay patterns together and the results will later be used to generate separate timetables for each group. This section first describes the feature engineering to create effective vectors for each month, and then presents a clustering mechanism based on K-Means algorithm.

Feature Engineering. Effective features that matches the objective of clustering is important. We assume the monthly delay patterns can be represented by the mean, median and standard deviation that derived from historical delay data. Considering a bus trip consists of n timepoints, there are $n-1$ segments between the timepoints. The mean value μ, the median value m, and the standard deviation σ of the historical travel times for each timepoint segment in each month are integrated to generate feature vectors to represent the historical delay data distribution:

$$[\mu_1, m_1, \sigma_1, \mu_2, m_2, \sigma_2, ..., \mu_{n-1}, m_{n-1}, \sigma_{n-1}] \tag{4.4}$$

Month Clustering. Clustering is an unsupervised/supervised learning technique for grouping similar data. We employ k-means algorithms to identify the homogeneous groups where months share similar patterns. The trip data per month is first normalized and then clustered using feature vectors (in Equation 4.4) by K-Means algorithm:

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 \tag{4.5}$$

where μ_i denotes the mean of all points in cluster S_i. Determining the optimal number of clusters in a data set is a fundamental issue in partitioning clustering. For k-means algorithms, the number of clusters is a hyper-parameter that needs to be set manually. An upper bound is set in advance. Elbow [119], silhouette [77] and gap statistic [120] methods are popular direct and statistical methods to find the optimal number of clusters. Particularly, Silhouette analysis is employed in this study to measure how close each point is to others within one cluster. The silhouette score $s(i)$ is defined as:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{4.6}$$

where for each data point with index i in the cluster, a_i is the average distance between $data_i$ and the rest of data points in the same cluster, b_i is the smallest average distance between $data_i$ and every other cluster.

Example. Figure 4.2 plots the [mean, standard deviation, median] vectors of the monthly travel time for a segment (WE23-MCC5_5) on a bus trip of route 5 (Figure 4.6). It clearly shows the variation between monthly data and these 5 months can be clustered into two groups: [May, June, July] and [August]. This variation is used to produce different schedule for these clusters.

4.4.2 Estimating On-time Performance of Transit Schedules

A testbed is an evaluation platform for scientific theories, computational tools and technologies. When a new timetable is generated, an evaluation testbed is needed to simulate the bus trips using new schedule and estimate the on-time performance of new schedule. We assume the distribution of historical travel times represent the traffic patterns and propose a simple evaluation mechanism for new timetables that only requires historical data and a timetable as inputs.

Historical Dwell Time Estimation. Travel demand at bus stops is important statistics for

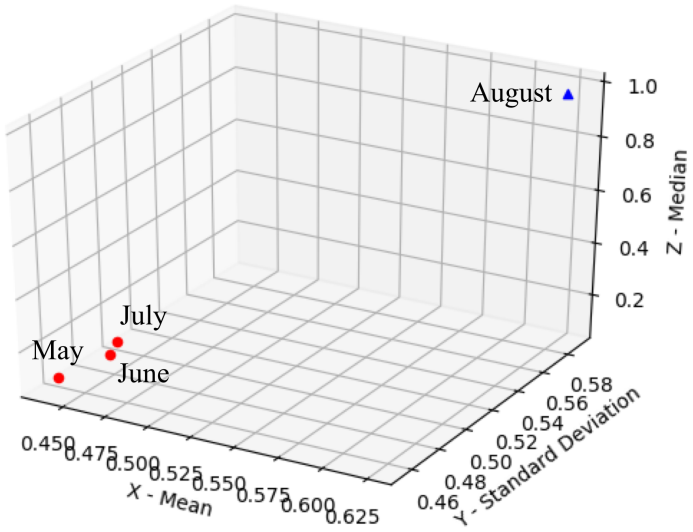


Figure 4.2: The feature vectors [mean, standard deviation, median] of the travel time in 4 months of 2016 for a segment (WE23-MCC5_5) on a bus trip of route 5.

setting up proper schedule times. However, for bus systems without automatic passenger counters (APCs), historical travel demand (represented by number of commuters boarding) is not available in original datasets. To get demand patterns, we utilize historical arrival and departure times to estimate the dwell time caused by passengers. Particularly, we consider the following two scenarios in historical records: (1) if a bus arrives earlier than scheduled time, the waiting time between the scheduled time and actual departure time is used, (2) if a bus arrives later than scheduled time, the waiting time between the actual arrival time and departure time is used. As shown in Table 4.1, for the timepoint SY19 on trip 1 with scheduled time of 11:02 AM:

- If a historical bus arrived earlier at 10:58 AM and departed at 11:04 AM, since the bus would always wait there as least for 4 minutes (between the actual arrival time 10:58 AM and the schedule time 11:02 AM) regardless of there were passengers or not, we assume the dwell time caused by passengers is the extra time after the scheduled time (11:04 AM - 11:02 AM = 2 minutes).

- If a historical bus arrived later at 11:05 AM and departed at 11:06 AM, then the dwell time caused by passengers is the extra time after the actual arrival time (11:06 AM -

11:05 AM = 1 minutes).

Arrival Time Estimation. The arrival time of a bus at a stop is impacted by two factors: (1) travel times at segments before the stop, and (2) dwell times at the previous stops. We assume that a bus will wait until the scheduled time if it arrives earlier than the scheduled time, and the historical travel time between two timepoints will remain the same in the simulation. To estimate arrival time, historical dwell time caused by passengers, which represent the historical travel demand, is added to the simulated arrival time at a timepoint, if the sum time is still earlier than the new scheduled time, then the simulation waits for extra time until the new scheduled time. The simulated departure time $st_{h,s_{j+1}}^{depart}$ at a timepoint s_{j+1} can be calculated using the simulated departure time st_{h,s_j}^{depart} at previous timepoint s_j, the actual travel time $t_{s_{j+1}}^{arrive} - t_{s_j}^{depart}$ between s_j and s_{j+1}, the dwell time $t_{s_{j+1}}^{dwell}$. Thus the new schedule time $T_{h,ss_{j+1}}^{depart}$ at s_{j+1} is calculated using the following equation:

$$st_{h,s_{j+1}}^{depart} = \max(T_{h,ss_{j+1}}^{depart}, st_{h,s_j}^{depart} + (t_{s_{j+1}}^{arrive} - t_{s_j}^{depart}) + t_{s_{j+1}}^{dwell}) \tag{4.7}$$

Example. Supposing the standard $[t_{early}, t_{late}]$ for on-time arrivals is between 1 minute early and 5 minutes late. In the example shown in Table 4.1, considering the three arrivals on trip 1 after the first timepoint (MCC4_14), the original on-time performance is 0% using Equation 4.1 (i.e., non of the arrivals are between 1 minute early and 5 minutes late). Assuming in a new timetable, the schedule times at timepoints SY19, PRGD, and GRFSTATO are updated (shown in Table 4.3). Since in the historical trip, the bus departed immediately after arriving at timepoints SY19 and PRGD, the historical dwell time caused by passengers are minimal, and arrival delays are mostly generated by travel delays on segments between the timepoints. The estimated arrival and departure times using the new timetable and the historical data of trip 1 are shown in Table 4.3. The estimated new on-time performance is 100%.

Table 4.3: The historical and estimated new departure and arrival times for a bus trip on route 4 on Aug. 8, 2016.

		Timepoints			
		MCC4_14	SY19	PRGD	GRFSTATO
Historical	Old Schedule Time	10:50 AM	11:02 AM	11:09 AM	11:18 AM
	Actual Arrival Time	10:36 AM	11:10 AM	11:18 AM	11:27 AM
	Actual Departure Time	10:50 AM	11:10 AM	11:18 AM	11:30 AM
Simulated	New Schedule Time	10:50 AM	11:11 AM	11:19 AM	11:29 AM
	Estimated Arrival Time	10:36 AM	11:10 AM	11:19 AM	11:28 AM
	Estimated Departure Time	10:50 AM	11:11 AM	11:19 AM	11:30 AM

4.4.3 Timetable Optimization Using a Greedy Algorithm

We first come up with a greedy algorithm that is simple and fast. The basic idea of the greedy algorithm is that it divides an entire trip into sequential optimization stages by timepoint segments, and then select a local optimal travel time at each stage. The schedule time at the first timepoint of each trip will remain the same. The greedy algorithm adjusts the scheduled arrival time greedily and sequentially for the succeeding segments between timepoints. The local optimal travel time is selected to maximize the percentage of on-time arrival delay within range $[t_{early}, t_{late}]$ using historical delay dataset.

Initialization The initialization step prepares the data for following steps. The actual travel time data between any two consecutive timepoints is aggregated using the historical dataset.

Optimization In the optimization step, the scheduled arrival time from the second timepoint to the last timepoint in a trip is optimized sequentially. Our goal is to pick new schedule time for two consecutive timepoints that can maximize the bus arrivals with delay within desired range $[t_{early}, t_{late}]$. It's a greedy algorithm because when adjusting the schedule time for a timepoint, only the on-time performance of the preceding timepoints and the current timepoint is considered. Instead of assuming the data follows any specific distribution (e.g. Gaussian distribution), we decide to utilize the empirical cumulative distribution function (CDF) to evaluate the percentage of historical delay in desired range.

Data: $D \leftarrow$ Historical timepoint datasets
Input : (1) $[t_{early}, t_{late}] \leftarrow$ on-time range, (2) $h \leftarrow$ bus trip for optimization, (3) $upperLimit$ $\leftarrow$ upper limit of the number of clusters
Output: Optimized schedule b at timepoints for bus trip h

```
[s_1, ..., s_n] ← GetAllTimepoints(D, h);
GetHistoricalData(D, h);
monthClusters ← ClusterMonthData(upperLimit);
for monthCluster ∈ monthClusters do
    b ← [];
    for s_i ∈ [s_1, ..., s_n] do
        maxCDF ← 0;
        optimizedTime ← 0;
        for candidate schedule time set x do
            if maxCDF ≤ CalculateEmpiricalCDF(x, t_early, t_late) then
                maxCDF ← CalculateEmpiricalCDF(x, t_early, t_late);
                optimizedTime ← x
            end
        end
        b ← b + optimizedTime
    end
end
```

Algorithm 1: Greedy algorithm for bus on-time performance optimization

An empirical CDF is a non-parametric estimator of the CDF of a random variable. The empirical CDF of variable x is defined as:

$$\hat{F}_n(x) = \hat{P}_n(X \leq x) = n^{-1} \sum_{n=1}^{n} I(x_i \leq x) \tag{4.8}$$

where $I()$ is an indicator function:

$$I(x_i \leq x) = \begin{cases} 1, & \text{if } x_i \leq x \\ 0, & \text{otherwise} \end{cases} \tag{4.9}$$

Then the CDF of x in range $[x+t_{early}, x+t_{late}]$ can be calculated using the following equation:

$$\begin{aligned} &\hat{F}_n(x+t_{late}) - \hat{F}_n(x+t_{early}) \\ &= n^{-1} \sum_{n=1}^{n} I(x+t_{early} \leq x_i \leq x+t_{late}) \end{aligned} \tag{4.10}$$

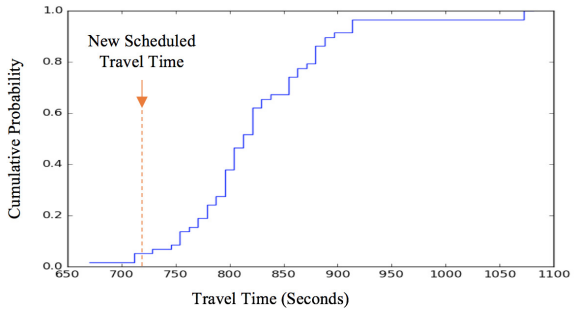


Figure 4.3: Empirical cumulative distribution function (CDF) of historical travel time between two timepoints (MCC5_5 and WE23) on route 3 in May, June, July 2016.

Figure 4.3 illustrates an example of the empirical cumulative distribution function (CDF) of historical travel time between two timepoints (MCC5_5 and WE23) on route 3 in May, June, July 2016. Choosing a new scheduled travel time of 720 seconds between these two timepoints could maximize the percentage of historical data points within range $[720 + t_{early}, 720 + t_{late}]$. Algorithm 1 shows the greedy algorithm's pseudo code.

4.4.4 Timetable Optimization Using Heuristic Algorithms

In classical optimization approaches driven by differential calculus, unique solutions are obtained. While these may work well for less complex problems or for problems with lower dimensionality, optimality for high dimensional and multimodal cost functions are not guaranteed. For problems where effective classical methods do not converge to the global optimum, heuristic optimization strategies provide an effective alternative approach. With the gradual increase in computational power, nature inspired heuristics and meta-heuristics are aimed at producing optimal or near optimal approximations of solutions in a stochastic manner within an ever decreasing bound of time. Heuristics are known to produce acceptable quality of approximate solutions fast when applied to NP-complete problems and are distinctively typified by the way the traversal of promising regions of the search hyperspace is accomplished.

The performance optimization for scheduling transit vehicles is a multidimensional

problem and as such the cost function is nonconvex in nature consisting of several troughs and ridges. Hence, to compute the optimally scheduled routing strategy with acceptable time constraints, an approach powered by high quality of solution estimation techniques such as evolutionary algorithms and metaheuristics can be considered. One such approach is particle swarm optimization (PSO) which is a nature inspired swarm intelligence algorithm described in subsection 5.4.2. PSO can be employed to optimize a large variety of stochastic cost functions including non-differentiable or discontinuous ones as it is less sensitive to the objective function at hand [121]. The implementation of PSO does not rely on the gradient of the underlying cost function and the particles are distributed over promising regions in the search space. The algorithm is easy to implement with a few parameters such as inertial weight and social and cognition acceleration coefficients, thereby making it highly customizable and popular.

4.4.4.1 Genetic Algorithm

Genetic algorithm is a heuristic optimization algorithm that derives from biology. The basic steps involved in genetic algorithms include initialization, selection, crossover, mutation, and termination. The timetable for each trip is decided by the scheduled departure time at the first stop as well as the scheduled travel time between any two subsequent timepoints along the trip. Since our goal is to update timetables to make the bus arrivals more on time, we assign the scheduled travel times between timepoints as chromosomes in populations, and use the on-time performance estimation mechanism proposed in Section 4.4.2 as cost functions. The chromosome of the individual solutions in the genetic algorithm is a vector of integers representing travel time between subsequent timepoints. In order to reduce the search space and match the real-world scenarios, the travel time in each individual is re-sampled to a multiple of 60 seconds and restricted to the unit of minutes.

Initialization For a genetic algorithm, the initialization step creates the initial state which involves population size and initial chromosome values. A population size to de-

termine how many chromosomes are there in one population, which affects the ultimate performance and computation efficiency [122]. Smaller population makes iterations faster but less various in chromosome crossover. Larger population will have the opposite effects. We chose 50 as the population size ps.

In order to initialize the first population, the actual travel time between timepoints is aggregated from the historical datasets. Then the travel time in each individual is randomly selected between the maximum and minimum of historical data. We observed that the seeding in the initial population with heuristic solutions such as original scheduled travel time or optimized results from the greedy algorithm (presented in Section 4.4.3) would only affect the fitness of initial population and had little effects on the final optimality, so the initial population is generated at random.

Selection At the beginning of each iteration step, a portion of the existing population needs to be selected as parents to breed a new generation. A fitness function is required to determine how fit a solution is and a selection strategy is needed to select the solutions with better fitness. In our case, the objective function, defined in equation 4.1 is used as the fitness function. Since the fitness function contains an indicator function $I(h_i, s_j)$, and the value of the indicator function is related to the arrival delay at timepoint s_j, a simulation mechanism is needed to evaluate the on-time performance of the new schedule using historical data. To simulate the bus arrival and departure activities at timepoints, historical travel times between two consecutive timepoints and historical dwell time at timepoints are used.

Our genetic algorithm uses tournament selection [123] to randomly select new solutions. Each time, we select 2 individuals at random from the current population and pick the one with better fitness to become a parent. This process is repeated until the number of parents reaches the population size. The specifics are:

Crossover Using crossover, sub-solutions on different chromosomes are combined at random. A uniform crossover [124] technique is used for the crossover operation in gene

level. Unlike one point or multi-point crossover, uniform crossover treats each gene separately. Two parents are randomly selected and their genes are exchanged (the scheduled travel time between two successive timepoints

with another on the same places of the solution vector. The individual travel times between two parents are swapped with a fixed probability of 60%. An example illustrating the crossover is shown in Figure 4.4.

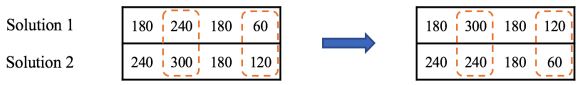


Figure 4.4: Crossover: two genes are swapped between two individuals.

Mutation Mutation generates genetic diversity from one generation of a population of chromosomes to the next. The mutation works in two steps: (1) a schedule travel time between two timepoints in a solution is selected at random, (2) randomly add or minus 60 seconds to the time with the requirement that the new time should be within the historical travel time distribution range. The mutation probability is set as 0.005 and the population size *ps* is 50. Suppose each individual has 5 genes, 250 genes in total should lead to the result that one gene will mutate in each iteration.

Termination The termination condition of a genetic algorithm is critical to determine whether the algorithm should end or not. According to the study of stopping criteria for genetic algorithm [125], the following three types of conditions are mostly employed: (1) an upper limit of generation number is reached, (2) an upper limit of fitness function value is reached, (3) the change or achieving significant changes in the next generation is excessively low. Since the best on-time performance that the GA can achieve for each bus trip varies, setting the upper limit of the fitness function value does not work here. We choose 1,000 as the upper generation number limit. At the same time, if the difference between the average fitness value of the solutions in the current generation and previous generation is below a pre-defined threshold 0.00001, then the algorithm will also terminate.

Data: $D \leftarrow$ Historical timepoint datasets
Input : (1) $[t_{early}, t_{late}] \leftarrow$ on-time range , (2) $maxGen \leftarrow$ maximum number of generations $maxGen$, (3) $pSize \leftarrow$ number of solutions in the population $pSize$, (4) $tt \leftarrow$ termination threshold, (5) $cP \leftarrow$ crossover probability, (6) $mP \leftarrow$ mutation probability, (7) $h \leftarrow$ bus trip for optimization, (8) $upperLimit \leftarrow$ upper limit of the number of clusters
Output: Optimized schedule b at timepoints for bus trip h

```
GetAllTimepoints(D, h);
GetHistoricalData(D, h);
monthClusters ← ClusterMonthData(upperLimit);
for monthCluster ∈ monthClusters do
    P ← [];
    for population size pSize do
        P ← P ∪ InitialIndividual();
    end
    i_population ← 0;
    while maxGen is reached or AverageFitness(P_{i_population}) - AverageFitness(P_{i_population−1}) ≤ tt
     do
        P ← TournamentSelect(P);
        P ← UniformCrossover(P, cP);
        P ← Mutation(P, mP);
    end
end
```

Algorithm 2: Genetic algorithm for bus on-time performance optimization

The pseudo code of the genetic algorithm is given in Algorithm 2. We utilize historical timepoint datasets to conduct the genetic algorithm for this optimization problem. The input includes on-time range, number of generation limit, number of solutions in the population, termination threshold, crossover and mutation probability, bus trip and upper limit of number of month clusters.

4.4.4.2 Particle Swarm Optimization

Eberhert and Kennedy [118] proposed particle swarm optimization (PSO) as a stochastic population based optimization algorithm which can work with non-differentiable cost function without explicitly assuming its underlying gradient disparate from gradient descent techniques, PSO has been shown to satisfactorily provide solutions to a wide array of complex real-life engineering problems, usually out of scope of deterministic algorithms

[126, 127, 128]. PSO exploits the collective intelligence arising out of grouping behavior of flocks of birds or schools of fish. This manifestation of grouping is termed as 'emergence', a phenomenon in which a cohort of individuals from a social network is aimed to accomplish a task beyond their individual capability. Likewise, each particle in the swarm, represents a potential solution to the multi-dimensional problem to be optimized.

Initialization Each particle has certain position which can be thought of as a collection of co-ordinates representing the particle's existence in a specific region in the multidimensional hyperspace. As a particle is a potential solution to the problem, the particle's position vector has the same dimensionality as the problem. The velocity associated with each particle is the measure of the step size and the direction it should move in the next iteration.

Each particle in the swarm maintains an n-dimensional vector of travel times. At first, the position for each particle in the population is initialized with the set of travel time between the timepoints randomly selected between the minimum and maximum of the aggregated actual historical data. With swarm size as p, every particle i ($1<i<p$) maintains a position vector x_i=(x_{i1},x_{i2},x_{i3},...,x_{in}) and a velocity vector v_i=(v_{i1},v_{i2},v_{i3},...,v_{in}) and a set of personal bests p_i=(p_{i1},p_{i2},p_{i3},...,p_{in}).

Optimization At each iteration, the position of a particle is updated, and compared with the personal best (*pbest*) obtained so far. If the cost due to the position obained at current iteration is more (as it is a cost maximization problem) than the *pbest* obtained upto the previous iteration, then the current position becomes the personal best or *pbest*, otherwise *pbest* remains unchanged. Thus the best position of a particle obtained so far is stored as *pbest*. The global best or *gbest* is updated when the population's overall current best, i.e., the best of the *pbsest*s is better than that found in the previous iteration.

After initializing positions and velocities, each particle updates its velocity based on previous velocity component weighted by an inertial factor, along with a component proportional to the difference between its current position and *pbest* weighted by a cognition acceleration coefficient, and another component proportional to the difference between its

current position and (*gbest*), weighted by a social acceleration coefficient. This is socio-cognitive model of PSO and facilitates information exchange between members of the swarm. Since all members are free to interact with each other, the flow of information is unrestricted and the PSO algorithm is said to have a 'fully-connected' topology. While updating the velocity, a particle's reliance on its own personal best is dictated by its cognitive ability, and the reliance on the entire swarm's best solution is dictated by its social interactive nature. Hence those factors in the velocity component are weighted by the cognition acceleration coefficient *c1* and social acceleration coefficient *c2*. The new positions of the particles are updated as the vector sum of the previous positions and the current velocities. Thus the positions of the particles, are updated aiming towards intelligent exploration of the search space, and subsequent exploitation of the promising regions in order to find the optimal solution based on cost optimization of the stated problem.

After each iteration is completed, the velocity and position of a particle are updated as follows:

$$v_{i,j}(t+1) = w.v_{i,j}(t) + c_1.r_1(t).(p_{i,j}(t) - x_{i,j}(t)) + c_2.r_2(t).(p_{g,j}(t) - x_{i,j}(t)) \tag{4.11}$$

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1) \tag{4.12}$$

$v_{i,j}$ and $x_{i,j}$ represent the velocity and position of the *i-th* particle in the *j-th* dimension. Cognition and social acceleration coefficients are indicated by c_1 and c_2, whereas r_1 and r_2 are random numbers uniformly distributed between 0 to 1. $p_{i,j}$ represents a particle$'$s personal best and $p_{g,j}$ represents the global best of the population. w acts as an inertial weight factor controlling the exploration and exploitation of new positions in the search space and t denotes the number of iterations.

The problem is formulated as fitness maximization problem in order to bring out optimal travel times to improve on-time performance. Hence the personal best of a particle is updated as follows at the end of each iteration.

Data: $D \leftarrow$ Historical timepoint datasets
Input : (1) $[t_{early}, t_{late}] \leftarrow$ on-time range , (2) $maxIter \leftarrow$ maximum number of iterations $maxIter$, (3) $npop \leftarrow$ number of particles in the population size $npop$, (4) $w \leftarrow$ inertia weight, (5) $c1 \leftarrow$ cognition acceleration coefficient, (6) $c2 \leftarrow$ social acceleration coefficient, (7) $h \leftarrow$ bus trip for optimization, (8) $upperLimit \leftarrow$ upper limit of the number of clusters
Output: Optimized schedule b at timepoints for bus trip h
GetAllTimepoints(D, h);
GetHistoricalData(D, h);
$monthClusters \leftarrow$ ClusterMonthData($upperLimit$);
for *monthCluster* $\in$ *monthClusters* **do**
  $P \leftarrow []$;
  for *population size npop* **do**
    Initialize each particle with random position and velocity
    $P \leftarrow P \cup InitialIndividual()$;
  end
  while *maxIter is reached* **do**
    Evaluate the cost function (J) for each particle's position (x)
    if *J(x) ¿ J(pbest)*, then *pbest = x*
    gbest $\leftarrow$ Update if the population's overall current best is better than that in previous iteration
    Update the velocity of each particle according to equation *(10)*
    Update the position of each particle according to equation *(11)*
  end
  Give *gbest* as the optimal schedule b at timepoints for bus trip h
end

Algorithm 3: particle swarm optimization for bus on-time performance optimization

$$p_{i,j}(t+1) = \begin{cases} p_{i,j}(t), & \text{if } cost(x_{i,j}(t+1)) < cost(p_{i,j}(t)) \\ x_{i,j}(t+1), & \text{if } cost(x_{i,j}(t+1)) \geq cost(p_{i,j}(t)) \end{cases} \tag{4.13}$$

As discussed earlier, our goal is to provide new schedule time for two consecutive timepoints that can maximize the bus arrivals with delay within desired range $[t_{early}, t_{late}]$. The empirical cumulative distribution function (CDF) for evaluating the percentage of historical delay in desired range is decribed in equation [7-9] which is considered in *cost* calculation. All other calculations including the historical dwell time cused by the passengers, the simulated departure time, the new scheduled time are done in the same way as described in the Genetic Algorithm and the calculations are described in the section 5.2.

Termination The termination condition set for PSO is the predefined maximum number of iterations. Since the optimized on-time performance is different for each trip, the termination condition is not set as any predefined upper limit of the fitness value. With other hyperparameters fixed PSO can produce the optimal solution approximately in 30 iterations for this problem.

The pseudo code for PSO is discussed in Algorithm 3. Historical timepoint datasets are used to conduct the particle swarm optimization algorithm for this problem. The input includes on-time range, maximum number of iterations, number of particles in the population size, inertia factor, cognition and social acceleration coefficient, bus trip and upper limit of number of month clusters.

4.4.4.3 Sensitivity Analysis on Hyper-parameters

In the context of heuristic searching algorithms, exploration ability and exploitation ability are two contradictory criteria that must be taken into account. Exploration searches undiscovered regions as much as possible, while exploitation tries to concentrate on regions neighbouring possible optima. The exploration ability and exploitation ability of genetic algorithm are controlled by the hyper-parameters such as population size, crossover rate, and mutation rate. For example, in genetic algorithms, the exploitation is conducted by crossover operations and the exploration is rendered by mutation. Changing the rates of the crossover operator and the mutation operator will result in different optimization performance in accuracy and time complexity.

In case of particle swarm optimization, the velocity vector is guided by three distinct factors. It has a component of its own past velocity, weighted by the inertia factor w, another component towards its own best so far, and another towards the global best so far. The inertia weight w is also responsible to control the exploration exploitation ratio of the algorithm to reach optima. Cognition and social acceleration coefficients $c1$ and $c2$ maintain a balance in determining the direction of a particle's next move providing a

measure of the extent, a particle should follow its own historical best and the global best in order to achieve optima.

The best values of hyper-parameters depend on the nature and implementation intricacies of the heuristic algorithms and are generally problem specific. In Section 4.5.3 we discuss results of the sensitivity analysis and present suggestions.

4.5 Evaluation and Sensitivity Analysis Results

This section describes the simulation results for evaluating the proposed on-time performance optimization mechanisms. We first setup experiments to check whether the clustering analysis could capture the monthly and seasonal delay variations, and then compare the optimization performance of the proposed greedy, genetic and PSO algorithms. Sensitivity analyses are performed in the end to help tune the model hyper-parameters. Our simulation experiments were conducted using real-world bus data collected from the Nashville transit system. Data leakage is a potential issue, especially for dealing with time series data. To prevent data leakage issues when dividing original datasets into training and testing datasets, we add constraints that the timestamps of the data samples in testing datasets are always later than the data in training datasets.

4.5.1 Experiment 1: Evaluating the Clustering Analysis

Hypothesis. The clustering analysis proposed in Section 4.4.1 could identify months with similar delay patterns and group them together. Compared with generating a single timetable for all months, creating separate timetables for different month groups could help improve the on-time performance further.

Simulation Setup. To evaluate the hypothesis about the clustering analysis, we designed two simulations to compare the optimized on-time performance with and without a clustering analysis step: (1) months are not clustered and a single timetable is generated for all months, (2) month clustering is conducted at first and the optimization algorithms

Table 4.4: The original and optimized on-time performance on average across all bus routes.

	Origin	GA without Clustering	GA with Clustering	PSO with Clustering
On-time Perf.	57.79%	66.24%	68.34%	68.93%

is applied on different month groups to generate separate timetables. For example, if the historical delay patterns of a bus trip are clustered into 2 groups (e.g., group 1 contains April, and group 2 contains May, June, July and August), then in the first simulation both groups will get the same timetable, while in the second simulation group 1 and group 2 will get two different bus schedules. The simulations use real-world data collected from 4 months (May, June, July and August) in 2016.

Simulation Results. Table 4.4 shows the original and optimized on-time performance on average across all bus routes. Using the genetic algorithm without clustering step improved the original performance from 57.79% to 66.24%. Added the clustering step which groups months with similar patterns improved it further to 68.34%.

4.5.2 Experiment 2: Comparing Optimization Performance of Greedy, Genetic, and PSO Algorithms

Hypothesis. Both proposed greedy and genetic algorithms could improve the on-time performance. However, since the genetic algorithm could optimize the schedule for all segments for each route all together, the genetic algorithm will outperform the greedy algorithm.

Simulation Setup. Two simulations are designed to evaluate the optimization ability of the proposed greedy and genetic algorithms. For each trip, months are first clustered according to their delay patterns, the greedy algorithm and the genetic algorithm are then applied in the two simulations.

Simulation Results. The original on-time performance, optimized on-time performance using greedy algorithm and genetic algorithm are illustrated in Figure 4.5. The

hypothesis is validated that while both algorithms can improve the on-time performance, the genetic algorithm outperforms the greedy algorithm because it optimizes the schedule for all timepoint segments on each trip all together. The original on-time performance of all bus routes in origin is 57.79%. The greedy algorithm improved it to 61.42% and the genetic algorithm improved it further to 68.34%. The PSO algorithm has a slightly better optimized on-time performance of 68.93%.

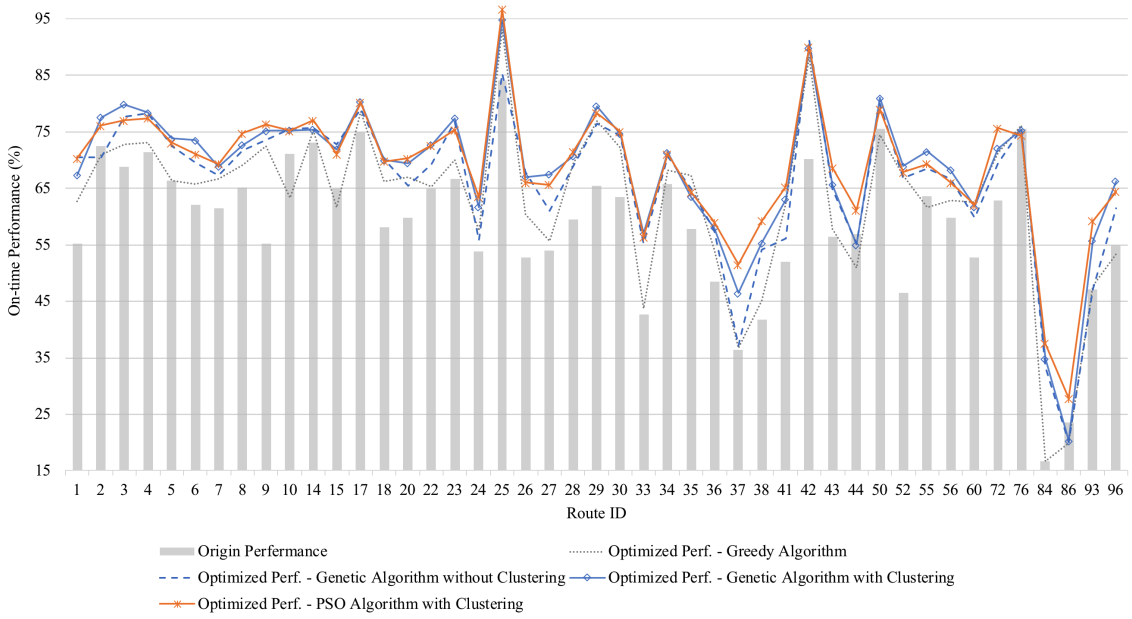


Figure 4.5: The original on-time performance and the optimized on-time performance using greedy algorithm, genetic algorithm with/without clustering analysis and PSO algorithm.

4.5.3 Experiment 3: Sensitivity Analysis on the Hyper-parameters of the Genetic algorithm

Hypothesis. Changing the population size and the rates of the crossover operator and the mutation operator will result in different optimization performance in terms of accuracy and time complexity. Even though the optimal hyper-parameter settings are problem-specific, there are some rules and patterns for selecting the best values.

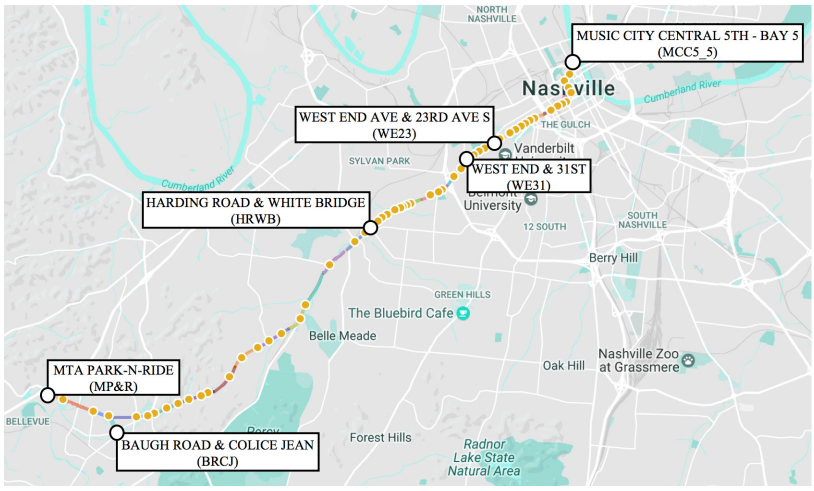


Figure 4.6: Timepoints on bus route 5 in Nashville

Simulation Setup. In this experiment, we designed three simulations that choose different hyper-parameters: (1) population sizes that range from 10 to 110, (2) crossover rates that range from 0.1 to 1.0, (3) mutation rates that range from 0.1 to 1.0. Real-world data collected from Route 5, which is one of major bus routes that connects downtown Nashville and the southwest communities in Nashville. The route contains 6 timepoint stops and 5 segments between the 6 timepoint stops. The bus trips with direction from Downtown are selected. The goal is to maximize the on-time performance for these trips by optimizing the schedule time at the 6 timepoint stops.

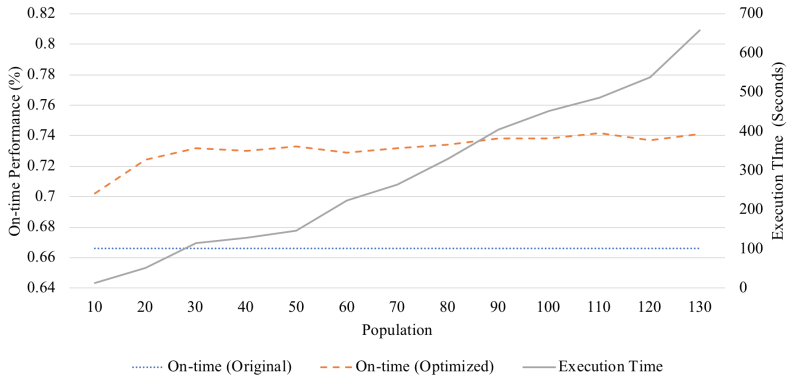


Figure 4.7: The chart shows the on-time performance and overall execution time for different population sizes.

Simulation Results. Figure 4.7 shows the simulation results of choosing different population sizes. Increasing the population size from 10 to 90 results a better on-time performance, however, increasing the size ever further doesn't help making the on-time performance any better. On the other hand, the total time increases linearly as the population size grows. So a population size around 90 is the optimal size to use.

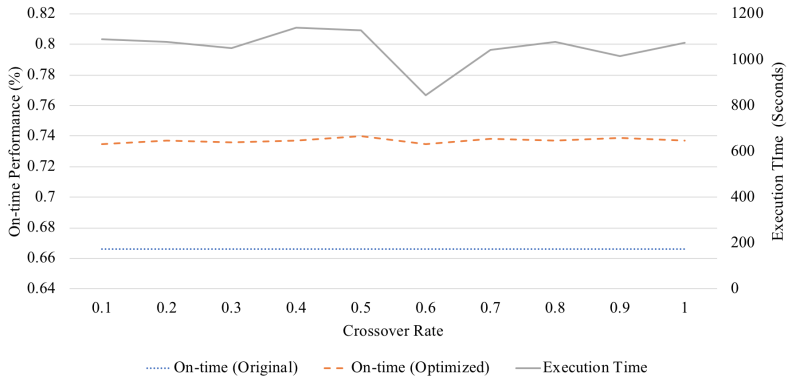


Figure 4.8: The chart shows the on-time performance and overall execution time for different crossover rates, which controls the exploitation ability of the GA.

Figure 4.8 illustrates results of using different crossover rates. The optimized on-time performance remains almost the same for the crossover range, but there is a significant difference in terms of the total execution time. The crossover rate impacts the exploitation ability. A proper crossover rate in the middle of the range can faster the process to concentrate on an optimal point.

Figure 4.9 show the simulation results when using different mutation rates. The total execution time is small when the mutation rate is either very small or very large. Mutation rates controls the exploration ability. During the optimization, a small mutation rate will make sure the best individuals in a population do not vary too much in the next iteration and thus is faster to get stable around the optimal points. So we suggest setting a very small mutation rates when running the proposed algorithm.

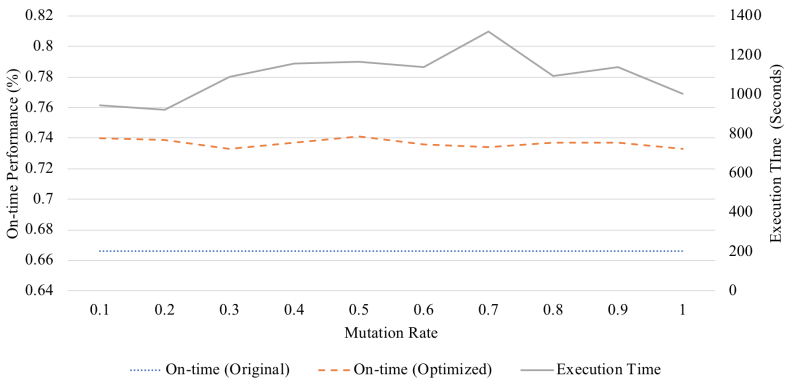


Figure 4.9: The chart shows the on-time performance and overall execution time for different mutation rates, which controls the exploration ability of the GA.

4.5.4 Experiment 4: Sensitivity Analysis on the Hyper-parameters of Particle Swarm Optimization

Hypothesis. Variation in the hyper-parameters like population size, inertia weight w, cognition acceleration coefficient $c1$, social acceleration coefficient $c2$, and their ratios to each other in the velocity update equation will lead to different solution quality with respect to accuracy and execution time. Although the selection of optimal values of the hyperparameters are problem specific, some patterns can be observed to present an overview of the feasible range of operation for acceptable solutions.

Simulation Setup. In this experiment, we designed four simulation setups that choose different hyper-parameters: (1) The inertial weight factor,w that range from 1 to 8, (2) Social acceleration coefficient $c1$ that range from 1 to 8, (3) Cognition acceleration coefficient $c2$ that range from 1 to 8, and (4) Number of particles that range from 2 to 36. Real-world data regarding bus timings is collected from Route 8, which is one of the major bus routes that connects Music City Central Nashville and the Lipscomb University in Nashville. The route contains 5 timepoint stops and 4 segments between the 5 timepoint stops. The goal is to maximize the on-time performance for these trips by optimizing the schedule time at the 5 timepoint stops.

Simulation Results. Figure 4.10 shows the simulation result while optimizing for the inertial weight *w* by varying it. It is observed that the optimized on-time performance is at its peak when *w* is nearly equal to 5 with less execution time. Performance deteriorates along with an increase in execution time as the selection is moved away from 5. So, an optimal value to choose for w, will be somewhere around 5.

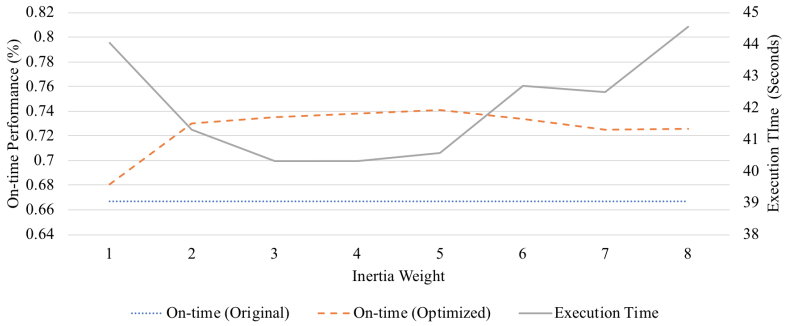


Figure 4.10: The chart shows the on-time performance and overall execution time for different inertia weights, exploring new regions of search space in PSO.

Figure 4.11 shows the simulation result for optimizing the cognition acceleration coefficient *c1* by varying it. The particle has a velocity component towards its own best position weighted by *c1*, hence the term 'cognitive'. It is observed that the optimized on-time performance is improved when *c1* increases from 3 to 5 with less execution time. After that the performance deteriorates along with increase in execution time. So, an optimal value to choose for *c1*, will be within the range specified.

Figure 4.12 shows the simulation result for optimizing for the social acceleration coefficient *c2* by varying it. The particle has a velocity component towards the global best position weighted by *c2*, hence the term social. It is observed that the optimized on-time performance is improved when *c2* is equal to 5 with less execution time. Also, *c2* being 4 produces good results, but there is an increase in execution time at that value. But the overall effect of parameter *c2* affects the on-time performance only within a range of two percent. Sometimes, PSO is able to produce optimal or near optimal performance, when

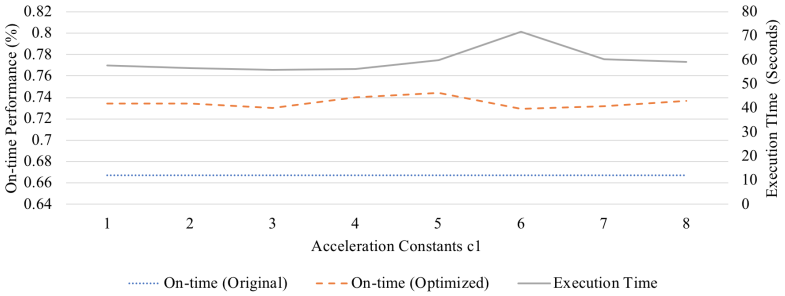


Figure 4.11: The chart shows the on-time performance and overall execution time for different cognition acceleration coefficients *c1*, in PSO.

all other hyperparameters are fixed, and thus is not sensitive to a particular hyperparameter, which is the case considered here. So an optimal value to choose for *c2*, may be close to 5, maintaining approximately a ratio near to 1:1:1 among *w*, *c1* and *c2*.

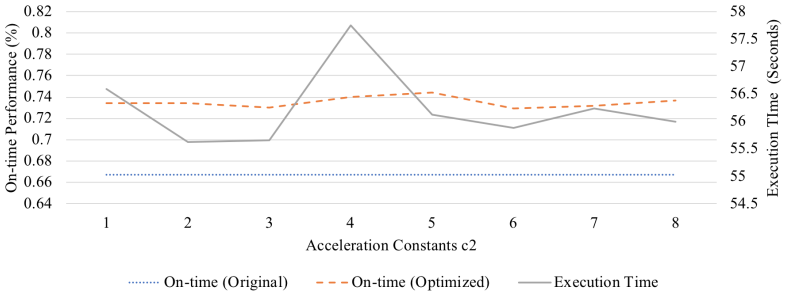


Figure 4.12: The chart shows the on-time performance and overall execution time for different social acceleration coefficients *c2*, in PSO.

Figure 4.13 shows the simulation result for optimizing the number of particles by varying the population size. It is observed that the optimized on-time performance is maximized when the number of particles reaches 30. The execution time increases with the number of particles, so it is better to choose such number of particles that produces the best pair in the accuracy-execution time tradeoff. So, the population size can be chosen as 30 in this case as it yields equally efficient results with a relatively small execution time.

Although a good insight about choices of hyperparameters can be obtained from this

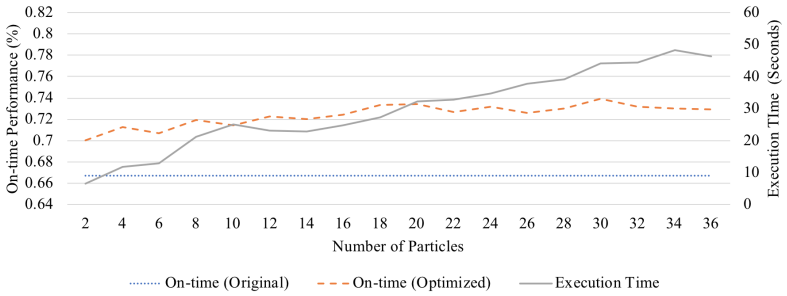


Figure 4.13: The chart shows the on-time performance and overall execution time for various population size, in PSO.

sensitivity analysis, variations of the hyperparameters may produce better results in specific routes.

4.6 Conclusion

In this chapter, we present research on a bus on-time performance optimization toolchain that significantly extends our prior work [59] by proposing an optimization toolchain and presenting sensitivity analysis on choosing the optimal hyper-parameters. Particularly, we describe an unsupervised analysis mechanism to find out how months with similar delay patterns can be clustered to generate new timetables. A genetic algorithm as well as greedy algorithm and a classical particle swarm optimization algorithm are proposed to optimize the schedule time to maximize the probability of bus trips that reach the desired on-time range. Simulations of optimization performance as well as sensitivity analysis on the hyper-parameters of the GA and PSO algorithms are conducted. The results indicate different strategies for choosing between the genetic, PSO and heuristic algorithms, and selecting optimal hyper-parameters.

Publication. This work has been published in the following place:

- Fangzhou Sun, Chinmaya Samal, Jules White and Abhishek Dubey, Unsupervised Mechanisms for Optimizing On-Time Performance of Fixed Schedule Transit Vehi-

cles, SMARTCOMP2017: Smart Computing Technologies and Applications, May 29-31, 2017, Hong Kong, China

CHAPTER 5

DEEP NEURAL NETWORKS FOR CONTEXT-AWARE ANOMALY DETECTION IN TRANSIT NETWORKS

Non-recurring traffic congestion is caused by temporary disruptions, such as accidents, sports games, adverse weather, etc. We use data related to real-time traffic speed, jam factors (a traffic congestion indicator), and events collected over a year from Nashville, TN to train a multi-layered deep neural network. The traffic dataset contains over 900 million data records. The network is thereafter used to classify the real- time data and identify anomalous operations. Compared with traditional approaches of using statistical or machine learning techniques, our model reaches an accuracy of 98.73% when identifying traffic congestion caused by football games. Our approach first encodes the traffic across a region as scaled images. After that the image data from different timestamps is fused with event- and time-related data. Then a crossover operator is used as a data augmentation method to generate training datasets with more balanced classes. Finally, we use the receiver operating characteristic (ROC) analysis to tune the sensitivity of the classifier. We present the analysis of the training time and the inference time separately. The content of this chapter has appeared in a conference paper [55].

5.1 Problem Overview

Emerging Trends. Traffic congestion in urban areas has become a significant issue in recent years. Because of traffic congestion, people in the United States traveled an extra 6.9 billion hours and purchased an extra 3.1 billion gallons of fuel in 2014. The extra time and fuel cost were valued up to 160 billion dollars [3]. Congestion that is caused by accidents, roadwork, special events, or adverse weather is called non-recurring congestion (NRC)

[28]. Compared with the recurring congestion that happens repeatedly at particular times in the day, weekday and peak hours, NRC makes people unprepared and has a significant impact on urban mobility. For example, in the US, NRC accounts for two-thirds of the overall traffic delay in urban areas with a population of over one million [129].

Driven by the concepts of the Internet of Things (IoT) and smart cities, various traffic sensors have been deployed in urban environments on a large scale. A number of techniques have been developed for knowledge discovery and data mining by integrating and utilizing the sensor data. Traffic data is widely available by using static sensors (e.g., loop detectors, radars, cameras, etc.) as well as mobile sensors (e.g., in-vehicle GPS and other crowdsensing techniques that use mobile phones). The fast development of sensor techniques enables the possibility of in-depth analysis of congestion and causes.

The problem of finding anomalous traffic patterns is called traffic anomaly detection. Understanding and analyzing traffic anomalies, especially congestion patterns, is critical to helping city planners make better decisions to optimize urban transportation systems and reduce congestion conditions. To identify faulty sensors, many data-driven and model-driven methods have been proposed to incorporate historical and real-time data [130, 22, 131, 23]. Some researchers [132, 133, 134, 135] have worked on detecting traffic events such as car accidents and congestion using videos, traffic, and vehicular ad hoc data. There are also researchers who have explored the root causes of anomalous traffic [8, 26, 9, 27, 30, 10].

Most existing work still mainly focuses on a road section or a small network region to identify traffic congestion, but few studies explore non-recurring congestion and its causes for a large urban area. Recently, deep learning techniques have gained great success in many research fields (including image processing, speech recognition, bioinformatics, etc.), and provide a great opportunity to potentially solve the NRC identification and classification problem. There are still many open problems: (1) using feature vectors to represent traffic conditions loses the spatial information of the road segments, (2) using small

and unbalanced dataset (traffic data with event labels is limited) to train neural networks downgrades the performance, a proper data augmentation mechanism is needed to balance the training data with different class labels, (3) building deep neural networks to model the traffic conditions of both recurring and non-recurring congestion.

Contributions. In this chapter, we propose DxNAT, a deep neural network model to identify non-recurring traffic congestion and explain its causes. To the best of our knowledge, our work is one of the first efforts to utilize deep learning techniques to study traffic congestion patterns and explain non-recurring congestion using events. The main contributions of our research are summarized as follows:

- We present an algorithm to efficiently convert traffic data in Traffic Message Channel (TMC) format to images
- We introduce a crossover operator as a data augmentation method for training class balancing.
- A convolutional neural network (CNN) is proposed to identify non-recurring traffic anomalies that are caused by events.
- We create three scenarios to evaluate the performance of the proposed model by using real-world data of three events types (football games, hockey games, and traffic incidents).

5.2 Related Work and Challenges

This section presents an overview of the related work on traffic anomaly detection, which includes studies about faulty traffic sensor detection, traffic event detection, and congestion cause indication. Three key research challenges and our contributions for detecting NRC are discussed in the end.

5.2.1 Related Work about Traffic Anomaly Detection

Faulty Traffic Sensor Detection. Robinson et al. [130] proposed an approach that used data from inductive loop detectors to estimate travel time on road segments. His approach included a data cleaning method to clean the collected traffic data. Lu et al. [22] reviewed previous work on faulty inductive loops data analysis. Widhalm et al. [136] presented a traffic anomaly detection method that used Floating-Car Data (FCD) as an independent information source. They developed a non-linear regression model to fit the traffic sensor data and FCD data. Zygouras et al. [131] proposed a method comparing correlations among nearby sensors to identify faulty sensor readings. Their system was based on MapReduce paradigm to work for crowdsourcing data. Ghafouri et al. [23] presented a faulty traffic sensor detection model based on Gaussian Processes. Particularly, they provided an effective approach for computing the parameters of detectors to minimize the loss due to false-positive and false-negative errors.

Event Detection Using Traffic Data. Monitoring traffic flow at intersections is important in the traffic event detection research. Kamijo et al. [132] developed an algorithm based on spatiotemporal Markov random field (MRF) for processing traffic images and tracking vehicles at intersections. Using the timeseries observed behaviors of vehicles, a hidden Markov model for accident detection is then proposed. Veeraraghavan et al. [133] presented a multiple cue-based approach combined with a switching Kalman filter for detecting vehicle events such as turning, stopping and slow moving. Terroso-Senz et al. [137] presented an event-driven architecture (EDA) that used vehicular ad hoc network and external data sources like weather conditions to detect traffic congestions. Yang et al. [134] proposed a coupled Bayesian RPCA (BRPCA) model for detecting traffic events that used multiple traffic data streams. Kong et al. [135] proposed LoTAD to explore anomalous regions with long-term poor traffic situations. To model the traffic condition, crowd-sourced bus data is grouped into spatiotemporal segments. The segments with high anomaly indexes were combined to get anomalous regions. Wang et al. [138] proposed a two-stage

solution to detect road traffic anomalies: (1) a Collaborative Path Inference (CPI) model that performs path inference incorporating static and dynamic features into a Conditional Random Field (CRF); (2) a road Anomaly Test (RAT) model calculates the anomalous degree for each road segment.

Congestion Cause Indication. Liu et al. [8] studied both known (planned) and unknown (unplanned) events behaving differently from daily network traffics as anomalies, and proposed algorithms that construct outlier causality trees based on temporal and spatial properties of detected outliers. Xu et al. [26] introduced an approach to identify urban congestion patterns based on the data cube. They proposed a multi-dimensional data analysis method for data cube. Chow et al. [9] presented an automatic number plate recognition technology to analyze urban traffic congestions and introduced a linear regression model to indicate the causes of the congestions. Kwoczek et al. [30] proposed an Artificial Neural Network (ANN) based classifier to detect the road segments affected by planned events. Mallah et al. [10] evaluated the performance of machine learning techniques for classifying congestions into different causes.

5.2.2 Research Challenge 1: Representing Heterogeneous Traffic Data and Event Labels Using Multi-Dimensional Images

A feature vector is an n-dimensional vector and is the most popular representation of data objects. Besides numerical values, feature vectors can also represent texts and images. However, feature vectors may not be the best solutions for representing traffic and corresponding event labels.

Traffic conditions are highly affected by different influencing factors [33], such as incidents, sports games, road work, weather, etc. The events and their physical locations are used as the labels. But since feature vectors have fixed length, it is not practical to manually encode the labels to a specific fixed length feature vector. More importantly, in pattern recognition and machine learning, features matter the most. When converting an image to

a feature vector, you can directly convert the two-dimensional pixels to a one-dimensional vector, or you can first take the histogram of the image and then construct a feature vector that has several comparison metrics, such as mean, standard deviation, etc. Both methods will lose some relative spatial information in the original images.

In contrast to feature vectors, images can preserve the original spatial relations by locating points on different pixels and can integrate multiple data sources by simply adding layers. Kwoczek et al. [30] showed a factor representation that integrates multiple features like event and weather into different layers in a data cube. However, though they mentioned the idea as a possible future work, they did not present any concrete solution to it. Ma et al. [139] proposed a CNN-based approach for traffic prediction. They represented the traffic speed and time using a time-space matrix. The problem with the time-space matrix is that the spatial information between segments is lost, which is important in detecting traffic patterns because nearby roads usually show similar or related patterns. Additionally, their model simply considered traffic data, but there are many other factors affecting the future traffic conditions. Thus representing heterogeneous traffic and corresponding event labels using images remains a research gap.

One of the key differences between our proposed approach and the existing ones is that we are trying to visualize the wide area sensor data distribution as Traffic Condition Images (TCIs), so that we can use CNN and other deep learning techniques for analytics.

5.2.3 Research Challenge 2: Training Deep Learning Models Using Limited Data Instances

The performance of deep learning techniques highly relies on the quality of training data instances. However, the collected urban data may not provide enough data for training because of the data sampling rates. For example, our proposed model first converts traffic data to Traffic Condition Images (TCIs) and then trains different models using these images. But the traffic data we obtain from HERE [34] is requested every minute. So for a day

that consists of 1440 minutes, we will only have 1440 traffic images, which are too few for effective training purposes. The availability issue of data instances becomes worse considering there is also limited label data. It remains a research challenge of getting more training data using the existing data.

Traditional ways of solving this problem are: (1) waiting and collecting until enough training data is collected, (2) manually labeling the data, (3) adding data sources, e.g., collecting more data from social media. Our solution uses the idea of crossover from the genetic algorithm. We assume that traffic conditions within a short time range are associated with the same events. So we can apply a crossover operator on the TCIs to generate more TCIs with the same event label.

5.2.4 Research Challenge 3: Modeling Traffic Patterns of Non-Recurring Events

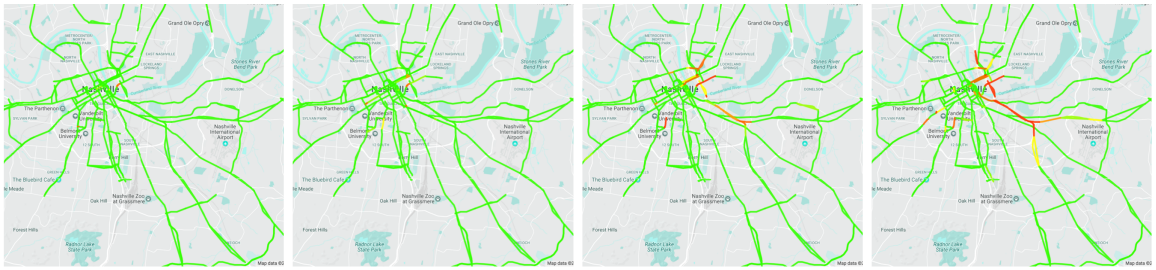
The existing work on traffic event detection focused on analyzing traffic videos or traffic sensor data streams to detect events that are directly related to traffic, such as vehicle stopping, car accidents, and road congestion. But few studies explored the contextual non-recurrent events whose impacts are also highly associated with certain traffic patterns.

Recently there has been an explosion in research of using deep neural networks. But still, few have applied deep learning on studying traffic patterns. Deep learning techniques have gained great success in research fields like image processing, speech recognition, bioinformatics, etc. Convolutional neural networks are similar to original neural networks but convolutional layers are added in the front of the model to learn patterns in the original images. If traffic and label data can be converted to images, then CNN can be employed to learn their labeled patterns. It is still a research gap of how to develop an effective and efficient deep learning network for identifying and classifying traffic patterns of non-recurring events.

We formulate the problem of identifying the specific traffic patterns associated with events in Section 5.5.1, and then present the details of our proposed approach that uses

Table 5.1: The information of the eight football games studied in the motivating example

Date	Start Time (CST)	Stadium	Attendance	Duration (HH:MM)
1/1/17	12:00 PM	Nissan Stadium	65205	3:11
12/11/16	12:00 PM	Nissan Stadium	68780	3:02
11/13/16	12:00 PM	Nissan Stadium	69116	3:36
10/27/16	7:26 PM	Nissan Stadium	61619	3:08
10/23/16	12:02 PM	Nissan Stadium	65470	3:21
10/16/16	12:02 PM	Nissan Stadium	60897	3:12
9/25/16	12:02 PM	Nissan Stadium	62370	3:03
9/11/16	12:05 PM	Nissan Stadium	63816	2:57



(a) (b) (c) (d)

Figure 5.1: Impact of football games on traffic congestion in four one-hour time windows before football games: (a) from 4 hours to 3 hours, (b) from 3 hours to 2 hours, (c) from 2 hours to 1 hour, (d) from 1 hour to 0 hour.

convolutional neural networks in Section 5.5.3.

5.3 Motivating Example

This section describes a motivating example in which we use the collected datasets to study the impact of football games on the traffic congestion in the city.

The motivation for our research comes from a brief experiment, in which we study the impact of football games on the traffic congestion in the city.

During the studied period between Sept. 1, 2016 and Jan. 1, 2017, there were eight football games (as listed in Table 5.1) at the Nissan Stadium at downtown Nashville. During this time, we collected data related to traffic (speed limit, real-time speed) and the football games (date, start time, duration, location)[1].

[1]Our dataset is larger. However, in this study we are focusing on these 4 months

To indicate the congestion condition, HERE [34] provides a jam factor (JF) that ranges between 0.0 and 10.0 for each TMC road segment. In this study we compare the JF between the days when there is a football game and the days when there is no football game, during four one-hour time window directly before the games: $[-4,-3]$, $[-3,-2]$, $[-2,-1]$, and $[-1,0]$ relative to the time when the game was scheduled[2].

As shown in Figure 5.1, the results of the JF difference on road segments in different time windows are visualized using heat maps. In the figure, colors ranging from green to red are used to indicate the small and big JF differences. The results show that the impact of football games on traffic congestion begins to increase from 4 hours before games. We have observed this pattern across several game events in the city. Our hypothesis is that every event has a unique pattern and we can learn that pattern over time and use it to identify if a current congestion pattern matches with the expected pattern. If the pattern does not match then we can classify it as an anomaly.

5.4 Problem Formulation

In this section, we first provide a formal definition of the problem and then describe the assumptions for solving the problem.

5.4.1 Definition

The goal of this research is to model traffic patterns around the locations of non-recurrent events so that we can use the model to identify non-recurring congestion and its causes. The traffic pattern that we use here refers to the spatiotemporal relations of traffic speeds on many road segments in an area, which can be modeled and detected by a classifier. The definitions of all relative notions can be found in Table 5.2.

The inputs to the system are data about traffic and events. Since the traffic data that we collected from HERE API is defined using Traffic Message Channel Location Code [140]

[2]Most football games are scheduled at 1 PM.

Table 5.2: Symbols used in the formulated problem

T	a timestamp
t_{day}	time in the day in seconds
t_{week}	weekdays encoded using integers (e.g., 0 for Sunday, 1 for Monday, etc.)
t_{event}	time windows relative to events (e.g., 1 for the 1-hour time window before events)
l_{event}	indicator of whether the current time is within a time window near the occurrence of an event
S_e	a set of events in the city
r	a road segment
S_r	a set of road segments defined by TMC location codes
S_t	a set of traffic data that contains speed limit and real-time speed for a set of road segments S_r
TMC_{key}	a string representing a road segment in S_r
TCI	traffic Condition Image, a gray-scale image to represent traffic conditions in a bounding box
I_w	the width of a TCI
p	a pixel in TCI. Its value shows the normalized traffic speed on a road segment
v_r	the real-time traffic speed (miles per hour) on a road segment r
TH	a threshold for the classifier to determine whether the input traffic data contains recurring or non-recurring congestion

format (a standard for encoding geographic information), the road segments used in this study are also defined using the same TMC location codes. Event data is categorized with labels for training and validating purposes. The labels used are as follows:

- Event-related: event indicator l_{event}, time window relative to the event t_{event}
- Time-related: time in the day t_{day}, weekday t_{week}

One of the key differences between our approach and the existing ones is that we are trying to visualize the wide area sensor data distribution as Traffic Condition Images (TCI), so that we can use CNN and other deep learning techniques to analyze and model the spatiotemporal relations. TCI is a I_w by I_w pixels image. Each pixel p corresponds to a road segment in the real world and the grayscale value of each pixel represents the real-time traffic speed v_r of the road segment r.

Formulation of the Non-recurring Congestion Identification Problem. Given a set of traffic data S_t that contains speed limit and real-time speed for a set of road segments S_r at a specific time t_{day} on weekday t_{week}, and a set of event labels S_e, the model should determine l_{event} that indicates whether the given traffic data contains congestion caused by a subset $S_e^{'}$ of event set S_e. If l_{event} is true, the model should also provide the time window t_{event} relative to events (i.e. t_{event} can be used to estimate the event occurrence time).

Figure 5.2 illustrates an example of the problem. Given raw traffic data at a specific time, the model should identify the possible non-recurring congestion and also provide an estimation of the event occurrence time.

5.4.2 Assumptions

The following assumptions are made when we design and formulate the non-recurring traffic congestion identification system:

- We assume the availability of both traffic speed data and event information for the studied area and period.

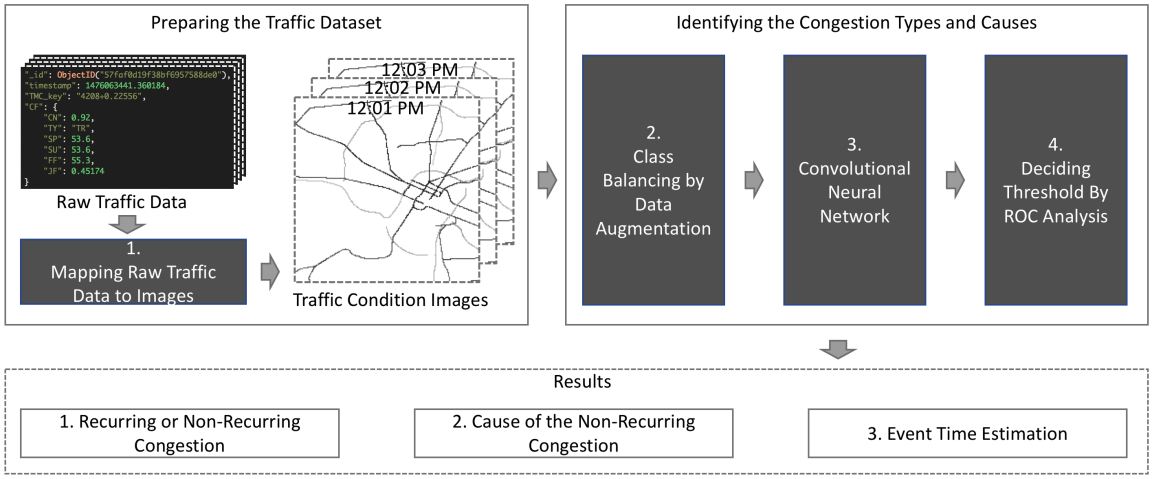


Figure 5.2: Overall workflow of the non-recurring congestion identification system

- We assume the traffic condition on a short road segment in a direction is the same anywhere on the segment.
- We assume that an event happening in the urban environment will affect the traffic conditions of nearby road segments.
- We assume that there is a robust correlation between the road segments affected by an event, and that the patterns can be identified by the image classification techniques of deep learning.

5.5 Our Approach

In order to identify the specific traffic patterns associated with non-recurring events as defined in Section 5.4, we present the details of our proposed approach in this section. The overall workflow of the system is shown in Figure 5.2. There are three key components in the system: (1) an algorithm that converts raw traffic data to images, (2) a convolutional neural network that classifies the traffic condition images, (3) ROC analysis that tunes the classification threshold to reduce the false positive and false negative rates.

5.5.1 Feature Extraction by Mapping Traffic Data to Images

Research challenge 1 describes the problem that feature vectors have limitations when representing urban data. To solve this issue, the first step is to convert the collected traffic data into images. We have been collecting real-time traffic data of Nashville area from Here Traffic API [34] since Oct. 2016. The traffic data is encoded in TMC location codes. Since the TMC database is not open to the public, here we present an algorithm to convert traffic data for a specific time T coded by TMC locations to traffic images. In order to project the traffic conditions to the pixels of images, we first initialize a gridded map and then re-sample the road segments defined by TMC location codes to the grids. The algorithm's input, output, and step details are as follows (for a set of road segments S_r, step 1 and 2 will run only once, but step 3 will run once for each timestamp):

Input: Traffic dataset S_t, road segment set S_r, and timestamp T. The raw traffic data of road segments S_r for timestamp T is queried from Traffic dataset S_t in the database.

Output: A Traffic Condition Image (TCI).

Step 1: Map grid initialization. The map of the area containing the road segment set S_r is divided into a map grid of squares. The length of each square is about 8.97 meters, so each grid cell covers about 80.51 square meters on the map.

Step 2: Road segment path re-sampling and smoothing. The points from road segments are re-sampled to the centers of grid cells if the points are covered by the cell. Also, if the distance between two original points is large enough that there are blank cells between the two cells projected by the two points, then points will be interpolated to fill the blank cells. After this step, we get a two-dimensional array, in which each cell contains a list of TMC keys TMC_{key} corresponding to points from road segments.

Step 3: Traffic data projection to the images. The two-dimensional array acts as a projecting table from original road segments to the image pixels. Now we can fill the images with traffic data by querying the traffic data using segment keys TMC_{key} and timestamp T. We use the following equation to convert a traffic speed to a pixel value:

$$p = \begin{cases} \frac{(80 - v_s^t) * 255}{80}, & \text{if } 0 \leqslant v_s^t \leqslant 80 \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

where p denotes the pixel value (0-255) and v_s^t denotes the real-time speed (miles per hour).

After getting initial projected TCI, simple image processing techniques are used to resize TCI to the desired size (I_w by I_w).

5.5.2 Data Augmentation by Crossover Operations

TCI is our image representation of traffic speeds on road segments. Since traffic data is collected every minute, without data augmentation we can only get 1440 (the number of minutes in a day) TCIs for one day, which is usually not enough for training deep learning image processing models. To address research challenge 2 (i.e., the lack of enough traffic data with labels), we create a crossover operator to generate more labeled traffic condition images for training deep learning models. Crossover is originally a genetic operator from genetic algorithms to vary the chromosomes of individuals from one generation to another. We are motivated by a similar idea and present the crossover operator for our system:

1. *Getting TCI candidates.* For a given timestamp T, instead of only getting one TCI for T, we generate n TCIs for time range $[T-t, T]$ (t denotes a time length to extend T, e.g., 3 minutes). While these TCIs are the same in image size, they differ in the pixel values because they correspond to traffic speeds at different times.

2. *Generating new labeled TCIs.* While looping through the pixels in TCI, for each pixel row there is a probability p_m that its values will mutate and randomly select a new row from the same corresponding pixels in other TCI candidates. After the second step, we get a new TCI. Because we assume traffic conditions within a small time range are caused by the same events, we can give the new TCI the same event label.

The crossover operator can be executed for many times to generate many new data instances. Through crossover, we not only have more labeled data, but also reduce the probability of over-fitting in the training phase.

5.5.3 Classifying Non-Recurring Congestion

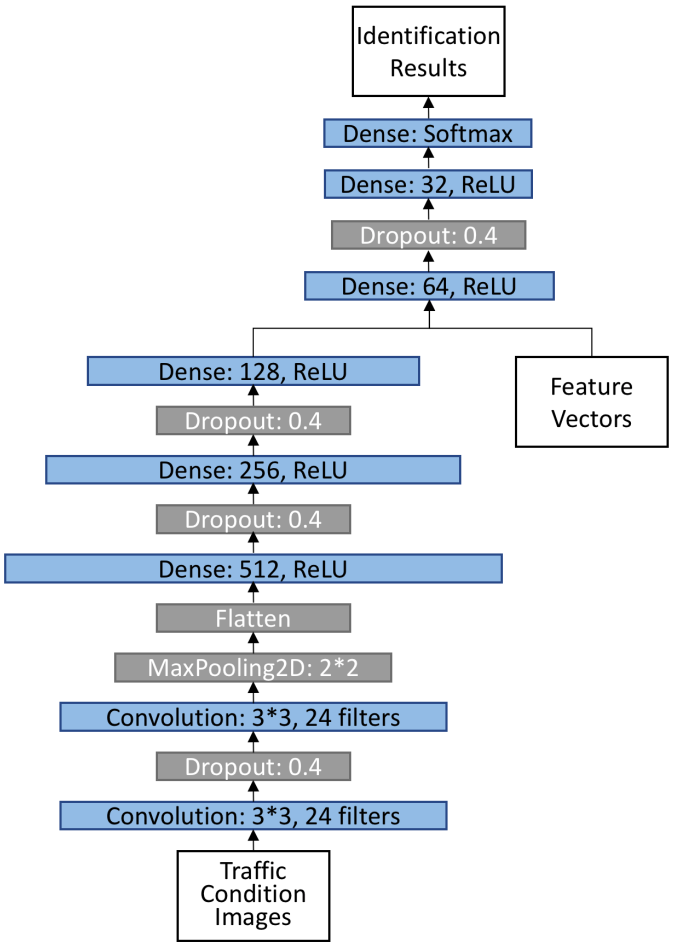


Figure 5.3: Our proposed convolutional neural network (CNN).

In the previous section, we have described an algorithm that converts raw traffic data to TCI. Since the inputs contain images, it makes sense to apply convolutional neural networks. This section introduces our CNN model to classify the TCI using event labels. CNN is a class of deep and feed-forward artificial neural networks that have shown great success in image analysis tasks. Here we apply CNN to our problem that assigns event and

congestion labels to a TCI.

CNN. The architecture of the proposed CNN is shown in Figure 5.3. Generally, the model consists of a stack of convolutional, fully-connected neural, dropout and max-pooling layers. Dropout layers are used throughout the model to prevent over fitting. Max pooling layers are used for spatial down-sampling. In the middle of the CNN, feature vectors that represent time of day and day of week that correspond to the TCI are concatenated to be input into the CNN to help it make better decisions. Details of the layer configuration, such as dimension, activation function, and dropout rate, can be found in the Figure 5.3. Since we use one-hot encoding and the vectors are in categorical format (i.e., dimensional vector is all-zeros except for a one at the index corresponding to the class of the sample), categorical cross entropy is used as the loss function to train the model.

One-hot Encoding. In the proposed CNN model, the input feature vectors are time in the day and weekday, and output labels are (1) whether the congestion in input TCI is recurring or non-recurring (2) the relative time windows that the TCI belongs to if it is non-recurring congestion. We use one-hot encoding to convert both input and output vectors to binary class matrix. The input matrix has 31 classes, in which 24 classes correspond to 24 hours and 7 classes correspond to 7 days of the week. As illustrated in Figure 5.4, the output matrix has several classes, of which the first class represents whether the traffic condition belongs to recurring congestion or non-recurring congestion, and the next classes represent time windows before and after events. The first class is tunable since it directly determines whether the input traffic condition contains non-recurring congestion or not. If the value of the first class output is higher than a predefined threshold TH, then the classifier will output that the input traffic data does not contain non-recurring congestion (even if the values of other classes are higher than the first class). The details of the tuning steps are presented in the following section.

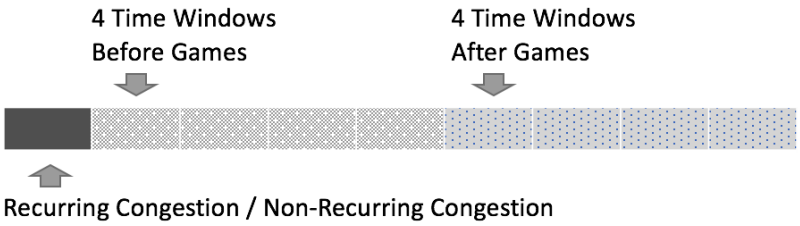


Figure 5.4: An example of the one-hot encoding format. Event labels are encoded using 9 classes: (1) first digit represents whether the traffic condition belongs to recurring congestion or non-recurring congestion, (2) if it's non-recurring congestion, the next 8 digits represent 8 time windows before and after events.

5.5.4 Tuning the Model Sensitivity by ROC Analysis

Our approach uses receiver operating characteristic (ROC) analysis to tune the sensitivity of the CNN classifier. ROC is a statistical plot that illustrates the diagnostic ability of a classifier system [141]. The ROC curve is a fundamental tool for diagnostic test evaluation. In an ROC curve, the true positive rate (TPR) is plotted in a function of the false positive rate (FPR). In machine learning, TPR represents sensitivity, recall or probability of detection, and FPR represents fall-out or false alarm [142]. By choosing a point from the curve, corresponding classification threshold can be decided.

In our model, the non-recurring congestion is considered as positive output and the recurring congestion is negative output.

We use the ROC analysis to tune the classification threshold that decides whether the traffic congestion in the input traffic data is recurring congestion or non-recurring congestion. We choose thresholds that range from 0.01 to 1.00 and the corresponding FPR and TPR of the training dataset are plotted (an example is shown in Figure 5.5). The curve's nearest point to the point (FPR: 0.0, TPR: 1.0) will be selected.

5.6 Experiments

In this section, we evaluate the proposed deep neural network's ability to identify non-recurring traffic anomalies by using real-world data of three event types: football games,

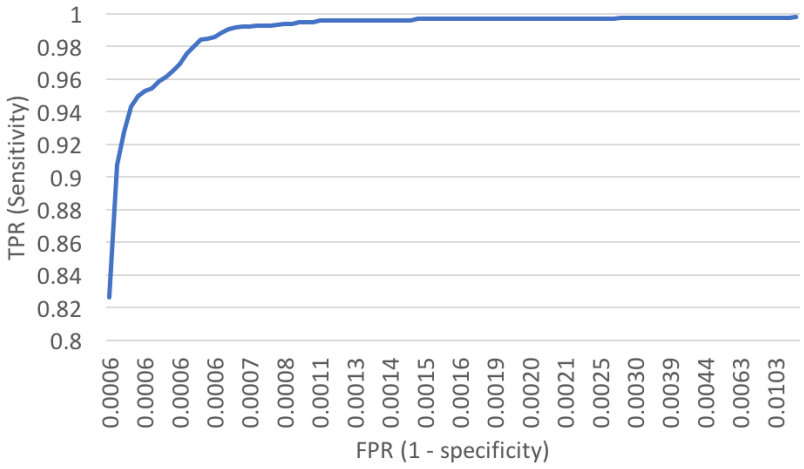


Figure 5.5: Receiver operating characteristics (ROC) curve analysis on the prediction threshold.

hockey games, and traffic incidents. Keras [96] Python deep learning library is used to construct the models and TensorFlow [143] is selected as the tensor manipulation library.

5.6.1 Scenarios

As illustrated in Figure 5.6, we create three scenarios to test the performance of the proposed model. In each scenario, we consider one of the three event categories for training and validating the proposed model:

- *Football Games.* Between Oct. 11, 2016, and Jan. 1, 2017, there were 8 NFL football games at the Nissan Stadium in Nashville. The traffic data in the bounding box (latitude range: [36.1120, 36.2052], longitude range: [-86.8475, -86.7543]) is used.

- *Hockey Games.* Between Oct. 14, 2016, and Jan. 03, 2017, there were 20 NHL hockey games at the Bridgestone Arena in Nashville. The traffic data in the bounding box (latitude range: [36.1237, 36.1936], longitude range: -86.8359, -86.7660]) is used.

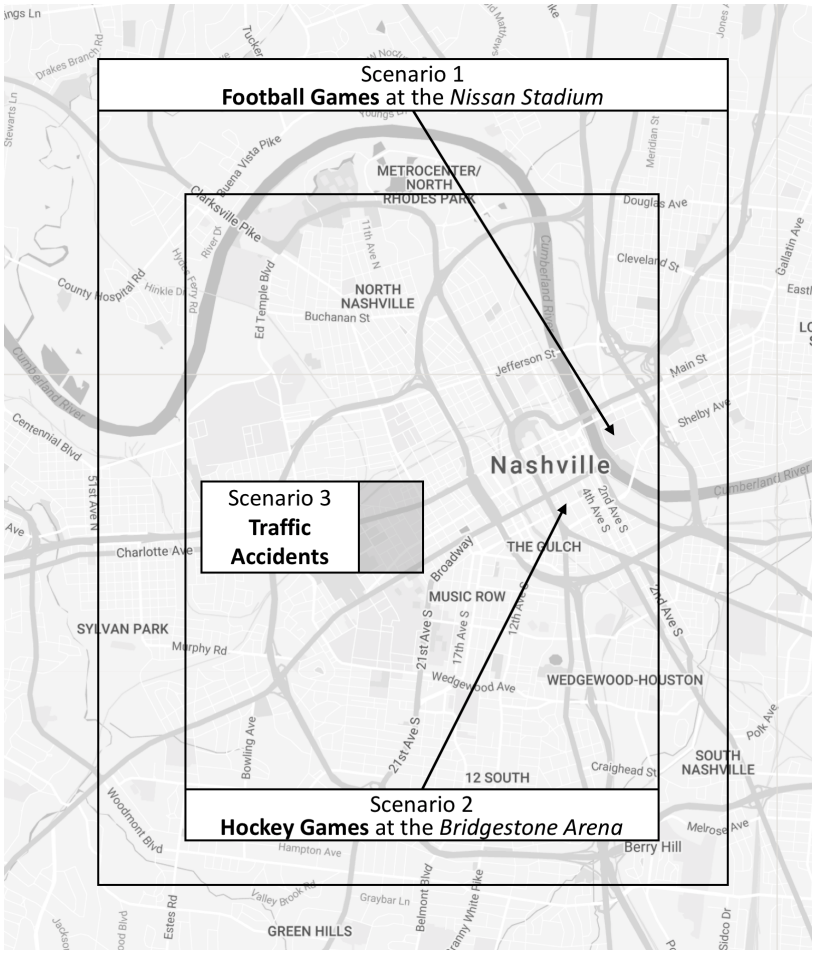


Figure 5.6: Experimental scenarios and their coverage areas: (1) detecting NRC caused by football games, (2) detecting NRC caused by hockey games, (3) detecting NRC caused by traffic accidents.

- *Traffic Accidents.* Between Oct. 15, 2016, and Mar. 10, 2017, there were 23 traffic accidents at the selected block area. The traffic data in the bounding box (latitude range: [36.1470, 36.1586], longitude range: [-86.8126, -86.8009]) is used.

The traffic and event datasets are divided into two subsets for training and validation. The event and time information is encoded using one-hot encoding as described in Section 5.5. An example of the output of our model is shown in Figure 5.4. Particularly, the following metrics are used to define if an output is positive or negative: (1) an output is considered to be positive if it determines that the input TCI contains non-recurring congestion, (2) an output is negative if it determines the input TCI only contains recurring congestion.

5.6.2 Experiment 1: Identifying NRC Caused by Football Games

As shown in the motivating example in Section 5.3, the 8 selected NFL football games have an average attendance of over 60,000 people, which shows a great impact on causing non-recurring traffic congestion. In the first scenario, we use the traffic data collected in 1-minute intervals between Oct. 11, 2016 and Jan. 1, 2017. Traffic data of 5 non-game days and two game days are used as the training dataset, and one non-game day and one game day are used as the validating dataset. As a comparison with the traditional machine learning techniques, we build a random forest model that uses the same training and validating dataset. Because random forests cannot use images directly as input, we first convert the traffic condition images to one-dimensional vectors, and then concentrate the traffic vectors with time of the day and day of the week vectors, and finally use the combined feature vector as input to the random forest model.

The accuracy, false positive rate (FPR) and false negative rate (FNR) of our model and the random forest model are shown in Table 5.5. Our model outperforms the random forest model with higher accuracy and lower FPR and FNR.

Table 5.3: Experiment results in scenario 1: identifying NRC caused by football games

	Accuracy	FPR	FNR
DxNAT	98.73%	1.57%	0.17%
Random Forest	84.06%	6.25%	2.17%

Table 5.4: Experiment results in scenario 2: identifying NRC caused by hockey games

	Accuracy	FPR	FNR
DxNAT	90.76%	8.11%	23.19%

5.6.3 Experiment 2: Identifying NRC Caused by Hockey Games

Compared with NFL football games, NHL hockey games in Nashville usually have less attendance (NHL 10,000 v.s. NFL 60,000). So we assume that an NHL hockey game has less impact on traffic conditions and it will be more difficult to detect the NRC related to hockey games.

In Scenario 2, we use traffic and hockey games data between Oct. 14, 2016, and Nov. 30, 2016, as the training dataset, and data of Dec. 15, 2016 (game day) and Dec. 16, 2016 (non-game day) as the validating dataset. The accuracy, FPR and FNR results are shown in Table 5.4. Compared with the results in Scenario 1, the model has lower accuracy and higher FNR. Our assumption is validated that the NRC associated with hockey games with less attendance is harder to be detected.

5.6.4 Experiment 3: Identifying NRC Caused by Traffic Accidents

In scenario 3, we explore the model's ability to detect NRC caused by road accidents. For the selected block area, there were eight traffic accidents on 7 different days between Oct. 18, 2016, and Dec. 13, 2016. We use six days with accidents as the training dataset and one day with an accident as the validating dataset. The DxNet model archives an accuracy of 86.59% with FPR of 13.71% and FNR of 4.44%

Table 5.5: Experiment results in scenario 3: identifying NRC caused by traffic accidents

	Accuracy	FPR	FNR
DxNAT	86.59%	13.71%	4.44%

Table 5.6: Summary of architectural decisions

Challenge	Approach	Section
Representing Heterogeneous Traffic Data and Event Labels	Using Multi-dimensional Images	5.2.2
Training Deep Learning Models Using Limited Data Instances	Developing crossover operator on original data	5.2.3
Modeling Traffic Patterns of Non-Recurring Events	Employing convolutional neural networks	5.2.4

5.7 Conclusion

In this chapter, we propose a deep neural network model to identify non-recurring traffic congestion and explain its causes. To our best knowledge, our work is one of the first efforts to utilize deep learning techniques to study traffic congestion patterns and explain non-recurring congestion using events. Our main contributions are listed in Table 5.6. Particularly, we present an algorithm to efficiently convert traffic data in Traffic Message Channel (TMC) format to images, as well as a data augmentation mechanism using crossover operators for class balancing. A convolutional neural network is proposed to identify non-recurring traffic anomalies that are caused by events. We evaluate the proposed model by using three types of events (football games, hockey games, and traffic incidents).

Publication. This work has been published in the following place:

- Fangzhou Sun, Abhishek Dubey, Hiba Baroud, Chetan S. Kulkarni, Chinmaya Samal, Short-term Transit Decision Support System Using Multi-task Deep Neural Networks, The 4th IEEE International Conference on Smart Computing (SMARTCOMP 2018), June 18-20, 2018, Taormina, Sicily, Italy.

CHAPTER 6

CONCLUDING REMARKS

Urban mobility networks are vital for people living in the city. Communities are now embracing data-driven smart solutions, which aim to improve the utilization, management, and effectiveness of available transportation options. However, the current state of the art approaches face many challenges that arise due to the heterogeneity, sparsity, and noise in the data collected in the urban environment. This thesis identified three key research challenges to build robust urban mobility networks: (1) predicting the transit delay using data with sparsity issues for multiple timescales, and identifying significant predictor variables from the environment, (2) optimizing the performance of transit networks under uncertainty in order to understand and mitigate the schedule of public transit considering seasonal delays, and (3) identifying and explaining the anomalous operations of transit networks over a large metropolitan area. A summary of research contributions are discussed below.

6.1 Summary of the Research Contributions

• Robust arrival delay prediction models that solve data sparsity issue, and a multivariate predictive model to explore the significance of contextual predictors:

1. Presented a real-time delay prediction model that combines clustering analysis and Kalman filters and uses real-time data from shared route segments.

2. Provided an algorithm that generates shared bus route segment networks from standard General Transit Feed Specification (GTFS) datasets.

3. Showed the efficacy of our real-time delay prediction model. When predicting the travel time delay of segments 15 minutes ahead of scheduled time, our model reduced

the root-mean-square deviation (RMSD) by about 30% to 65% compared with a SVM-Kalman model [15].

4. Proposed a generic tool-chain that takes transit feed (in standard and real-time GTFS format), forecasted weather condition, and time as input, is developed to provide expected delays and service alerts as output for short-term.

5. Designed a multi-task deep neural network architecture that consumes contextual information in the augmented datasets and makes delay predictions for nearby segments in a bounding box all at once.

6. Illustrated how the analytical algorithms can be packaged into independently deployable and self-contained micro-services.

7. Proposed multivariate linear regression models and random forest models that not only utilize various factors from traffic and weather to predict bus delay but also identify the significant predictors.

• Optimizing the performance of transit network under uncertainty that comes from many internal and external factors:

1. Described an unsupervised mechanism to find out how months can be grouped to generate new timetables.

2. Presented greedy, genetic and PSO algorithms to optimize the scheduled arrival and departure time at timepoints to maximize the probability of bus trips that reach the desired on-time range.

3. Described sensitivity analyses which provide practical strategies to select optimal hyper-parameters.

4. Evaluated the proposed mechanisms via simulation using real data from Nashville bus system. The average on-time performance on all bus routes was improved from

57.79% to 68.93%.

• Modeling traffic patterns and transit delay by considering contextual information to identify anomalies in the system:

1. Described algorithms to map traffic data (i.e. traffic speed on different road segments) as well as event labels (i.e. event features that include type, time, location) to images.

2. Proposed a crossover operator to augment and balance the number of labeled data samples for training purposes.

3. Designed convolutional neural networks (CNN) to identify non-recurring traffic anomalies that are caused by scheduled or detected events.

4. Created three scenarios to evaluate the performance of the proposed model by using real-world data of three events types (football games, hockey games, and traffic incidents).

6.2 Future Work

6.2.1 Distributed Neural Networks for Edge Computing

The conventional way to train a deep neural network model and infer results is usually done on a single computation node, either on the cloud or on an edge device. However, the traditional architecture has some drawbacks and concerns. Firstly, there is a great network communication cost caused by transferring the raw sensor data from edge devices to the cloud. Secondly, since the raw data usually contains sensitive information of users, there may be security and privacy issues caused by transmitting data from edge sensors to the cloud. Thirdly, the forward inference process may cause large latency, considering modern neural networks are trending to become deeper and deeper. For example, compared to the 8-layer AlexNet [144] which was proposed in 2012, the ResNet that presented in 2015

already has 152 layers. Finally, there are also cost and scalability issues. A neural network running on a single node is expensive to be scaled.

Distributing the neural networks to edge devices can be a solution to the drawbacks and concerns. Existing work mostly focuses on reducing training time by utilizing CPU or GPU clusters [145, 146, 147] on the cloud, where neural networks are not actually distributed in a geographically distributed network. To improve the inferring efficiency of neural networks, Teerapittayanonet et al. presented a distributed deep neural networks [148] that used distributed computing hierarchies for more efficient classification on three layers (local, edge and cloud). Their work is based on an early exiting technique for removing the inference time of neural networks on the cloud [149]. However, Their study simplifies the problem by assuming (1) the architecture of local, edge and cloud nodes are pre-defined and fixed (e.g., in the experiment they assume there are six devices associated with an edge device) (2) the neural networks are divided and offloaded exactly by the number of edge and local nodes. In the real world, the nodes are usually dynamically changing and some mechanisms are needed to offload small neural networks to the proper edge and local nodes.

6.2.2 Context-aware Anomaly Detection with Rich Features and Fine-tuned Bounding Boxes

Integrating more contextual features. Existing work usually focuses just on traffic data, but there are many types of urban data available to help identify traffic patterns, like real-time bus travel time, speed, and weather. Besides events, traffic conditions are affected by multiple environmental factors. The current work only considers time of day and day of week as the environmental training features. In the next step, we will include various new features like weather conditions (such as humidity, nearest storm distance, visibility, etc.).

Identifying sizes of block areas and length of time windows. For each event type (e.g., sports games, accidents), the size of impacting block areas as well as the number of impacting time windows in the experimental scenarios are selected arbitrarily. A mechanism

is needed to automatically select the best impacting area size and time windows for each event type.

6.2.3 Transfer Learning as Another Potential Solution for Data Sparsity Issue

In our research, several types of neural networks are proposed for tasks like delay prediction and anomaly identification. These models are trained and tuned using months of urban data. However, in practice very few deep neural networks (especially convolutional networks) are trained completely from scratch since it is relatively difficult or takes a long time to get sufficient training datasets. Transfer learning is a method for solving data availability issues. In the domain of public transportation, areas that share similar spatio-temporal characters (e.g., distance to downtown, population, demand trends, etc.) are probably able to share the trained models and only small post-training or post-tuning is needed. Future works related to transfer learning include: (1) pretraining models for different types of urban environments, and then testing their performance as a baseline for new areas that share similar characteristics, (2) using the trained models as fixed feature extractors and re-training the last fully-connected layer for new datasets.

6.3 Summary of Publications

- Journal and Book Chapter Publications:

1. Fangzhou Sun, Abhishek Dubey, Jules White, Aniruddha Gokhale, Transit-Hub: A Smart Public Transportation Decision Support System with Multi-Timescale Analytical Services, Journal of Cluster Computing, Special Issue on Dynamic Data Driven Applications Systems (DDDAS)

2. Chinmaya Samal, Liyuan Zheng, Fangzhou Sun, Lillian J. Ratliff, Abhishek Dubey, Towards a Socially Optimal Multi-Modal Routing Platform, ACM Transactions on Cyber-Physical Systems (TCPS) (Under Review)

3. Yao Pan, Fangzhou Sun, Jules White, Douglas Schmidt, Jacob Staples, Lee Krause, Detecting Web Attacks with End-to-End Deep Learning, IEEE Transactions on Dependable and Secure Computing (Under Review)

4. Shashank Shekhar, Fangzhou Sun, Abhishek Dubey, Aniruddha Gokhale, Himanshu Neema, Martin Lehofer, Dan Freudberg, Transit Hub: A Smart Decision Support System for Public Transit Operations, Internet of Things and Data Analytics Handbook, John Wiley & Sons, 2016

- Conference Publications:

1. Fangzhou Sun, Abhishek Dubey, Hiba Baroud, Chetan S. Kulkarni, Short-term Transit Decision Support System Using Multi-task Deep Neural Networks, The 4th IEEE International Conference on Smart Computing (SMARTCOMP 2018) (Under Review)

2. Fangzhou Sun, Abhishek Dubey, Jules White, DxNAT - Deep Neural Networks for Explaining Non-Recurring Traffic Congestion, IEEE BigData 2017 - 3rd Special Session on Intelligent Data Mining, December 11-14, 2017, Boston, MA, USA

3. Fangzhou Sun, Yao Pan, Jules White, and Abhishek Dubey, Real-time and Predictive Analytics for Smart Public Transportation Decision Support System, 2016 IEEE International Conference on Smart Computing, May 18-20, 2016, St. Louis, Missouri, USA

4. Fangzhou Sun, Peng Zhang, Jules White, Douglas C. Schmidt, Jacob Staples, and Lee Krause. A Feasibility Study of Autonomically Detecting In-process Cyber-Attacks. 3rd IEEE International Conference on Cybernetics (CYBCONF-2017), Special Session on Cyber Security, June 21-23, 2017, Exeter, UK

5. Fangzhou Sun, Chinmaya Samal, Jules White and Abhishek Dubey, Unsupervised Mechanisms for Optimizing On-Time Performance of Fixed Schedule Transit Vehi-

cles, SMARTCOMP2017: Smart Computing Technologies and Applications, May 29-31, 2017, Hong Kong, China

6. Aparna Oruganti, Fangzhou Sun, Hiba Baroud, Abhishek Dubey, DelayRadar: A Multivariate Predictive Model for Transit Systems, IEEE Big Data 2016 Conference Special Session on Intelligent Data Mining, December 5-8, 2016, Washington D.C. USA

- Workshop Publications:

1. Abhishek Dubey, Ali Guarneros, Fangzhou Sun, Distributed and Stacked Neural Network for Anomaly Detection in Small Satellites, 15th Annual CubeSat Developers Workshop, April 30-May 2, 2018, San Luis Obispo, CA, USA

2. Chinmaya Samal, Fangzhou Sun, Abhishek Dubey, SpeedPro: A Cluster-Based Predictive Model for Urban Traffic Speed Estimation, SmartSys2017: Second International Workshop on Smart Service Systems, May 29-31, 2017, Hong Kong, China

3. Shashank Shekhar, Subhav Pradhan, Fangzhou Sun, Abhishek Dubey, and Annirudha Gokhale, Empowering the Next Generation City-Scale Smart Systems, In Proceedings of the 2015 IEEE 22nd International Conference on High Performance Computing Workshops (HiPCW), December 19-22, Hyderabad, India

- Poster Publications:

1. Abhishek Dubey, Fangzhou Sun, Chinmaya Samal, Anne Zou, Baosen Zhang, Lillian Ratliff, Liyuan Zheng, Tanner Fiez, Socially Optimal Multi-modal Routing Platform, US Ignite Application Summit 2017

2. Fangzhou Sun, Abhishek Dubey, PhD Forum: Robust Sensing and Analytics in Urban Environment, SMARTCOMP2017: Smart Computing Technologies and Applications, 2017

3. Abhishek Dubey, Jules White, Fangzhou Sun, Hiba Baroud, Martin Lehofer, Public Transportation Decision System with Multi-Timescale Analytical Services, 2016 CPS PI Meeting

4. Fangzhou Sun, Abhishek Dubey, PhD Forum: Heterogeneous and Multi-Domain Data Analytics Platforms for Smart Cities, 2016 IEEE International Conference on Smart Computing, May 18-20, 2016, St. Louis, Missouri, USA

5. Abhishek Dubey, Subhav Pradhan, Fangzhou Sun, Aniruddha Gokhale, Resilient Platform for Heterogeneous Big Data Driven CPS, NSF Workshop

6. Abhishek Dubey, Subhav Pradhan, Fangzhou Sun, Aniruddha Gokhale, Gautam Biswas, Martin Lehofer, Dan Freudberg, Platform for Enabling Optimal Multi-Modal Transportation Planning Service, The 2016 Global City Teams Challenge (GCTC) Expo

7. Fangzhou Sun, Shashank Shekhar, Abhishek Dubey, Himanshu Neema, Aniruddha Gokhale, Sandeep Neema, Jules White, Transit Hub Smart Decision Support System for Public Transportation, The 2015 Global City Teams Challenge (GCTC) Expo

BIBLIOGRAPHY

[1] United Nations. World urbanization prospects: The 2014 revision, highlights. department of economic and social affairs. *Population Division, United Nations*, 2014.

[2] U.N. Population Division. Population Distribution, Urbanization, Internal Migration and Development: An International Perspective. page 1, 2011.

[3] David Schrank, Bill Eisele, Tim Lomax, and Jim Bak. 2015 urban mobility scorecard. 2015.

[4] Frank Elswick. How much does it cost to build a mile of road?, 2016. [Online; accessed 12-March-2018].

[5] Elizabeth Mynatt, Jennifer Clark, Greg Hager, Dan Lopresti, Greg Morrisett, Klara Nahrstedt, George Pappas, Shwetak Patel, Jennifer Rexford, Helen Wright, et al. A national research agenda for intelligent infrastructure. *arXiv preprint arXiv:1705.01920*, 2017.

[6] Richard Gilbert and Anthony Perl. *Transport revolutions: moving people and freight without oil*. New Society Publishers, 2010.

[7] Gartner says 8.4 billion connected things will be in use in 2017, up 31 percent from 2016.

[8] Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and Xie Xing. Discovering spatio-temporal causal interactions in traffic data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1010–1018. ACM, 2011.

[9] Andy HF Chow, Alex Santacreu, Ioannis Tsapakis, Garavig Tanasaranond, and Tao

Cheng. Empirical assessment of urban traffic congestion. *Journal of advanced transportation*, 48(8):1000–1016, 2014.

[10] Ranwa Al Mallah, Alejandro Quintero, and Bilal Farooq. Distributed classification of urban congestion using vanet. *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[11] Correction: Broken traffic sensors story. http://www.sandiegouniontribune.com/sdut-dim-traffic-sensors-dull-how-smart-freeways-are-2013nov23-story.html.

[12] Katrin Dziekan and Karl Kottenhoff. Dynamic at-stop real-time information displays for public transport: effects on customers. *Transportation Research Part A: Policy and Practice*, 41(6):489–501, 2007.

[13] Yu Bin, Yang Zhongzhen, and Yao Baozhen. Bus arrival time prediction using support vector machines. *Journal of Intelligent Transportation Systems*, 10(4):151–158, 2006.

[14] Cen Zhang and Jing Teng. Bus dwell time estimation and prediction: A study case in shanghai-china. *Procedia-Social and Behavioral Sciences*, 96:1329–1340, 2013.

[15] Cong Bai, Zhong-Ren Peng, Qing-Chang Lu, and Jian Sun. Dynamic bus travel time prediction models on road with multiple bus routes. *Computational intelligence and neuroscience*, 2015:63, 2015.

[16] Ehsan Mazloumi, Mahmoud Mesbah, Avi Ceder, Sara Moridpour, and Graham Currie. Efficient transit schedule design of timing points: a comparison of ant colony and genetic algorithms. *Transportation Research Part B: Methodological*, 46(1):217–234, 2012.

[17] Maged Dessouky, Randolph Hall, Ali Nowroozi, and Karen Mourikas. Bus dispatch-

ing at timed transfer transit stations using bus tracking technology. *Transportation Research Part C: Emerging Technologies*, 7(4):187–208, 1999.

[18] Liping Fu, Qing Liu, and Paul Calamai. Real-time optimization model for dynamic scheduling of transit operations. *Transportation Research Record: Journal of the Transportation Research Board*, (1857):48–55, 2003.

[19] Aichong Sun and Mark Hickman. The real–time stop–skipping problem. *Journal of Intelligent Transportation Systems*, 9(2):91–109, 2005.

[20] Unit cost entry. http://www.itsknowledgeresources.its.dot.gov/ITS/benecost.nsf/ID/1E126E4A3607BD0985257B1E00553DD5?OpenDocument&Query=CApp.

[21] NIST. Global city teams challenge, 2017. [Online; accessed 18-September-2017].

[22] Xiao-Yun Lu, Pravin Varaiya, Roberto Horowitz, and Joe Palen. Faulty loop data analysis/correction and loop fault detection. In *15th World Congress on Intelligent Transport Systems and ITS America's 2008 Annual Meeting*, 2008.

[23] Amin Ghafouri, Aron Laszka, Abhishek Dubey, and Xenofon Koutsoukos. Optimal detection of faulty traffic sensors used in route planning. In *Proceedings of the 2nd International Workshop on Science of Smart City Operations and Platforms Engineering*, pages 1–6. ACM, 2017.

[24] Top reasons people stop using public transit. http://www.governing.com/blogs/view/gov-reasons-riders-abandon-public-transit.html.

[25] John Bates, John Polak, Peter Jones, and Andrew Cook. The valuation of reliability for personal travel. *Transportation Research Part E: Logistics and Transportation Review*, 37(2):191–229, 2001.

[26] Lin Xu, Yang Yue, and Qingquan Li. Identifying urban traffic congestion pattern

from historical floating car data. *Procedia-Social and Behavioral Sciences*, 96:2084–2095, 2013.

[27] Simon Kwoczek, Sergio Di Martino, and Wolfgang Nejdl. Predicting and visualizing traffic congestion in the presence of planned special events. *Journal of Visual Languages & Computing*, 25(6):973–980, 2014.

[28] Randolph W Hall. Non-recurrent congestion: How big is the problem? are traveler information systems the solution? *Transportation Research Part C: Emerging Technologies*, 1(1):89–103, 1993.

[29] Eric J Horvitz, Johnson Apacible, Raman Sarin, and Lin Liao. Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service. *arXiv preprint arXiv:1207.1352*, 2012.

[30] Simon Kwoczek, Sergio Di Martino, and Wolfgang Nejdl. Stuck around the stadium? an approach to identify road segments affected by planned special events. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, pages 1255–1260. IEEE, 2015.

[31] Xiaoyan Zhang and John A Rice. Short-term travel time prediction. *Transportation Research Part C: Emerging Technologies*, 11(3):187–210, 2003.

[32] JWC Van Lint. Online learning solutions for freeway travel time prediction. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):38–47, 2008.

[33] Jaimyoung Kwon, Michael Mauch, and Pravin Varaiya. Components of congestion: Delay from incidents, special events, lane closures, weather, potential ramp metering gain, and excess demand. *Transportation Research Record: Journal of the Transportation Research Board*, (1959):84–91, 2006.

[34] Here traffic api. https://developer.here.com/rest-apis/documentation/traffic/topics_v6.1/flow.html.

[35] 2017 Metropolitan Government of Nashville and Davidson County. Nashville fire department, 2017. [Online; accessed 30-September-2017].

[36] General transit feed specification (gtfs) static overview. https://developers.google.com/transit/gtfs/, 2016. Accessed: 2016-09-18.

[37] General transit feed specification (gtfs) real-time overview. https://developers.google.com/transit/gtfs-realtime/, 2016. Accessed: 2016-09-18.

[38] The mongodb 3.2 manual. https://docs.mongodb.com/manual/, 2016. Accessed: 2016-09-25.

[39] Fangzhou Sun, Yao Pan, Jules White, and Abhishek Dubey. Real-time and predictive analytics for smart public transportation decision support system. *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8, 2016.

[40] Fangzhou Sun, Abhishek Dubey, Hiba Baroud, and Chetan Kulkarni. Short-term transit decision support system using multi-task deep neural networks (under review). In *Smart Computing (SMARTCOMP), 2018 IEEE International Conference on*. IEEE, 2018.

[41] Shashank Shekhar, Fangzhou Sun, Abhishek Dubey, Aniruddha Gokhale, Himanshu Neema, Martin Lehofer, and Dan Freudberg. Transit hub. *Internet of Things and Data Analytics Handbook*, pages 597–612, 2016.

[42] Fangzhou Sun, Abhishek Dubey, Jules White, and Aniruddha Gokhale. Transit-hub: A smart public transportation decision support system with multi-timescale analytical services. *Cluster Computing*, 2017.

[43] American Public Transportation Association (APTA). Americans took 10.6 billion trips on public transportation in 2015, 2016.

[44] American Public Transportation Association (APTA). Record 10.7 billion trips taken on u.s. public transportation in 2013, 2014.

[45] Federal Highway Administration. Travel monitoring and traffic volume, 2014.

[46] Aaron Gooze, Kari Watkins, and Alan Borning. Benefits of real-time transit information and impacts of data accuracy on rider experience. *Transportation Research Record: Journal of the Transportation Research Board*, (2351):95–103, 2013.

[47] nmotion 2015. https://www.surveymonkey.com/r/3KVYKVL?sm=UBsMYkzrQRFWv3mJyRiN5Q%3d%3d, 2015. Accessed: 2017-12-31.

[48] Real-time port authority bus tracking system not always real. http://www.post-gazette.com/news/transportation/2014/10/16/Real-time-Port-Authority-tracking-not-always-real/stories/201410160155, 2014. Accessed: 2016-09-30.

[49] Cota says its real-time bus-tracking system doesn't work. http://www.dispatch.com/content/stories/local/2014/07/23/COTA-says-its-GPS-system-doesnt-work.html, 2014. Accessed: 2016-09-30.

[50] Ceder Avishai. Public transit planning and operation theory." modelling and practice. *Public transit planning and operation, Civil and Environmental Faculty*, 2007.

[51] SB Pattnaik, S Mohan, and VM Tom. Urban bus transit route network design using genetic algorithm. *Journal of transportation engineering*, 124(4):368–375, 1998.

[52] Partha Chakroborty. Genetic algorithms for optimal urban transit network design. *Computer-Aided Civil and Infrastructure Engineering*, 18(3):184–200, 2003.

[53] Yang Hairong and Luo Dayong. Optimal regional bus timetables using improved genetic algorithm. In *Intelligent Computation Technology and Automation, 2009. ICICTA'09. Second International Conference on*, volume 3, pages 213–216. IEEE, 2009.

[54] Jing-Quan Li, Myoung Song, Meng Li, and Wei-Bin Zhang. Planning for bus rapid transit in single dedicated bus lane. *Transportation Research Record: Journal of the Transportation Research Board*, (2111):76–82, 2009.

[55] Fangzhou Sun, Abhishek Dubey, and Jules White. Dxnat - deep neural networks for explaining non-recurring traffic congestion. In *Big Data (Big Data), 2017 IEEE International Conference on*. IEEE, 2017.

[56] Muhammad Ali Nayeem, Md Khaledur Rahman, and M Sohel Rahman. Transit network design by genetic algorithm with elitism. *Transportation Research Part C: Emerging Technologies*, 46:30–45, 2014.

[57] Wai Yuen Szeto and Yongzhong Wu. A simultaneous bus route design and frequency setting problem for tin shui wai, hong kong. *European Journal of Operational Research*, 209(2):141–155, 2011.

[58] Joana Hora, Teresa Galvão Dias, and Ana Camanho. Improving the robustness of bus schedules using an optimization model. In *Operations Research and Big Data*, pages 79–87. Springer, 2015.

[59] Fangzhou Sun, Chinmaya Samal, Jules White, and Abhishek Dubey. Unsupervised mechanisms for optimizing on-time performance of fixed schedule transit vehicles. In *Smart Computing (SMARTCOMP), 2017 IEEE International Conference on*, pages 1–8. IEEE, 2017.

[60] Bin Yu, William HK Lam, and Mei Lam Tam. Bus arrival time prediction at bus

stop with multiple routes. *Transportation Research Part C: Emerging Technologies*, 19(6):1157–1170, 2011.

[61] Fangzhou Sun, Yao Pan, Jules White, and Abhishek Dubey. Real-time and predictive analytics for smart public transportation decision support system. In *Smart Computing (SMARTCOMP), 2016 IEEE International Conference on*, pages 1–8. IEEE, 2016.

[62] Aparna Oruganti, Fangzhou Sun, Hiba Baroud, and Abhishek Dubey. Delayradar: A multivariate predictive model for transit systems. In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 1799–1806. IEEE, 2016.

[63] Frederica Darema. Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements. *Computational Science-ICCS 2004*, pages 662–669, 2004.

[64] Chun-Hsin Wu, Jan-Ming Ho, and Der-Tsai Lee. Travel-time prediction with support vector regression. *Intelligent Transportation Systems, IEEE Transactions on*, 5(4):276–281, 2004.

[65] Steven I-Jy Chien and Chandra Mouly Kuchipudi. Dynamic travel time prediction with real-time and historic data. *Journal of transportation engineering*, 129(6):608–616, 2003.

[66] Ran Hee Jeong. *The prediction of bus arrival time using automatic vehicle location systems data*. PhD thesis, Texas A&M University, 2005.

[67] Li Weigang, W Koendjbiharie, RC de M Juca, Yaeko Yamashita, and Andrew MacIver. Algorithms for estimating bus arrival times using gps data. In *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, pages 868–873. IEEE, 2002.

[68] Dihua Sun, Hong Luo, Liping Fu, Weining Liu, Xiaoyong Liao, and Min Zhao. Predicting bus arrival time on the basis of global positioning system data. *Transportation Research Record: Journal of the Transportation Research Board*, (2034):62–72, 2007.

[69] Steven I-Jy Chien, Yuqing Ding, and Chienhung Wei. Dynamic bus arrival time prediction with artificial neural networks. *Journal of Transportation Engineering*, 128(5):429–438, 2002.

[70] Jayakrishna Patnaik, Steven Chien, and Athanassios Bladikas. Estimation of bus arrival times using apc data. *Journal of public transportation*, 7(1):1, 2004.

[71] Amer Shalaby and Ali Farhan. Bus travel time prediction model for dynamic operations control and passenger information systems. *Transportation Research Board*, 2, 2003.

[72] Jiann-Shiou Yang. Travel time prediction using the gps test vehicle and kalman filtering techniques. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 2128–2133. IEEE, 2005.

[73] Mei Chen, Xiaobo Liu, Jingxin Xia, and Steven I Chien. A dynamic bus-arrival time prediction model based on apc data. *Computer-Aided Civil and Infrastructure Engineering*, 19(5):364–376, 2004.

[74] Ranhee Jeong and Laurence R Rilett. Bus arrival time prediction using artificial neural network model. In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 988–993. IEEE, 2004.

[75] Ehsan Mazloumi, Sara Moridpour, Graham Currie, and Geoff Rose. Exploring the value of traffic flow data in bus travel time prediction. *Journal of Transportation Engineering*, 138(4):436–446, 2011.

[76] Stuart P Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[77] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[78] Zhenliang Ma, Luis Ferreira, Mahmoud Mesbah, and Sicong Zhu. Modeling distributions of travel time variability for bus operations. *Journal of Advanced Transportation*, 50(1):6–24, 2016.

[79] Fangzhou Sun. Transit hub - shared route segment network generation algorithm. https://github.com/visor-vu/thub-shared-route-segment-network, 2016.

[80] Wenhao Huang, Guojie Song, Haikun Hong, and Kunqing Xie. Deep architecture for traffic flow prediction: deep belief networks with multitask learning. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2191–2201, 2014.

[81] Yuan-yuan Chen, Yisheng Lv, Zhenjiang Li, and Fei-Yue Wang. Long short-term memory model for traffic congestion prediction with online open data. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 132–137. IEEE, 2016.

[82] Yun Wang, Faiz Currim, and Sudha Ram. Deep learning for bus passenger demand prediction using big data. 2016.

[83] Rich Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

[84] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. 2015.

[85] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[86] Ali Abdelfattah and Ata Khan. Models for predicting bus delays. *Transportation Research Record: Journal of the Transportation Research Board*, (1623):8–15, 1998.

[87] George AF Seber and Alan J Lee. *Linear Regression Analysis*. John Wiley & Sons, 2012.

[88] Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)? *Geoscientific Model Development Discussions*, 7:1525–1534, 2014.

[89] Nico JD Nagelkerke. A note on a general definition of the coefficient of determination. *Biometrika*, 78(3):691–692, 1991.

[90] Girish Maskeri Rama and Naineet Patel. Software modularization operators. In *Software Maintenance (ICSM), 2010 IEEE International Conference on*, pages 1–10. IEEE, 2010.

[91] Santonu Sarkar, Shubha Ramachandran, G Sathish Kumar, Madhu K Iyengar, K Rangarajan, and Saravanan Sivagnanam. Modularization of a large-scale business application: A case study. *IEEE software*, 26(2):28–35, 2009.

[92] Sam Newman. *Building Microservices*. " O'Reilly Media, Inc.", 2015.

[93] Ribbon, a inter process communication (remote procedure calls) library. https://github.com/Netflix/ribbon, 2016. Accessed: 2016-09-29.

[94] Rabbitmq. https://www.rabbitmq.com/, 2016. Accessed: 2016-09-24.

[95] Openstack documentation. http://docs.openstack.org/, 2016. Accessed: 2016-09-30.

[96] Keras. Keras: The python deep learning library, 2017. [Online; accessed 30-September-2017].

[97] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[98] John Neff and Matthew Dickens. 2016 public transportation fact book. 2017.

[99] American Public Transportation Association. Ridership report archives. 2017.

[100] HP Benn. Bus route evaluation standards, transit cooperative research program, synthesis of transit practice 10. *Transportation Research Board, Washington, DC*, 1995.

[101] Yadan Yan, Qiang Meng, Shuaian Wang, and Xiucheng Guo. Robust optimization model of schedule design for a fixed bus route. *Transportation Research Part C: Emerging Technologies*, 25:113–121, 2012.

[102] James H Bookbinder and Alain Desilets. Transfer optimization in a transit network. *Transportation science*, 26(2):106–118, 1992.

[103] Wei Fan and Randy B Machemehl. Optimal transit route network design problem with variable transit demand: genetic algorithm approach. *Journal of transportation engineering*, 132(1):40–51, 2006.

[104] Yinghui Wu, Hai Yang, Jiafu Tang, and Yang Yu. Multi-objective re-synchronizing of bus timetable: Model, complexity and solution. *Transportation Research Part C: Emerging Technologies*, 67:149–168, 2016.

[105] AI Diveev and OV Bobr. Variational genetic algorithm for np-hard scheduling problem solution. *Procedia Computer Science*, 103:52–58, 2017.

[106] Stephen A Arhin, Errol C Noel, and Olaoluwa Dairo. Bus stop on-time arrival performance and criteria in a dense urban area. *International Journal of Traffic and Transportation Engineering*, 3(6):233–238, 2014.

[107] Moshe Friedman. A mathematical programming model for optimal scheduling of buses' departures under deterministic conditions. *Transportation Research*, 10(2):83–90, 1976.

[108] Rachel CW Wong, Tony WY Yuen, Kwok Wah Fung, and Janny MY Leung. Optimizing timetable synchronization for rail mass transit. *Transportation Science*, 42(1):57–69, 2008.

[109] Yinghui Wu, Jiafu Tang, Yang Yu, and Zhendong Pan. A stochastic optimization model for transit network timetable design to mitigate the randomness of traveling time by adding slack time. *Transportation Research Part C: Emerging Technologies*, 52:15–31, 2015.

[110] Partha Chakroborty, Kalyanmoy Deb, and PS Subrahmanyam. Optimal scheduling of urban transit systems using genetic algorithms. *Journal of transportation Engineering*, 121(6):544–553, 1995.

[111] Fang Zhao and Xiaogang Zeng. Simulated annealing–genetic algorithm for transit network optimization. *Journal of Computing in Civil Engineering*, 20(1):57–68, 2006.

[112] Surafel Luleseged Tilahun and Hong Choon Ong. Bus timetabling as a fuzzy multiobjective optimization problem using preference-based genetic algorithm. *Promet-Traffic&Transportation*, 24(3):183–191, 2012.

[113] Avishai Ceder, B Golany, and O Tal. Creating bus timetables with maximal synchronization. *Transportation Research Part A: Policy and Practice*, 35(10):913–928, 2001.

[114] Anitha Eranki. A model to create bus timetables to attain maximum synchronization considering waiting times at transfer stops. 2004.

[115] Omar J Ibarra-Rojas and Yasmin A Rios-Solis. Synchronization of bus timetabling. *Transportation Research Part B: Methodological*, 46(5):599–614, 2012.

[116] Weitiao Wu, Ronghui Liu, and Wenzhou Jin. Designing robust schedule coordination scheme for transit networks with safety control margins. *Transportation Research Part B: Methodological*, 93:495–519, 2016.

[117] Ching-Jung Ting and Paul Schonfeld. Schedule coordination in a multiple hub transit network. *Journal of urban planning and development*, 131(2):112–124, 2005.

[118] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, Nov 1995.

[119] Trupti M Kodinariya and Prashant R Makwana. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95, 2013.

[120] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[121] Yannis Marinakis, Georgia-Roumbini Iordanidou, and Magdalene Marinaki. Particle swarm optimization for the vehicle routing problem with stochastic demands. *Applied Soft Computing*, 13(4):1693 – 1704, 2013.

[122] John J Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on systems, man, and cybernetics*, 16(1):122–128, 1986.

[123] Brad L Miller and David E Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3):193–212, 1995.

[124] Gilbert Syswerda. Uniform crossover in genetic algorithms. 1989.

[125] Martín Safe, Jessica Carballido, Ignacio Ponzoni, and Nélida Brignole. On stopping criteria for genetic algorithms. In *Brazilian Symposium on Artificial Intelligence*, pages 405–413. Springer, 2004.

[126] S. Dhabal and S. Sengupta. Efficient design of high pass fir filter using quantum-behaved particle swarm optimization with weighted mean best position. In *Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT)*, pages 1–6, Feb 2015.

[127] Asgarali Bouyer and Abdolreza Hatamlou. An efficient hybrid clustering method based on improved cuckoo optimization and modified particle swarm optimization algorithms. *Applied Soft Computing*, 67:172 – 182, 2018.

[128] Alec Banks, Jonathan Vincent, and Chukwudi Anyakoha. A review of particle swarm optimization. part ii: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing*, 7(1):109–124, Mar 2008.

[129] S Lockwood. The 21st century operation oriented state dots, nchrp project 20–24. *Transportation research board, American Association of State Highway and Transportation Officials, Washington, DC*, 2006.

[130] Stephen Peter Robinson. *The development and application of an urban link travel time model using data derived from inductive loop detectors*. PhD thesis, University of London, 2006.

[131] Nikolaos Zygouras, Nikolaos Panagiotou, Nikos Zacheilas, Ioannis Boutsis, Vana Kalogeraki, Ioannis Katakis, and Dimitrios Gunopulos. Towards detection of faulty traffic sensors in real-time. In *MUD@ ICML*, pages 53–62, 2015.

[132] Shunsuke Kamijo, Yasuyuki Matsushita, Katsushi Ikeuchi, and Masao Sakauchi. Traffic monitoring and accident detection at intersections. *IEEE transactions on Intelligent transportation systems*, 1(2):108–118, 2000.

[133] Harini Veeraraghavan, Paul Schrater, and Nikolaos Papanikolopoulos. Switching kalman filter-based approach for tracking and event detection at traffic intersections. In *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, pages 1167–1172. IEEE, 2005.

[134] Shiming Yang, Konstantinos Kalpakis, and Alain Biem. Detecting road traffic events by coupling multiple timeseries with a nonparametric bayesian method. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):1936–1946, 2014.

[135] Xiangjie Kong, Ximeng Song, Feng Xia, Haochen Guo, Jinzhong Wang, and Amr Tolba. Lotad: long-term traffic anomaly detection based on crowdsourced bus trajectory data. *World Wide Web*, pages 1–23, 2017.

[136] Peter Widhalm, Hannes Koller, and Wolfgang Ponweiser. Identifying faulty traffic detectors with floating car data. In *Integrated and Sustainable Transportation System (FISTS), 2011 IEEE Forum on*, pages 103–108. IEEE, 2011.

[137] Fernando Terroso-Sáenz, Mercedes Valdés-Vela, Cristina Sotomayor-Martínez, Rafael Toledo-Moreo, and Antonio F Gómez-Skarmeta. A cooperative approach to traffic congestion detection with complex event processing and vanet. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):914–929, 2012.

[138] Hongtao Wang, Hui Wen, Feng Yi, Hongsong Zhu, and Limin Sun. Road traffic anomaly detection via collaborative path inference from gps snippets. *Sensors*, 17(3):550, 2017.

[139] Xiaolei Ma, Zhuang Dai, Zhengbing He, Jihui Ma, Yong Wang, and Yunpeng Wang. Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. *Sensors*, 17(4):818, 2017.

[140] OpenStreetMap Wiki. Tmc/location code list/location types, 2017. [Online; accessed 12-September-2017].

[141] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[142] MathWorks. Detector performance analysis using roc curves, 2017. [Online; accessed 30-September-2017].

[143] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[144] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[145] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012.

[146] Forrest N Iandola, Matthew W Moskewicz, Khalid Ashraf, and Kurt Keutzer. Firecaffe: near-linear acceleration of deep neural network training on compute clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2592–2600, 2016.

[147] Jeff Dean. Large scale deep learning. In *Keynote GPU Technical Conference*, volume 3, page 2015, 2015.

[148] Surat Teerapittayanon, Bradley McDanel, and HT Kung. Distributed deep neural networks over the cloud, the edge and end devices. In *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, pages 328–339. IEEE, 2017.

[149] Surat Teerapittayanon, Bradley McDanel, and HT Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 2464–2469. IEEE, 2016.