IoT-Driven Scheduling of Residential HVAC and Virtual Bus Lanes for Energy Savings

by

Daniel Petrov

B.S., Sofia University "St Kliment Ohridski", 2008

M.S., Sofia University "St Kliment Ohridski", 2010

M.S., University of Pittsburgh, 2019

Submitted to the Graduate Faculty of

the Department of Computer Science in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2021

UNIVERSITY OF PITTSBURGH

SCHOOL OF COMPUTING AND INFORMATION

This dissertation was presented

by

Daniel Petrov

It was defended on

April 29th 2021

and approved by

Panos K. Chrysanthis, Department of Computer Science

Daniel Mossé, Department of Computer Science

Alexandros Labrinidis, Department of Computer Science

Vladimir I. Zadorozhny, Department of Information Systems

Dissertation Advisors: Panos K. Chrysanthis, Department of Computer Science,

Daniel Mossé, Department of Computer Science

IoT-Driven Scheduling of
Residential HVAC and Virtual Bus Lanes
for Energy Savings

Daniel Petrov, PhD

University of Pittsburgh, 2021

The availability of commodity Internet connection and the decrease in price and form factor of consumer electronics led to the emergence of Internet of Things (IoT), with which our world becomes more connected and instrumented. IoT is a great vehicle for enabling solutions to problems in the connected environment that surrounds us (i.e., smart homes and smart cities). An example is the use of sensors and IoT to address issues related to energy efficiency, the broad area of this dissertation.

Our hypothesis is that data processing and decision making need to be carried out at the network edge, specifically as close to the physical system as possible, where data are generated and used, to produce results in real-time and make sure the data is not exposed to privacy and security risks. To this end, we propose to leverage scheduling principles and statistical techniques in the context of two applications, namely aiming to reduce duty cycle of HVAC (Heating, Ventilation, and Air Conditioning) systems in smart homes and to mitigate road congestion in smart cities. The common goal in these two aims is the reduction of energy consumption and the reduction of atmospheric pollution.

To achieve our first aim we propose intelligent scheduling of the duty cycles of HVAC systems in residential buildings. Our solution combines linear and polynomial regression enabled estimator that drives the calculations about the amounts of time thermally conditioned air should be supplied to each room. The output from our estimator is fed into our scheduler based on integer linear programming to decrease the duty cycle of the home's HVAC systems. We evaluate the effectiveness and efficiency of our HVAC solution with a dataset collected from several residential houses in the state of Pennsylvania.

To achieve the second aim, we propose the concept of virtual bus lanes, that combines on-demand creation of bus lanes with dynamic control of traffic lights. Moreover, we propose

to guide drivers through less congested routes using light boards that provide to drivers information in real-time for such routes. Our methods are anchored to priority scheduling, incremental windowed-based aggregation, and shortest path first Dijkstra's algorithm. We evaluate the effectiveness and efficiency of our virtual bus lanes solution with a real dataset from the city of Beijing, China, and a synthetic traffic scenario from the city of Luxembourg.

Table of Contents

List of Tables

List of Figures

Preface

First and foremost I would like to thank my advisor Panos K. Chrysanthis. It has been a great pleasure to be his student. He guided me through the sometimes convoluted and challenging journey of becoming a scientist. His invaluable advises and the lessons I have learned from him will be amongst the things I will carry with me from Pittsburgh. I would also like to thank my co-advisor Daniel Mossé. He has always been the sounding board that I needed when I got stuck. His comments have always helped me to make the next leap. I am very grateful for having Alex Labrinidis on my committee. His dedication to perfect the presentation of my work, both in written and when presented in front of an audience, made a significant difference during my studies. The insightful thoughts and the hints of Vladimir Zadorozhny were critical for the completion of this work.

I would also like to thank my collaborator and co-author of several of my publications - Rakan Alseghayer. Rakan, thank you for all of the fruitful and passioned discussions in front of the whiteboard, all weekends spent in the laboratory brainstorming, debugging code, and running experiments, and the sleepless nights writing manuscripts during this adventure. I would also like to thank the other members and affiliates of the Advanced Data Management Technologies laboratory for their valuable feedback - my friends Nikos Katsipoulakis, Anatoli Shein, Cory Thoma, Constantinos Costa, Judicael Briand Djoko, Hany Saleeb, Salim Malakouti, Xiaoyu Ge, Xiaozhong Zhang, Evangelos Karageorgos, Vineet Raghu, Amanda Crawford, Ekaterina Dimitrova, Xiaoyu Liang, Longhao Li, Xiaolong Cui, Debarun Das, Qun Yu, and Xerandy.

I am very grateful to all of the other students of the Department of Computer Science at PITT, who I shared the experience with - (in no particular order) Kenrick Fernandez, Mohammad Mofrad, Alireza Samadian, Mahbaneh Eshaghzadeh, Zuha Agha, Constance Clive, Wenchen Wang, Fan Zhang, Mark Silvis, Antony Sicilia, Zhipeng Luo, Henrique Potter, Adriano Maron, Chris Thomas, and Yuyu Zhou.

The Ph.D. program made me meet a lot of great people in Pittsburgh from all around the world. It is my pleasure and honor to be your friend Lory Al Moakar, Peter Djalaliev, Molly

Latinova, Jak Latinov, Adriana Kovashka, Logan Gordon, Deema Abdalah, John Feghali, Roxana Neophytou, Panickos Neophytou, Maria Tomprou, Cargi Cinkilic, Stefano Gridelli, Gemma Wallnau, Amrit Pandey, Deirdre Morris, Thanasis Karamalidis, Toula Protopapa, Rakan Maddah, Christine Shein, Zheru Liu, Tim James, Anthonis Tasoglu, Katia Liagaki, Elena Markoska, Stephanie Nicolaas, Melissa Hem, Dimitra Agapitou, Christos Kaltso, Sadaf Tayefeh, Anna Thoma, Shadi Sanoubar, Annia Maisa, Crysovalantis Anastasiou and George Constantinou.

To my friends, who shared the happiest and the most difficult moments in my doctorate, regardless of the distance and the time difference between us: This journey would not have been the same without you! Thank you, Bozhidar Georgiev, Liliya Lozanova, Dimo Statkov, Mira Menkova, Eli Stefanova, Borislav Stefanov, Gergana Boyadzhieva, Viktor Traykov, Irena Simeonova, Krasimir Simeonov, Janet Dimitrova, Filip Dimitrov, Jordan Todorov, Nevena Tacheva, Iva Popova, Krassimir Tzvetanov, Snejana Georgieva, Konstantin Nikolov, Mariya Zheleva and Petko Bogdanov.

I dedicate this work to my amazing family - my mother Mariana, my father Peter, my super brother Krasimir, my sister-in-law Yoana, my nephew Christian and my niece Ema. You have always been there for me, you have been my motivation and inspiration, my greatest supporter, and my most honest critic. You planted my interest in science, stimulated and supported my continuous development throughout the years and you have always believed in me. Thank you!

Dear Reader,

I hope that You will enjoy the rest of the read!

1.0 Introduction

1.1 Motivation

Internet of Things (IoT) is an emerging field in academia and business. Many factors such as the decrease in the form factor of computer components (i.e., CPUs, memory, battery), the development of system-on-chip general-purpose computers, and the decrease in the price of those computers played major role for the commodity spread of IoT. Furthermore, IoT became a vehicle for enabling a world where the objects around us are connected and communicate with each other–phones, cars, houses, refrigerators, ovens, light bulbs, and others [64, 65, 74, 23]. Instrumented houses whereby a hub computer is controlling and facilitating the communication of objects with one another as well as the communication between objects and humans became known as *smart homes*. Similarly, instrumented vehicles, road infrastructure, and city infrastructure (i.e., street lamps, information boards, sprinklers) are referred to as *smart cities*.

There is an increasing trend in the media outlets sharing concerns about the pollution, resulting from human activities. A significant proportion of the pollution is attributed to transportation and space conditioning. Parallel to this, the Internet of Things is making its way into becoming a disruptive technology that leads to leaps in many aspects of our life. Today our home appliances are smart, our cars get "over air updates" of their software, many industries used IoT as a springboard to optimize their production processes. Additionally, the ubiquitous cloud computing enabled possibilities for carrying out computations in environments that are not resource-constrained, theoretically. The latter gave rise to an increase of privacy and security concerns amongst users. This dissertation addresses the challenges related to reducing pollution, caused by space conditioning in residential buildings (for the rest of this dissertation *residential building, house*, and *residential house* will be used interchangeably unless otherwise specified) and public transportation buses, equipped with internal combustion engines.

Problem 1: The energy consumption of private houses, commercial, and public buildings

has been increasing. The biggest amount of energy consumed in the US is for space heating and cooling of residential buildings–47.7% in 2009 [48]. This induces higher costs and calls for enabling new power plants that harm the environment [47]. To mitigate the negative impact on the environment it is imperative to optimize the energy consumption in residential buildings.

Smart homes are instrumented with sensors and actuators to support fine-grain temperature control, among other advancements (i.e., lighting control, blinds, and shades control). Predominantly residential buildings are equipped with a single ducting system and a single HVAC system (i.e., furnace and air conditioning unit). The temperature control in such buildings drives the duty cycle of their HVAC systems. Often the length of the duty cycle increases due to the nature of people's habits to change the desired temperature in the room when they feel uncomfortable (i.e., cold or hot). The increased length of duty cycles of HVAC systems increases the amounts of greenhouse gases, fine particulate matter (FPM), and pollution emitted in the atmosphere. This observation motivated the first aim of this dissertation, which is to *effectively control the lengths of duty cycles of HVAC systems in smart homes*, therefore resulting in less pollution while maintaining the inhabitants' comfort.

Problem 2: The greenhouse gas emissions produced by vehicles increased by 21.5% between 1990 and 2016 in the US (Figure 1) [77]. The traffic jams in many large cities increase despite the efforts of the local authorities to build additional infrastructure to mitigate road congestion. Traffic jams have a significant negative impact on humans' physical and mental health–people get stressed in traffic jams, and the idling engines of the vehicles emit greenhouse gases and FPM in the atmosphere. This observation motivated the second aim of this dissertation, which is to *decrease the idling time of internal combustion engines of vehicles (ICEVs) and promote the usage of public transportation* by exploiting the infrastructure of smart cities, therefore reducing pollution.

The common goal of the two aims of this dissertation is to reduce the generation of greenhouse gases. We propose to achieve these two aims by developing effective models and efficient methods for next-generation IoT systems, which exhibit low latency, low operational cost and protect data privacy. Moreover, our solutions are expected to be effective and to meet the following additional requirements:

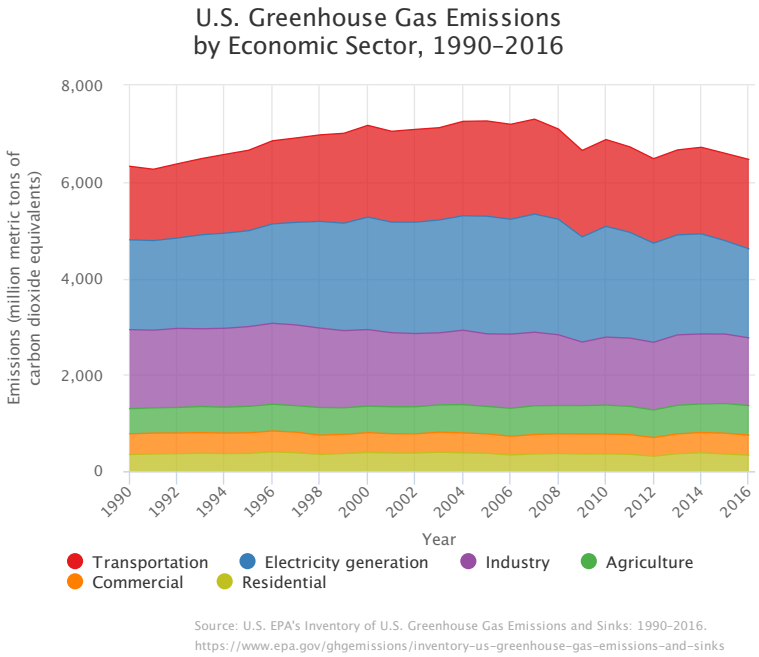

Figure 1: US Greenhouse Emissions by Economic Sector, 1990-2016

(1) towards our first aim for effectively controlling the lengths of duty cycles of HVAC systems in smart homes, the proposed system should be capable of processing vast volumes of data in short periods of time, as it will be deployed into large residential buildings and it should accurately estimate the amounts of thermally conditioned air that is needed for each room to satisfy the requirements of the occupants; and

(2) towards our second aim to decrease the idling time of internal combustion engines of vehicles and promote the usage of public transportation in smart cities, the proposed solution should be capable of processing vast volumes of data in short periods of time as it will be deployed into large cities and it should ensure mass transit vehicles are not slowed down by traffic jams, and real-time guidance is provided to drivers about the less congested route to their destination.

1.2 Current Approaches

Several approaches have been proposed that address our aims. To put our proposed solutions in context, we discuss the limitations of these existing approaches below.

Problem 1 (Control HVAC Duty Cycle) A room-level zoning of HVAC systems has been proposed in [83]. The crux of that work is a 3-dimension study that focuses on HVAC dimensioning, occupancy prediction, and leaks in the ducts and the dampers. Energy Plus framework is used to simulate heating in a residential building in the study about HVAC dimensioning. This work relies on an idealized model that does not model thermal energy leakage out of the building. This is not the case in reality.

The work has been extended further into [82], whereby a novel accurate room occupancy detection mechanism has been proposed. It helps saving energy by avoiding the thermal conditioning of rooms that are not occupied, but it does not address the following shortcoming of fine-grain temperature control on a per room basis—it often increases the duty cycle of the HVAC system due to the humans' habits. Specifically, people get to the thermostat to change the temperature in the room they occupy when they feel uncomfortable. Moreover, the vents in smart homes can be opened and closed remotely. It is not uncommon for people, who feel comfortable, to close the vents in their rooms to mitigate the possibilities for changing the temperatures from the current ones in those rooms. Thereby, the furnace is often started with the goal of thermal conditioning of a single room only. Furthermore, another occupant may need to change the temperature in their room soon after the furnace supplied enough thermally conditioned air to reach the temperature desired by a particular occupant and went off. This real-life scenario presents a very inefficient way of scheduling the thermal conditioning in a smart home.

Furthermore, a plethora of algorithms has been proposed for room temperature estimation, linear, polynomial and logistic regressions, neural networks–to name a few [49, 36]. The ideal machine learning (ML) technique to be used in this context should not be computationally expensive so that the computations can be carried out at the edge of the network where data is generated and consumed. This ideal ML technique should tolerate varying times between two consecutive measurements and it should provide accurate results [40, 91, 63].

Problem 2 (Control Mass Transit Buses Delay and City Road Network Congestion) Rapid proliferation of smart mobile devices that are equipped with positioning sensors (e.g., GPS and Galileo), and ubiquitous Internet connectivity facilitated the growth of near real-time traffic analysis [71]. Some cities aimed to reduce pollution by implementing smart traffic lights that adaptively steer the traffic to mitigate congestions–left turns are forbidden during rush hours in some areas of Washington, D.C., and mid-lane direction is switched on some bridges in Pittsburgh. While forbidding left turns eliminates the presence of cars that are waiting on the opposite traffic to go to make their turns, this solution increases the length of the route traveled. Moreover, there is no limit on how much the traveled distance is increased. Changing the direction of lanes on-demand brings a different challenge. Specifically, the traffic flow gets increased on part of the road network, but this causes congestion in the following road segments that have a fixed number of lanes per direction. This phenomenon is caused by the fact that the throughput of a given road is equal to the throughput of its most narrow segment.

A slightly different approach is taken in Pittsburgh and Los Angeles—they installed cameras at intersections across the city (500 cameras in the city of Los Angeles). The images from them are fed to a system that counts the cars and adaptively adjusts the timing of the green light cycles of the traffic signals [68, 81]. Moreover, the system has additional functionality whereby human operators can overrule the system and adjust the green light signals as they like. To this end, dynamic adjustment of green cycles of traffic lights has merit, but having a small number of operators to control the traffic lights at five hundred intersections is prone to error—they can easily cause starvation to some streets over others. Additionally, this solution does not address the idling internal combustion engines that emit pollution.

The City of San Jose collaborates with private companies in performing a case study that aims to help the drivers to adapt their speed to the green light cycles of the traffic lights on their route [80]. Specifically, the cellular phones of the drivers are running an application that sends the location of their cars and their velocity to a system in the cloud. The system returns to the phone information about the traffic lights on the next intersection along the way of the vehicle. The system suggests speed adjustment to drivers that aims to

minimize the waiting time for drivers before they can cross the intersection. This solution is computationally intensive and exposes sensitive information to privacy and security risks. The information about each driver/car is processed in the cloud. Moreover, the transmission of data to the cloud induces delay that may be critical in reality due to the induced delay, i.e., the proposed adjustment of speed may be impossible.

An ideal solution will terminate the traffic jams once and for all. Some studies show that the additional infrastructure built does not solve the problem of traffic jams. It only attracts new traffic and changes the scale of the problem [61, 25, 94]. The latter phenomenon, also known as "induced demand", is worth exploring because the direct effect of traffic jams leads to an increase in the amount of fuel spent to transport one person per mile. This suggests a balanced solution, that promotes the use of public transportation while reducing air pollution due to buses idling, and decreasing the number of cars driving on the streets.

To achieve our aims, we have to circumvent the aforementioned technical challenges to support large-scale complex pollution reduction and environmental applications.

1.3 Objective, Approach & Novelty

The premise of this dissertation is that energy savings and pollution reduction are achievable by developing effective models and efficient methods for IoT systems, which exhibit low latency, low operational cost and protect data privacy. Our hypothesis is that data processing and decision making needs to be carried out at the network edge as much as possible, where data are generated and used, to produce results in real time and make sure the data is not exposed to privacy and security risks. The objective of this dissertation is to develop a set of novel interactive approaches and implement corresponding prototype platforms that embody our hypothesis and address the aforementioned limitation of existing approaches in addressing the problems of "Control of HVAC duty cycle", and "Control of mass transit buses delay and city road network congestion". The novelty of our approaches is anchored to leveraging scheduling, statistical techniques, window-based aggregation, and indexing techniques to support practical solutions for smart cities and smart homes.

Aim 1 To achieve our smart homes aim, we propose intelligent scheduling of the duty cycles of HVAC systems in residential buildings. Our solution employs a statistical technique as an estimator that drives the calculations about the amounts of time thermally conditioned air should be supplied to each room to reach the temperature, desired by the inhabitants of the room. The output from our estimator, as well as the requirements of the guardian, which provides quality of services (QoS) guarantees, are fed into our scheduler that uses integer learning programming to decrease the duty cycles of HVAC systems.

In addition to providing estimates, the selected statistical technique should be computationally cheap to be run on cost-efficient commodity computers with modest capabilities. Additionally, the scheduler does not require significant computational capabilities either. This enables the proposed solution to be executed at the edge of the network where data is produced. Another advantage of avoiding data transmission and processing in the cloud eliminates the exposure of data to privacy and security risks.

Our work using linear regression as an estimator has great performance when run on Raspberry Pi for a relatively large residential building of 32 rooms. The estimator is run, the data is fed into the scheduler and a schedule is generated, and results are produced in a sub-second time frame. We experimented with additional statistical techniques and other approaches to estimate thermal energy exchange. The results showed that linear regression outperforms them with respect to accuracy for most of the cases. In the rest of the cases, it produces comparable results but does not require fine-tuning, unlike the other techniques.

Our solution, called ILPSS, takes the desired temperature along with the latest time by which the user expects the temperature to be reached (which we call *a deadline*). Its innovation is that ILPSS combines scheduling and regression techniques. The former optimizes HVAC duty cycles and the latter estimates the time needed to reach the desired temperature for each request. Moreover, we added guarantees that the desired temperature will be reached and maintained by the *deadline*. That is achieved by inducing additional requirements to the HVAC duty cycle schedule.

Aim 2 To achieve our smart cities aim, we propose the concept of virtual bus lanes, that combines on-demand creation of bus lanes with dynamic control of traffic lights. By creating bus lanes on demand, we eliminate the slowdown of buses and having them stuck in

traffic jams. Our approach, called *Environment Protective Traffic Orchestration* (EPTrOn), implements a monitoring system that collects information about the traffic. The monitoring system identifies the congested road segments and the current location of the buses. This information is fed to schedulers at each intersection that dynamically adjusts the green light cycles of traffic lights to facilitate the generation of virtual bus lanes. Moreover, guidance information for less congested routes to given destinations is calculated at each intersection. The information is provided to drivers in an effort to speed up their travels.

The developed technique for congestion identification should provide accurate results with respect to both the average speed of the vehicles on each road segment and the number of vehicles per road segment. It should not expose sensitive information about the drivers and the vehicles to privacy and security risks. Furthermore, the scheduling of traffic lights should be computationally cheap in order to be run at the edge network. Similarly, the guidance information should be bounded in order to achieve the feasibility of generating it within seconds.

We employ priority scheduling techniques to drive the implicit creation of "green waves" for traffic. Furthermore, we use R^+ *tree* spatial indexing and window-based aggregation techniques to assign vehicles to streets and determine traffic jams. Dijkstra's shortest path algorithm is used to calculate less congested routes.

1.4 Contributions

In this dissertation, our objective is to provide solutions that get us towards our two aims. Specifically, we make the following two contributions:

(1) Practical HVAC Scheduling: We present an IoT solution that schedules the duty cycles of HVAC systems in residential buildings intelligently. It reduces the energy consumption for space conditioning while meeting users' comfort requirements for target temperature, which are guaranteed by what we call *guardian*. The solution works on a per-room basis. Our experimental evaluation with real data showed that our approach achieves energy savings up to 49% (26% on average), compared to the baseline commodity HVAC. Fur-

thermore, we demonstrated that our computationally cheap solution can be deployed on low-cost commodity hardware, such as Raspberry Pi Zero, and it is capable of addressing the demands for HVAC control of real-world residential buildings. Moreover, our solution is based on the IoT computational framework that we also propose. The framework consists of three building blocks—*estimator*, *guardian*, and *scheduler*.

(2) On-demand Dynamic Bus Lanes Creation: We present a proactive solution that ameliorates the traffic ahead of public buses in congested areas, called *Environment Protective Traffic Orchestration* (EPTrOn). EPTrOn mitigates congestion by establishing virtual bus lanes and shaping traffic by controlling traffic lights and directing traffic using light boards at intersections. Moreover, our EPTrOn solution pushes ICEV vehicles away from congested streets, where their engines will be idling for a prolonged amount of time. EPTrOn shortens the time lost in traffic by buses three times, on average. Moreover, the penalty induced on cars in order to create the virtual bus lanes does not go beyond 10 seconds on average (less than 1% on average).

We evaluate the effectiveness and efficiency of our solution on real datasets. Our smart home solution is evaluated with two datasets collected from residential houses in Pennsylvania. Our smart city solution is evaluated with two different datasets–one contains data for taxis in Beijing, China, and the second is a synthetic traffic scenario in the city of Luxembourg.

1.5 Dissertation Outline

This dissertation is structured thematically along the lines of the two aims. Specifically Chapters 2 + 3 cover the background and the computation framework, and experimental evaluation towards aim 1. Chapters 4 + 5 cover the background and the computation framework, and experimental evaluation towards aim 2, respectively. It is to be noted, however, that we discuss the related work, relative to each contribution in the respective chapter - Chapter 2 for HVAC scheduling and Chapter 4 for dynamic traffic orchestration. Finally, in Chapter 6 we provide conclusions and purposed future work.

2.0 Residential HVAC: Computational Framework

In this chapter, we present *ILPSS (Integer Linear Programming for Smart Scheduling)*, a novel solution for optimizing the duty cycle of the HVAC (Heating, Ventilation and Air Conditioning) equipment and improving users' comfort by allowing users to specify comfort levels at specific times in each room of a residence. Our proposal builds on multiple variable, linear regression model, and integer linear programming and can be run on a home IoT (Internet of Things) hub. The IoT hub is a computer controlling and facilitating the communication of home devices.

In Section 2.1 we present the background of our solution. The system model is discussed in Section 2.2, followed by the regression models for our time-temperature estimator in Section 2.3. Our ILPSS solution is presented in Section 2.4. Section 2.5 discusses the comparison of our solution to the state-of-the-art. We summarize our findings in this chapter in Section 2.6. The results of the experimental evaluation are pretended in Chapter 3.

2.1 Problem Statement and Solution Overview

Many buildings in the US are equipped with thermostats that control the temperature. In commercial buildings often each room has a thermostat, while in residential houses one thermostat controls the temperature on one floor or in the whole house. Furthermore, sensors to detect the presence/absence of humans are installed in more and more buildings. Such "smart" buildings have systems in place to control the lighting as well. Recently, buildings are built with sensors and actuators that allow fine-grain control of the temperature at the room level and the duty cycles of the lighting. Moreover, the emergence of the Internet of Things enabled new technologies and it also facilitated increased autonomy in space conditioning and lighting—smartphone-based geo-fencing, as well as connected thermostats, power plugs, and light bulbs aim to improve quality of life [64, 65, 74, 23].

Problem Statement Given a smart building—that has IoT infrastructure installed and is enabled for fine-grain control of the temperature, the system can open or close the vents "on-demand". Further, the (smart) thermostat will command the HVAC unit to turn on at the appropriate time to regulate the temperature in the room for which temperature adjustment is needed. Some commercially available smart thermostats offer geofencing and static thermo-temporal programs to address the need for reaching the temperature, desired by the user at a time, specified by the user. The usage of geofencing raises a number of privacy and security concerns. Specifically, the distance between the users and the thermostat is calculated in the cloud, using their geolocation. Some individuals may oppose having their location information shared with an application, as they do not have control over whether the application sends their location to the cloud or not [26]. Moreover, both approaches (i.e., the usage of geofencing and statically configured programs) fail to address the expectation of the user for having their desired temperature at the time they need it. Consequently, the on-per-room-basis temperature control may lead to multiple starts and stops of the HVAC (i.e., HVAC short cycling). This increases the energy consumption and the pollution produced by the HVAC system [69]. In this dissertation, we address the problem of maximizing comfort (i.e., having the right temperature at the right time, as requested by the users). Moreover, we aim at decreasing the energy consumption and the HVAC short cycling, while maximizing comfort. Additionally, the necessary computations should not expose users' personal data to any privacy and security concerns.

Approach In this chapter, we propose an IoT computational framework, which consists of an *estimator*, *guardian*, and *scheduler*. The estimator models the context in which the temperature in the room changes and provides estimates for changes in the temperature in the future. The guardian provides quality of service (QoS) guarantees, for example, the desired temperature will be reached by the deadline, provided by the user. The scheduler gets input from the estimator and the guardian, as well as the requests of the users, and generates a schedule for the HVAC that enables energy savings. Our solution leverages this IoT computational framework. The latter is depicted in Figure 2.

This IoT computational framework can lead to different solutions, with estimators, different QoS, and different schedulers. We proposed a two-dimensional *comfort zone* model

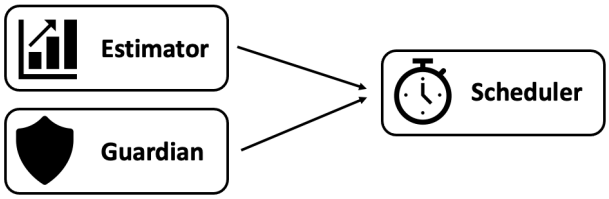

Figure 2: IoT Computational Framework

of *guardian*, namely a *temperature-temporal model*, that is combined with *ILPSS* time-temperature estimation model to optimize HVAC duty cycles using integer linear programming (ILP). Our *ILPSS* solution is depicted in Figure 4 and it is discussed in details in Section 2.4). Given the comfort zones requirements of users, our proposed *ILPSS* IoT-based solution schedules the duty cycles of HVAC systems intelligently for energy reduction. Moreover, the requirements are to achieve target temperature within a user-defined interval of time, on a per-room basis, and the HVAC systems are installed in residential buildings. The time-temperature estimation model is based on multiple linear regression (MLR) (this is one of the *estimators* that we evaluated; the other one is multiple polynomial regression (MPR)) that estimates the time needed to reach the desired temperature for each request. The time estimation is calculated using sensor readings from each room.

ILPSS is a lightweight computational solution that can run on a "smart" gateway in a real-life IoT "hub" and keep the computations local in the hub, avoiding exposure of users' data to privacy and security concerns.

All requests and data sensor readings from all rooms are delivered to the smart gateway (defined in the next section), which uses them to prepare a schedule that controls the duty cycle of the HVAC system to minimize energy while maintaining users' comfort.

Furthermore, our solution is oblivious to the underlying networking infrastructure. The information it operates on is fed into the gateway from the sensor readings and user requests. The gateway may be connected to wired, wireless, or ad-hoc networks in order to receive data from sensors and users. Often these gateways are deployed on Raspberry Pi computers. We deployed successfully our ILPSS on Raspberry Pi Zero (as discussed in the next chapter).

2.2 System Model

In this section, we discuss our system model of residential buildings. The notation that we have adopted for the rest of this chapter is summarized in Table 1.

A room has direct space conditioning capabilities if there is a vent installed in the room, and that vent is connected to the controller that is in charge of space conditioning (of part of) the building. Each such room is equipped with a self-contained sensing unit whose measurements are denoted $\{x_{ij}\}$, whereby i denotes the sample number and j denotes the sensor that generated the measurement. It is to be noted, however, that the sensors may not be combined in a single sensing unit. We combined them for practical reasons—it is easier to install, maintain and operate a single unit, rather than individual sensors [59].

Definition 1. *(Window of measurements) A window w is a vector of n consecutive measurements of the sensors ordered in time. The oldest measurement in the window is at time $t - w$ and the most recent one is at time t. Each measurement contains the values from all available sensors that are fed into the thermal energy exchange function (discussed in Section 2.4) used in the multiple linear/polynomial regression.*

Definition 2. *(Request)* request u_i *is a tuple*

$$u_i(i, t_c, \theta_{i,target}, t_{i,target}) \tag{2.1}$$

whereby t_c is the timestamp that marks when the request was generated in room i, $\theta_{i,target}$ is the desired temperature to be achieved for this room, $t_{i,target}$ is a moment in the future by which the desired temperatures should be reached (the "deadline"*).*

The requests should be received enough time in advance in order to achieve the feasibility of satisfying them. For example, if a user submits a request for a change of the temperature in the room by 20 degrees and $ts_{i,target}$ is 5 seconds from now, the request is infeasible.

Definition 3. *(*Objective criterion*) Given the set Q_{curr} of current sensor readings for all rooms i, $1 \leq i \leq m$, the previous window w's sets of sensor readings $Q_{curr-1}, Q_{curr-2}, ...,$ Q_{curr-w}, and the set R of requests u_i for each room i in a house, generate a schedule S that*

Table 1: Notation for HVAC Scheduling

$x_{i,j}$	sensor reading
p	number of sensors
i, j	running counter
w	window length
u	user request
R	set of requests
t_c	current time
θ	temperature
Q_c	a set of sensor readings at time c
m	number of rooms
f_{ee}	thermal energy exchange function
g	temperature change function
δ_t	period of time
f	user-defined function
cz	comfort zone
b, c	vector of values
A	matrix
y	binary variable
k	coefficient
C	constant
st	decision variable
$pen,\ penPlus,\ penMinus$	"penalty" variables

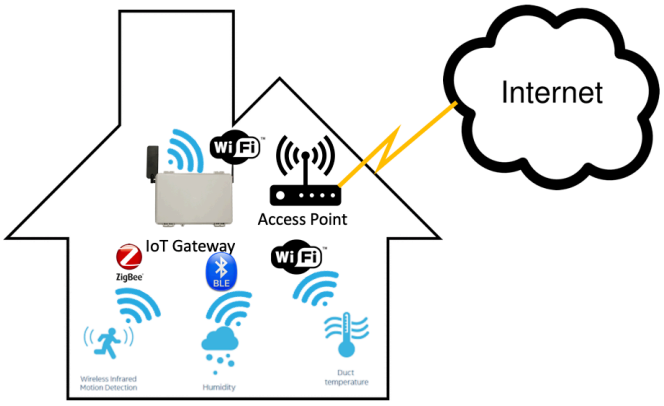

Figure 3: System Model - gateway and sensor connectivity

minimizes the duty cycle of the furnace/AC and achieves the requested target temperatures $\theta_{i,target}$ *by the users' deadlines* $t_{i,target}$.

The schedule should be produced in real-time and it should meet all deadlines. Moreover, we assume the furnace works at a single stage.

The residential building has an access point (AP). The AP has wired connectivity to the Internet. Moreover, it provides connectivity to the IoT gateway. The latter provides different protocols for wireless connectivity to the sensors, namely Bluetooth low energy (BLE), Zigbee, and Wi-Fi, to name a few. The gateway works in conjunction with the AP and provides means for users to submit their requests for space conditioning. Furthermore, our system also runs on the gateway.

2.3 Regression Models

In this section, we present the two statistical techniques that are used in our thermo-temporal estimator with our thermal energy exchange function, namely *Multiple Linear Regression* (MLR) and *Multiple Polynomial Regression* (MPR).

2.3.1 Multiple Linear Regression

Both MLR and MPR are used in our solution in a *sliding window* model, whereby n consecutive sensor readings for each of the p sensors for each room $(x_{11}, x_{12}, ..., x_{ij}, ..., x_{np})$ for all of the rooms in the building and run regression on them to derive the coefficients β_i of $g(x_1, .., x_p, \delta_t)$ of the regression model. When new sensor readings are available, we drop the oldest readings and add the newest, ordered by time. An example is introduced in the next section of this chapter.

Multiple linear regression (MLR) models the relationship between two or more explanatory variables x_i, δ_t, and a response variable g by fitting a linear equation to observed data (recall that the individual readings for sensor x_i are denoted $x_{i,j}$). For example, in our problem, $x_{i,j}$ are the sensor readings for a room and g is the function that can be used to calculate the change in temperature in that room. For simplicity, we will discuss the temperature change function for one room only. It is to be noted, however, that each room has its temperature change function. Every value of the independent variable x_i is associated with a value of the dependent variable g. Furthermore, multiple observations for each of the p explanatory variables, as well as for the dependent variable are necessary for calculating the coefficients of the relationship. The population regression line for p explanatory variables $x_1, x_2, ..., x_p$ is defined as

$$\mu_g = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p \tag{2.2}$$

This line describes how the mean response μ_g changes with the explanatory variables. The observed values for g vary about their means μ_g and are assumed to have the same standard deviation σ. The fitted values $b_0, b_1, ..., b_p$ estimate the parameters $\beta_0, \beta_1, ..., \beta_p$ of the population regression line. We will get back to them later.

Since the observed values for g vary about their means μ_g, the multiple linear regression model includes a term for this variation. Mathematically, the model is expressed as

$$data = fit + \varepsilon \tag{2.3}$$

where "fit" represents the following expression from Eq. 2.2.

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p \tag{2.4}$$

The ε is also called the "residual" term and it represents the deviations of the observed values of g from their means μ_g, which are normally distributed with mean 0 and variance σ.

Definition 4. *(MLR) The model for multiple linear regression, given n observations, is*

$$g_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \varepsilon_i \tag{2.5}$$

for $i = 1, 2, ..., m$. In the least-squares model, the best-fitting line for the observed data is calculated by minimizing the sum of the squares of the deviations from each data point to the line (if a point lies on the fitted line exactly, then its deviation is 0). Because the deviations are first squared, then summed, there are no cancellations between positive and negative values.

The values calculated from the expression $b_0 + b_1 x_{i1} + b_2 x_{i2} + ... + b_p x_{ip}$ are denoted $\hat{g}_i$ and the residuals ε_i are equal to $g_i - \hat{g}_i$, the difference between the observed and fitted values. The sum of the residuals is ideally equal to zero.

The variance σ^2 may be estimated by

$$\sigma^2 = \frac{\Sigma \varepsilon_i^2}{n - p - 1} \tag{2.6}$$

also known as the mean-squared error (or MSE) [12, 38, 32, 60].

2.3.2 Multiple Polynomial Regression

In this subsection, we discuss the other estimating model, namely the Multiple Polynomial Regression (MPR), that we use with our thermal energy exchange function.

Recall from the previous subsection that Equation 2.2 is the mathematical model of linear regression. Often the linear model does not fit closely the values of the dependent variable g, i.e., the model deviation is significant. In such cases a model of higher degree is a better fit:

$$\beta_0 + \beta_1 x_1 + \beta_{12} x_1 x_2 + \beta_2 x_2 + ... + \beta_p x_p \tag{2.7}$$

The polynomial from Equation 2.7 is also *hierarchical* (a definition is provided later in the section).

MPR is a special case of multiple linear regression, whereby the independent variables x_i are of order $k > 1$. Moreover, we formalize the problem as:

$$g = X\beta + \varepsilon \tag{2.8}$$

where g is a vector of observations of the dependent variable, X is a matrix of observations of the explanatory variables (i.e., p explanatory variables and n observations), β is a vector $p + 1$ coefficients (i.e., one for each explanatory variable, as well as the intercept) and ε is a vector of the residual. When $k = 2$, for at least one variable x_l, the model is called *second-order* or also *quadratic* model. The coefficients β_i of the variables x_i of degree one are called *linear effect parameters* and the coefficient β_l of the variable x_l is called *quadratic effect parameter*. Often the order of the polynomial model is kept as low as possible. In case a linear model is not a good fit, the second-order model is tried. It is known that a polynomial of degree $(n - 1)$ can fit through n points. There is a tradeoff because finding such polynomial induces significant computational costs, but it also helps to understand better the underlying function. In other words, a polynomial of higher degree fits the data points with smaller error/residual. Furthermore, the polynomial models should be maintained *hierarchical*, because only *hierarchical polynomials* are *invariant* under linear transformations.

Definition 5. *(Hierarchical polynomial) A polynomial of order k is called hierarchical if it contains terms of all degrees of the independent variables from the highest to* 0*.*

Two different strategies are available for finding the "best" polynomial model, namely *forward selection procedure* and *backward selection procedure*. The *forward selection procedure* starts with building a simple model of degree one. The model keeps getting enhanced with monomials (terms) of higher degree until the deviation ε is satisfactorily small. The *backward selection procedure* starts with a polynomial model of the highest degree possible (i.e., $n-1$ for n observations). The monomials from the highest degrees kept being removed one by one. The procedure concludes when the deviation ε is no more satisfactory small. Specifically, too many monomials have been removed and the polynomial does not fit the population. Regardless of which strategy is selected for the calculation, the addition/removal of terms induces significant computational costs. The strategy to circumvent this shortcoming is the usage of orthogonal polynomials.

Definition 6. *(Orthogonal polynomials) Two polynomials $P_r(x_1, .., x_n)$ and $P_s(x_1, .., x_n)$ are orthogonal if their inner product is equal to zero for $r \neq s$.*

2.4 ILPSS Solution

Our *ILPSS* solution is depicted in Figure 4. An MLR–based estimator (why we selected MLR over MPR is discussed in the next chapter) and an ILP–based scheduling mechanism S are interwoven in *ILPSS* to generate an HVAC schedule that optimizes energy consumption and keeps the temperature in all rooms within the comfort zones of their occupants. Furthermore, the comfort zones and the requests are provided by the users for their rooms of occupancy.

In this section, the modeling of the comfort zone is discussed first, followed by the principle of adversarial change of temperature in each room, caused by the environment, namely Newton's Law of Cooling. The section is concluded with an elaborate description of the scheduler.

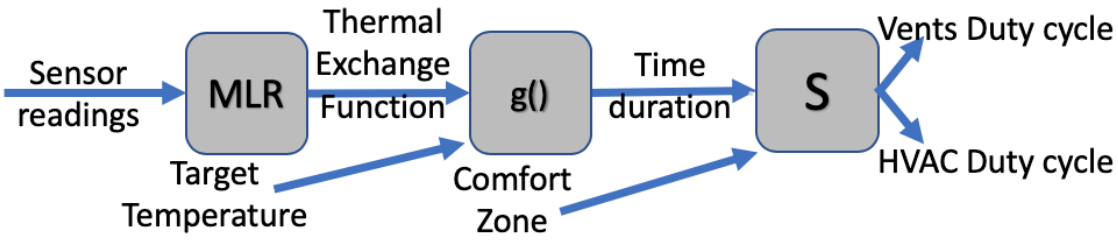

Figure 4: ILPSS Solution with scheduler S

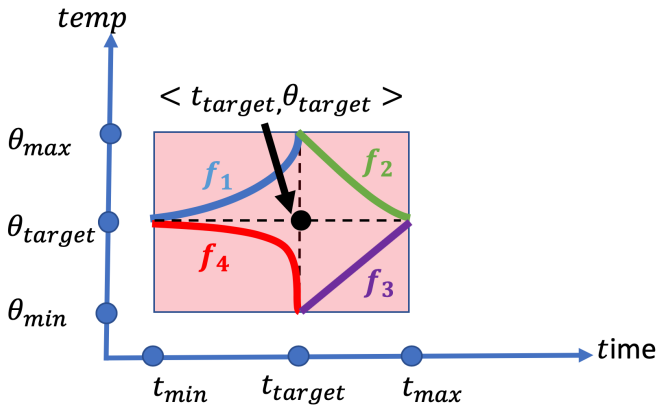

Figure 5: Comfort Zone of a user for a room

2.4.1 Modeling Comfort Zone

Recall that users' requests for a room i are in the format $u_i(i, t_c, \theta_{i,target}, t_{i,target})$ (Section 2.2). Note that users will be satisfied if the appropriate temperature is reached several minutes/seconds before or after the deadline. This tolerance in temperature and time is what defines "comfort zone", and is depicted in Figure 5. t_{min} and t_{max} show the earliest and latest point in time that the user tolerates reaching a temperature between θ_{min} and θ_{max}. Functions f_1, f_2, f_3, and f_4 are user-defined functions of time that yield temperature. The simplest way to model the comfort zone is to set f_1, f_2, f_3, and f_4 to be constants: comfort zone within the rectangle in Figure 5. It is to be noted, however, that this will be achieved only if two of the functions have explanatory variable *time* and dependent variable *temperature*, while the other two have explanatory variables *temperature* and dependent variable *time*. Furthermore, users with smaller tolerance to deviations will specify a smaller comfort zone. We constrain the comfort zone to be continuous (i.e., there is no "bubble" in it). Taking a practical perspective, it does not make sense for a user to tolerate two different temperatures at the same point in time, but to have no tolerance to a temperature in between them (i.e., the bubble). Formally:

Definition 7. *(*Comfort Zone*) The comfort zone cz_u of user u is defined by:*

$$cz_u =< t_{min,u}, t_{max,u}, \theta_{min,u}, \theta_{max,u}, f_{1,u}, f_{2,u}, f_{3,u}, f_{4,u} > \tag{2.9}$$

where $t_{min,u} \leq t_{i,target} \leq t_{max,u}$, $\theta_{min,u} \leq \theta_{i,target} \leq \theta_{max,u}$ for all requests u and $f_{1,u}, f_{2,u}, f_{3,u}$ and $f_{4,u}$ are defined by the user. We assume each user has one comfort zone for each room they visit. The comfort zone is modeled with Equation 2.10:

$$cz \;=\; \int_{t_{min}}^{t_{i,target}} f_1(\partial t) - \int_{t_{min}}^{t_{i,target}} f_4(\partial t) + \int_{t_{i,target}}^{t_{max}} f_2(\partial t) - \int_{t_{i,target}}^{t_{max}} f_3(\partial t) \tag{2.10}$$

Additionally, functions f_1, f_2, f_3, and f_4 should be integrable in the regions for which they are defined. It is to be noted, however, that for the case, whereby the comfort zone is a rectangle, function f_1 is the same as function f_2. Moreover, function f_3 is the same as function f_4. We can use the sum rule for integrals and simplify the comfort zone to two

integrals only that cover the time from t_{min} to t_{max}. By replacing f_2 with f_1 and f_3 with f_4, we get Equation 2.11

$$cz_{rectangle} = \int_{t_{min}}^{t_{i,target}} f_1(\partial t) - \int_{t_{min}}^{t_{i,target}} f_4(\partial t) + \int_{t_{i,target}}^{t_{max}} f_1(\partial t) - \int_{t_{i,target}}^{t_{max}} f_4(\partial t) \quad (2.11)$$

Moreover, we use the sum rule for integrals to simplify the comfort zone further to Equation 2.12:

$$cz_{rectangle} = \int_{t_{min}}^{t_{max}} f_1(\partial t) - \int_{t_{min}}^{t_{max}} f_4(\partial t) \quad (2.12)$$

Once the time-temperature estimations and the comfort zones of the users are defined, the solution has to schedule them. If the temperature change is needed in a room to achieve the temperature for a given request, the temperature should be increased or decreased. The amounts of time, for which thermally conditioned air has to be delivered to the rooms to achieve the target temperatures in them, can have partial, complete, or no overlap. Even though there is an infinite number of possible schedules, they can be reduced to a finite number of unique cases, which we call *canonical scheduling cases*. The canonical scheduling cases are discussed next.

2.4.2 Canonical Scheduling Cases

Intuitively, when scheduling the space conditioning of the rooms in a residential house, the need of serializing the heating and cooling should be considered. Moreover, one may wonder if there is a finite number of cases that should be addressed when a schedule for space conditioning is produced. In this subsection, we show that there are twelve scheduling cases that need to be handled when running our *ILPSS Solution*. We prove the following theorem:

Theorem 1 (HVAC Scheduling Cases)**.** *There are 12 canonical scheduling cases and the space conditioning of the rooms in a residential building fits in one of them, no matter how many rooms the house has.*

We use mathematical induction to show that all possible scheduling combinations fall into one of the cases that are discussed in this subsection. The case of a single room in a

house is not interesting, as it is trivial. Therefore, we will prove the theorem for $n \geq 2$ rooms in the house.

Base Case: The base case is a house with two rooms, $n = 2$. All possible combinations for 2 rooms are depicted in Figure 6. There are 12 cases on the diagram, labeled a to l. Each case consists of 2 lines, one for each room. The cases that require cooling are depicted in blue (dotted line) and the cases that require heating are depicted in red (solid line). A longer line depicts a longer amount of time for which thermally conditioned air should be blown into the room. Often the rooms require different amounts of conditioned air to be provided to reach their desired temperatures. For simplicity, we depicted only the case when the request that has arrived earlier also requires more air (and thus time) to reach the goal. The opposite case when the shorter request has an earlier arrival time is symmetric. The case when the temperature does not have to be changed implies that the target temperature has already been achieved and such rooms can be ignored. Given the intervals for two rooms there are three scenarios to be considered:

- the intervals for the two rooms do not overlap
- the intervals overlap completely (i.e., one contains the other one) and
- partial overlap

For example, these scenarios are depicted by the cases in Figures 6(a), 6(b), and 6(c), respectively. In these three cases, the temperature in both rooms should be decreased. The three cases when both rooms should be heated follow in Figures 6(d), 6(e), and 6(f). The mixture of heating and cooling is depicted in Figures 6(g) to 6(l). The cases when the cooling precedes the heating arrival are depicted in Figures 6(g), 6(h), and 6(i), and the opposite case in Figures 6(j), 6(k), and 6(l).

If the temperature in both rooms is expected to be adjusted in the same direction, there is no difference from a scheduling point of view if both rooms require heating or cooling. This effectively implies that Case d is identical to Case a, Case e to Case b, and Case f to Case c. The other possibility is to have a mixture of heating and cooling Similarly, the order of arrivals for cooling and heating when one room requires heating and the other cooling, does not make a difference from a scheduling point of view. It is to be noted that Cases j,

k, and l are identical to Cases g, h, and i, respectively. We conclude that there are six base cases for scheduling and we refer to them as *canonical cases*—they are depicted in Figures 6(a), 6(b), 6(c), 6(g), 6(h), and 6(i).

Induction hypothesis: Assume that for some $n = k$ rooms, whereby k is a positive integer and $k > 2$, the time frames for space conditioning of the rooms in a given house can be split into two equally large groups. One group will contain one room more than the other if the number of rooms is an odd number. If we consider only the earliest starting and the latest ending time frames in the group, it covers one of the canonical cases, discussed in the base case. Further, each group can be split into two subgroups. This operation can be repeated recursively until each subgroup contains no more than two rooms. Two rooms cover one of the canonical cases, as discussed in the base case. The induction hypothesis is that $k > 2$ rooms can be scheduled as one of the canonical cases from the base case.

Induction step: We will now show that for $n = k + 1$ the time frames for conditioning all rooms can be split into two groups that are in one of the *Canonical Cases*. If we arbitrarily take 1 room out of $k + 1$ rooms, we will have a set of k rooms, that we know that can be split into two groups in a way that builds a *canonical case*. If we consider one of these two groups arbitrarily and the room that has been taken out of the set earlier, as a group itself, they can be thought of as two groups. The time frame of the room that has no other room in its group, may not overlap with any of the time frames in the other group. This is a canonical case and it does not matter if the temperature in the single-room group should go in the same direction as all of the rooms from the other group because our *canonical cases* cover both uni- and mix-direction cases. If the time frame for the single-room group overlaps completely with the time frames of the rooms from the other group, it can be added to their group and this will be also a *canonical case*. Similarly, if the overlap of the time frame of that room with the other rooms is partial, this is another *canonical case*. This exhausts the possibilities for the relative position of the single-room group time frame with respect to the other time frames. Hence by mathematical induction, the scheduling of n rooms, where $n \geq 2$, falls into one of the twelve *canonical cases*. □

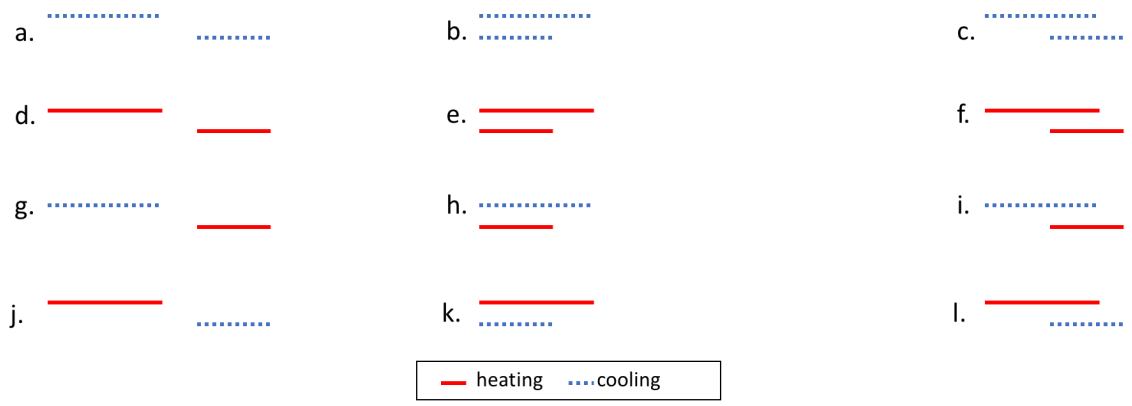

Figure 6: Canonical cases for scheduling

2.4.3 Newton's Law of Cooling

The outdoor environment impacts the temperature in all rooms of a building. Often the change of the temperature in rooms caused by the environment is in a direction that is opposite to the buildings' inhabitants' requests. The temperature difference between any two rooms that have a common surface (wall, floor/ceiling) also changes the temperatures in any two rooms, despite the insulation. Our estimation model captures the impact of both the environment and neighboring rooms. Note that when no thermally conditioned air is supplied for a sufficiently long amount of time, the exchange of heat between rooms will usually stop way ahead of the exchange of heat between the building and the environment. We adopted this optimization, as it is a conservative approach, whereby we consider the exchange of heat between each room and the environment when calculating divergence from the desired temperature once the delivery of thermally conditioned air for that room is over. We use this when we evaluate the deadlines in the comfort zones of users and the exchange is calculated with Newton's Law of cooling:

Definition 8. *(*Newton's Law of Cooling*) The rate of change of temperature, $\partial\theta$, with respect to time, ∂t, should be proportional to the difference between the temperature of the room, θ, and the ambient temperature θ_a (i.e., the temperature outside), and k is a non-negative*

constant:

$$\frac{\partial\theta}{\partial t} = -k(\theta - \theta_a) \tag{2.13}$$

When we multiply both sides by ∂t and divide by $\theta - \theta_a$, we will receive:

$$\partial\theta\frac{1}{\theta - \theta_a} = -k\partial t \tag{2.14}$$

Now we can differentiate both sides:

$$\int \partial\theta\frac{1}{\theta - \theta_a} = \int -k\partial t \tag{2.15}$$

Note that the left-hand side can be negative and after integration, it will be equal to the natural logarithm of the absolute value of the denominator:

$$ln|\theta - \theta_a| = -kt + C \tag{2.16}$$

Now we can raise to e both sides of the equation:

$$|\theta - \theta_a| = e^{-kt+C} \tag{2.17}$$

We know that e^C is a constant, thus we can substitute $e^C = C_1$:

$$|\theta - \theta_a| = C_1 e^{-kt} \tag{2.18}$$

The last step is to remove the absolute values:

$$\theta = C_1 e^{-kt} + \theta_a,\ when\ \theta \geq \theta_a \tag{2.19}$$

$$\theta = \theta_a - C_1 e^{-kt},\ when\ \theta < \theta_a \tag{2.20}$$

By integrating both sides of Equation 2.13, we receive Equation 2.19 and Equation 2.20. Having the two latest temperature measurements in a room, the ambient/outside temperature and the temperature $\theta_{min,u}$, we can calculate the maximum length of the time interval, $t_{Newton,i}$ between the end of the supply of thermally conditioned air for that room and $t_{min,u}$ (see Figure 5) [27]. The estimator is discussed next.

2.4.4 ILPSS Estimator

The intuition is to estimate what amount of thermally conditioned air is needed for a room to reach the temperature desired by its users. Once this information is available, it is translated into a period of time in which the vent in the room should be open and air should be blown through it. These estimated times are used to make a schedule that optimizes the duty cycle of the HVAC to reach the target temperatures in all rooms and maintains the temperatures within the comfort zones of users. The change of temperature for a room is influenced by the exchange of energy between the air in the room and the environment. We use the sensor readings to formally express as "energy exchange function f_{ee}", defined in [72]:

Definition 9. *(*Thermal energy exchange*) Given p sensors installed in a room to measure the factors that affect the change of the temperature in that room, the linear function:*

$$f_{ee}(x_1, x_2, ..., x_p) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p \tag{2.21}$$

measures the thermal energy exchange *at a given unit of time t. The x_i values are the measurements at time t, read from the sensors i, and β_i values are the coefficients of the function f_{ee}, $1 \leq i \leq p$.*

Our temperature change function is defined as follows:

Definition 10. *(*Temperature change*) Given function $f_{ee}(x_1, x_2, ..., x_p)$, the function*

$$g(x_1, x_2, ..., x_p, \delta_t) = f_{ee}(x_1, x_2, ..., x_p) \times \delta_t \tag{2.22}$$

calculates the temperature change (in degrees) in the room, if the sensor readings $x_1, x_2, ..., x_p$ do not change for a period of time δ_t.

The variables x_i and δ_t are called the *explanatory variables* and g is called *the response variable* in the MLR–based estimator. An example that explains how the coefficients of function g are calculated follows.

For this example, we used data from one of the datasets that we used for our experimental evaluation. A full description of the dataset is provided in Chapter 3. The example uses

Table 2: Example Sensor Convention

Parameter	**Unit**	**Description**
θ_{avg}	F	the average of the two temperature readings
P	inHg	barometric pressure
H	%	humidity in the room
θ_{out}	F	the temperature outside
ST	-	HVAC operating state
δ_t	sec	time difference between 2 cons. readings
$\Delta\theta$	F	temperature difference between 2 consecutive readings

eight consecutive readings (i.e., $w = 8$)[1] from the sensors in one of the rooms of the building that was used to collect the dataset. For this example, we used the following five types of sensors, namely the average of the readings from two temperature sensors, barometric pressure, humidity, the outside temperature, and the HVAC operating state. Moreover, we calculated the time difference of each reading from the previous reading, δ_t. We also calculated the temperature difference between two consecutive sensor readings, $\Delta\theta$. The notation for the different sensors is summarized in Table 2. The sensor readings, the time differences, and the temperature differences are presented in Table 3.

Following the definition of temperature change for function g, Equation 2.22, we multiplied each sensor reading with the time difference δ_t that we just calculated. We studied the effects of the time differences on the function accuracy in the next chapter. The last step is to add the error ε, needed for running the MLR. This is a requirement of the software package that we used to run MLR [35]. For this example, the error is denoted x_0 and it is set to 1. Moreover, it is multiplied by the time difference as all other sensor readings. Furthermore, all sensor readings are multiplied by δ_t and this is how the variables x_i are calculated. The values are presented in Table 4.

[1]Note that a window of size 8 will keep the Table 3 and Table 4 relatively short; window length is discussed further in Section 3.4.1.

Table 3: Example Readings

#	θ_{avg}	P	H	θ_{out}	ST	δ_t	$\Delta\theta$
1	67.51	988.36	44.80	0	2	74	-0.02
2	67.53	987.93	44.40	0	2	73	-0.09
3	67.61	987.93	44.32	0	2	72	0.09
4	67.53	988.14	44.21	0	2	1	0
5	67.53	988.14	44.21	0	2	73	0.04
6	67.49	988.14	44.40	0	2	74	-0.09
7	67.58	987.90	44.53	0	2	72	0.03
8	67.55	988.39	44.59	30.58	2	1	0

Table 4: Function g observed values for running MLR

#	x_0	x_1	x_2	x_3	x_4	x_5	g
1	74	4996.11	73138.64	3315.2	0	148	-0.015
2	73	4929.69	72118.89	3241.2	0	146	-0.085
3	72	4868.28	71130.96	3191.04	0	144	0.085
4	1	67.53	988.14	44.21	0	2	0
5	73	4929.69	72134.22	3227.33	0	146	0.035
6	74	4994.63	73122.36	3285.6	0	148	-0.085
7	72	4865.76	71128.8	3206.16	0	144	0.03
8	1	67.55	988.39	44.59	30.58	2	0

Table 5: Temperature Change Function Coefficients

Coefficient	**Value**
β_0	2.44E+12
β_1	0.03664154
β_2	0.007207253
β_3	0.001009652
β_4	-8.85E-05
β_5	-1.22E+12

Once we run the regression, we receive the coefficients β_i, presented in Table 5.

If a request for given target temperature for this room is received and there are no newer sensor data (i.e., the readings on row 8 in Table 3 are the latest data received from the sensors for the room), we calculate the temperature difference between the current temperature (i.e. 67.55) and the target temperature (from the request), namely 75. We use the latest sensor readings and the latest coefficients of the temperature change function (i.e., the coefficients from Table 5) in order to calculate the length of time for which the HVAC has to supply thermally conditioned air to this room in order to reach the desired target temperature. In this case, the time δ_t is unknown and because we can solve an equation for it (i.e., Equation 2.22), whereby g is the temperature difference, we use the coefficients from Table 5, for this example, and the latest sensor readings from the last row of Table 3.

It is to be noted, however, that the aforementioned regression techniques should collect a sufficient amount of data before they can be run and provide an estimation. This is known as the *"cold start"* issue and it is a legitimate concern. If the system produces new sensor readings not frequently enough, it may take a long time for the system to overcome the *"cold start"* issue. Fine-tuning the maximum amount of time between two consecutive deliveries of new sensor data should take into consideration the capabilities of the system to ingest data and the length of battery life for the power-constrained sensors. Producing data too often induces additional computations and, therefore, increases the requirements

to the estimator, which ingests the data. Moreover, the batteries of the battery-powered sensors will be depleted much faster when the sensors have to transmit data more often. On the contrary, not reporting the changes in the sensors' measurements to the estimator will decrease the accuracy of the provided estimations. Our system reports new data at least once every two minutes, as discussed in Section 3.3. Moreover, Section 3.4.1 explains why the estimator produces the most accurate results for a window of length 8. In other words, our system will need no more than 16 minutes to overcome the *"cold start"* issue. This time length is short enough to be accepted by most users.

2.4.5 ILPSS Scheduler

The *ILPSS* scheduling algorithm takes two types of input, namely new sensor readings x_{ij} and user requests.

When a new request is received, it is parsed in the *parseRequest()* primitive. In case several requests are received simultaneously, they are parsed sequentially in our framework, but the task trivially parallelizable. Moreover, in case more than one request comes for the same room, we consider the latest one received in time only. The *useCoefficientsToDeriveTime()* primitive is executed to derive the expected amount of time needed to reach the temperature for each request, given the last known sensor measurements for that room, assuming all air from HVAC goes to that room. When all estimations are in place, the ILP scheduler is run to generate a schedule, adhering to the objective of the solution, as defined in Section 3. Specifically, given the latest sensors readings for all rooms in a house and a set of requests for reaching the target temperature at each room at a specific time, an HVAC schedule should be produced that meets all user requests and optimizes the duty cycle of the HVAC. The algorithm is shown in Algorithm 1.

The *recalculateSchedule()* primitive is based on the *Integer Linear Programming Model* (ILP). ILP is a mathematical optimization problem, whereby the decision variables *st* are restricted to be integers, and the objective function and the constraints are linear. The

Algorithm 1 ILPSS Scheduling Algorithm

Input: *sensor readings, requests*

Output: *HVAC Schedule*

```
1: while 1 do
2:    if newData x_ij is available then
3:       slideWindow();                    /*ingest next reading*/
4:       recalculateRegressionCoefficients(x_ij);
5:    end if
6:    if newRequests are available then
7:       parseRequest();
8:       useCoefficientsToDeriveTime();
9:    end if
10:   recalculateSchedule();
11: end while
```

canonical form of ILP is as follows:

$$\begin{aligned} maximize\ c^T st \\ subject\ to\ Ast \leq b, \\ st \geq 0 \\ and\ st \in Z \end{aligned} \tag{2.23}$$

whereby c and b are vectors, c^T is the transpose of c and A is a matrix and st is the decision variable [84, 78]. The notation is presented in Table 1. There are two algorithms used for solving ILP problems, namely the *Branch and Bound* algorithm, and the *Cutting-Plane* algorithm. The former is computationally cheaper and thus is wildly adopted into ILP libraries.

We model the scheduling problem as a classical scheduling problem. The decision variables st_i define when the thermal conditioning of room i starts. The estimator provides the

duration δ_i of thermal conditioning needed to reach the target temperature $\theta_{i,target}$ by the deadline $t_{i,target}$. We calculate the coefficients c_i as follows:

$$c_i = \frac{|(\theta_{i,c} - \theta_{i,target})|}{\delta_i} \tag{2.24}$$

Our objective is to minimize the function $c^T st$. The coefficients c_i are the "penalty" we have to pay if we miss the deadline $t_{i,target}$. Furthermore, we define the constraints that (1) st_i should not start before the length of the duration δ_i and (2) the length of $t_{Newton,i}$ should be subtracted from the deadline. This will ensure the temperature in room i will be within the comfort zone. Formally:

$$st_i \geq t_{i,target} - (t_{Newton,i} + \delta_i) \tag{2.25}$$

Usually, residential buildings have single ducting that is shared for both heating and cooling. Thus, for our ILP-modeled scheduling, we have to ensure that cooling and heating jobs are not scheduled at the same time. We have to ensure that for each pair of rooms i and j, whereby one needs heating and the other one needs cooling, having durations δ_i and δ_j and times by which the temperatures should be reached $t_{i,target}$ and $t_{j,target}$, respectively, the difference between the starting times st_i and st_j is at least δ_i for $st_i < st_j$. The same is valid if st_j precedes st_i. This decision structure is modeled with the introduction of a new binary variable $y_{i,j}$. It has value 1 when st_i is smaller than st_j and 0 otherwise. We add two additional constraints for each pair of rooms that require a change of temperature in different directions:

$$\begin{aligned} My_{i,j} + (st_i - st_j) \geq \delta_j \; and \\ M(1 - y_{i,j}) + (st_j - st_i) \geq \delta_i \end{aligned} \tag{2.26}$$

where M is a very big constant, in particular, $M > \delta_i \; \forall i$. Given that M will be multiplied by 0 for exactly one of the cases in Equation 2.26, we want the two starting points st_i and st_j to have a difference of at least δ_k for $k = min\{i, j\}$. Moreover, for the equation where M is multiplied by 1, the value of the expression in the parenthesis will be negative, thus adding M ensures that the inequality holds. We do not need additional pairs of constraints for rooms that either both need heating, or both need cooling.

Table 6: Energy Saving Approaches Using HVAC Scheduling

	Non-HVAC Scheduling	HVAC Scheduling
Non-ML Solution	[83, 82]	Naïve
ML Solution	[40, 91, 63, 12] [38, 32, 60, 7, 8, 41]	D-DUAL

The last step is to incorporate the "penalty". Let pen_i be an unrestricted variable and $pen_i \geq 0$. Then

$$st_i + \delta_i + t_{Newton,i} + pen_i = t_{i,target} \tag{2.27}$$

When pen_i is positive, the deadline is met. When it is negative, the deadline is not met. We substitute

$$pen_i = penMinus_i - penPlus_i \tag{2.28}$$

$$st_i + penMinus_i - penPlus_i = t_{i,target} - (\delta_i + t_{Newton,i}) \tag{2.29}$$

All variables st_i, $penMinus_i$, $penPlus_i$ are non-negative. Given the duality nature of ILP problems, the objective function becomes:

$$minimize \sum_{\forall rooms} c_i * penPlus_i \tag{2.30}$$

Once the ILP problem is solved, the variable st_i for each room i will provide the starting time for supplying thermally conditioned air to the respective room i.

2.5 Comparison with State-of-the-art

To the best of our knowledge, there is no prior work that combines ML and scheduling in an effort to optimize energy consumption in space conditioning of buildings only with local resources (i.e., no external network connectivity). However, as summarized in Table 6, ML techniques have been employed in HVAC optimization. One example of a novel regression approach, which aims at decreasing the computational cost of running auto-regression on IoT gateways, is presented in [63]. The paper discusses an incremental approach to calculate auto-regression, whereby the model is updated in $O(k)$ calculations at every step—a step is the next set of variables to be fed into the regression model. The auto-regression coefficients are calculated only once over a span of several steps. The computational cost to recompute the coefficients has complexity of $O(k^3)$. The alternative naive approach is to recalculate the coefficients at each step, having cost of $O(k^2)$. Moreover, the incremental update of the model and the sparse recalculation of coefficients raises a trade-off between computation cost and accuracy that is not thoroughly studied in the paper. Our approach recalculates the coefficients of the regression at each step. This ensures improved accuracy at computational cost with complexity of $O(k^2)$.

The current state-of-the-art HVAC scheduling is predictive control for energy-efficient buildings [41]. That solution does not consider deadlines by which the desired temperature should be achieved (i.e., it relies on static programs that control at what point in time the HVAC should start supplying thermally conditioned air to a room, in order to reach the temperature, specified in the program). Furthermore, it is tailored to buildings, whose ducting allows heating and cooling of spaces to happen in parallel. Specifically, there are heating coils mounted in each vent and the air gets thermally conditioned right before it is delivered to the rooms. Additionally, that solution assumes that the HVAC system uses recirculated air—i.e., air that is sucked from the rooms back into the HVAC system.

Room-level zoning of HVAC is tackled in [83, 82]. The three pillars of the work are HVAC dimensioning, occupancy prediction, and thermal leakage. The first study, presented in the paper argues that the HVACs installed into houses are under-dimensioned and thus their efficiency is decreased. Smaller HVAC installations and retrofitting of the buildings to room-

level zoning is one of the directions to optimize energy consumption for space conditioning. Another dimension is improved insulation and the last one is "smart" resource allocation, whereby rooms that have no occupants are not conditioned. Our work also tackles room-level zoning of HVAC and to a certain extent we base our work on the studies in [83]. We assume that we can accurately detect room occupation. In contrast, our focus is on HVAC duty cycle scheduling that aims to decrease energy consumption without affecting the comfort of the users rather than sensing the presence of occupants in rooms and use that information to control vents.

Many buildings in the US are equipped with thermostats to control the temperature and sensors to detect the presence/absence of humans. Such "smart" buildings have systems in place to control the lighting as well. Recently, buildings are built with sensors and actuators that allow fine-grain control of the temperature at the room level and the duty cycles of the lighting. Moreover, the emergence of the Internet of Things (IoT) enabled new technologies that facilitate increased autonomy in space conditioning and lightning—smartphone-based geo-fencing, as well as connected thermostats, power plugs, and light bulbs aim to improve quality of life [64, 65, 74, 23].

The scope of the work in [82] is extended from residential to commercial buildings. Sensors are installed in common areas of the building to improve the accuracy of sensing the presence of occupants. The intuition is that for a given room to be occupied, its occupants walk down the hallway in order to reach the room. Walking in hallways that are usually long provide opportunities to detect the direction of walking of people and thus improve the accuracy of sensing of people in rooms—i.e., the cases whereby no people walking towards the rooms cannot lead to people being present in rooms. The assumption is there are no external entrances to those rooms. The occupancy information is used further to control vents on a per-room basis and thus optimize the energy spent for space conditioning by avoiding the conditioning of rooms that lack the presence of people. We argue that such an approach is impractical as most houses have at least 2 entrances, therefore the approach may violate the comfort of the occupants and thus is not optimal. Our framework achieves energy savings and does not affect the comfort of users. Moreover, our framework controls the scheduling of the HVAC systems in terms of duty cycles of furnaces/air-conditioning units. The authors

of [7, 8] used a model-predictive control mechanism that detects occupants in a room and uses that information to instruct the HVAC to compensate for the presence of people in the thermally conditioned room. The solution relies on semiparametric regression to calculate the temperature change in the room, given the presence of occupants. The energy gain they achieve is driven by the fact that the presence of people in a room increases the temperature in the room. Our work builds on top of that assumption and achieves energy saving from both an accurate *thermal energy exchange function* that can make use of occupants if such information is available, but also our novel integer linear programming scheduling mechanism that optimizes the work cycles of the HVAC.

D-DUAL [72] is our initial solution. It combines ML and HVAC scheduling. Similar to [82], our solution is applicable to smart homes. Data from sensors in smart homes are fed into an MLR-based estimator that identifies the length of time frames for blowing thermally conditioned air in each room in order to reach the temperatures desired by users. Our solution combines three scheduling principles, namely *elevator algorithm*, *latest deadline*, and *shortest job first* to optimize the duty cycles of furnaces and air conditioning [29, 15]. Our *ILPSS* solution uses an MLR-based estimator as well, but it differs from *D-DUAL* in the scheduling solution—we employ an ILP-based scheduler in order to guarantee that the target temperatures will be achieved within the comfort zones of users. Moreover, we incorporated quality of service guarantees in *ILPSS* that leverage our *comfort zone*.

Even though statistical techniques have been used in schemes to save energy in HVAC [40, 91, 63], none of these existing solutions aimed at optimizing the HVAC scheduling (for examples see Table 6). These solutions used various ML algorithms for estimation as appropriate. In our case, we selected MLR and MPR for their relative simplicity of implementation, highly accurate results, and modest memory footprint requirements. They also do not require tuning of parameters for achieving accurate results.

2.6 Summary

In this chapter, we introduced *ILPSS*, an IoT solution that schedules the duty cycles of HVAC systems, that considers both energy savings and user comfort. We presented its components, specifically its estimator and its scheduler. Moreover, we proved that there are only 12 canonical scheduling cases for HVAC scheduling. We intentionally selected only sensors that either explain the temperature in a room or that have a high impact on changing the temperature in a room. Specifically, the temperature sensors and the pressure sensors explain the temperature in the room, while the higher the humidity, the easier it to exchange thermal energy with the surroundings, given that water is a conductor. Furthermore, the temperature outside is an indicator of whether the environment will be adversarial or cooperative to reaching the target temperature as per the request for the room. Similarly, the state of the HVAC can be an adversary or help in reaching our goal. We presented a lightweight computational solution that can be deployed on a cheap IoT gateway, as shown in the next chapter. It carries the computations locally to avoid data shipping, which would raise security and privacy concerns.

3.0 Residential HVAC: Experimental Evaluation

In this chapter, we present the details about our experimental framework, the three datasets we used to run experiments on, as well as the results from the experimental evaluation, performed on a resource-contained device. Our experimental evaluation shows that our proposed solution saves energy (up to 45%) and meets users' comfort needs, compared to commodity and current smart HVAC systems.

Our experimental testbed is presented in Section 3.1. In Section 3.2 we discuss the metrics for our evaluation, followed by the datasets in Section 3.3. The results of the evaluation are presented in Section 3.4. The chapter is concluded with a summary in Section 3.5.

3.1 Testbed

We implemented our solution and its algorithm in Java 1.8. We ran the experiments on a Raspberry Pi Zero W (Figure 7) with 1 ARM CPU, running at 1GHz, 512MB of RAM memory, and 64GB MicroSDEX micro SD card. The operating system used was Raspbian Stretch Lite, based on Debian 9. We used the GEKKO Optimization Suite for solving the ILP problem [10]. The Raspberry Pi Zero W is an example of an affordable computer that can be installed and carry out the computation locally on-premise, and, thus, mitigating the privacy and security risks of shipping data to the cloud. Moreover, for our experimental evaluation, we adopted the canonical scheduling cases, as defined in [72], as they cover all possible scheduling cases.

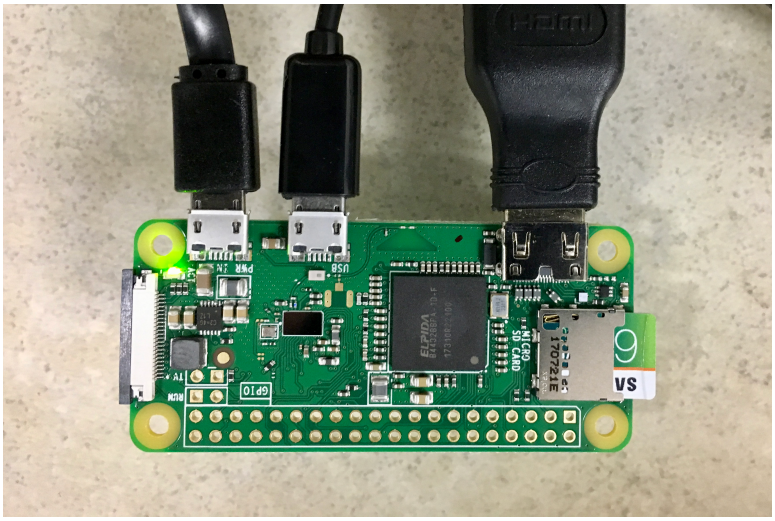

Figure 7: Raspberry Pi Zero W

3.2 Metrics

We evaluated the performance of the systems in terms of *wall-clock time*, *energy decrease*, and *monetary cost*.

Wall-Clock Time: We measure the scalability of our solution when deployed on low-cost hardware. It is measured as the amount of time it takes for the computer (i.e., Raspberry Pi Zero W) to run our algorithm with the number of sensor readings in our experiments. We used 5 sensors, specifically the average of the readings of the two temperature sensors in the room, barometric pressure, humidity in the room, outside temperature, and HVAC operating state.

Energy Decrease: This is our optimization criterion. The length of the duty cycle of HVAC can be translated to the energy spent; that is, the longer the HVAC works, the higher the amount of energy spent on space conditioning. It is to be noted, however, that the optimization of the duty cycle directly translates to energy savings, as explained further into detail later in this subsection. The metric estimates the amount of energy spent as a percentage of the amount of energy spent by a naive system to achieve the same goal.

The energy consumption of the furnace per unit of time is calculated as the fraction of the capacity of the furnace and the product of the BTU metric of the fuel, used by the furnace, the efficiency of the furnace, and the constant for conversion of BTU to joule. The formula is presented in Equation 3.1:

$$Energy_{perhour} = \frac{furnace_{capacity}}{BTUmetric_{fuel} \times Efficiency_{furnace} \times Constant_{BTUtoJ}} \tag{3.1}$$

Moreover, the furnace is a single-stage furnace and, therefore, it has a uniform consumption of gas per unit of time, as discussed in Section 2.2. The energy consumption of the furnace for a period of δ_t will be the product of δ_t and $Energy_{perhour}$, as presented in Equation 3.2:

$$Energy_{\delta_t} = \delta_t \times \frac{furnace_{capacity}}{BTUmetric_{fuel} \times Efficiency_{furnace} \times Constant_{BTUtoJ}} \tag{3.2}$$

Our energy decrease metric is presented in Equation 3.3:

$$e_{savings} = \frac{\delta_t \times Energy_{perhour}}{\delta_{naive} \times Energy_{perhour}} \tag{3.3}$$

We can remove $Energy_{perhour}$ from both numerator and denominator. Equation 3.4 presents the simplified formula for energy decrease:

$$e_{savings} = \frac{\delta_t}{\delta_{naive}} \tag{3.4}$$

Moreover, we refer to *"Naive"* as the family of commodity systems which are widely available and adopted today, whereby the users cannot submit a request for temperature adjustments upfront. When the users feel the need to change the temperature in the room they are in, they reach out to the thermostat and change the temperature on it. Furthermore, there is no additional context, which is considered by the control system of the HVAC unit in order to schedule its duty cycle. For our experimental evaluation, we calculated the consumption of *"Naive"* by extracting from our datasets the actual time length for which the furnace/AC was on in the time frames in which our requests were submitted and satisfied. The datasets and the requests are discussed in detail in the following subsection.

3.3 Datasets

HiberSense Historical Data[1] : The dataset we used in our experiments is the proprietary dataset we used in [72]. It consists of 5000 measurements from the HVAC-related data per room within one family house for three days, collected between 2018-02-01 and 2018-02-03. The house is a duplex house in Pittsburgh, PA. It has two rooms on the first floor and two rooms on the second floor. Both of the rooms on the second floor are bedrooms. The first floor has a kitchen and living room. Smart vents are installed in all four rooms. The vents are equipped with smart sensing units that can measure temperature, humidity, atmospheric pressure and also report the battery voltage of the units. Information about the state of the vent in each room is available as well. Vents can be either open or closed. Moreover, each room is equipped with an additional sensing unit, which is typically installed on the wall, opposite to the biggest window in the room. The sensing units are installed about 4.5 feet from the ground. The data, measured by these sensing units and available for each room, contains the measurements for motion, the voltage of the sensors' battery, two different temperature measurements, humidity level, atmospheric pressure, and light level. Furthermore, the dataset contains the following reading for the thermostat in the house: state of the HVAC fan (on/off; i.e., shows whether the fan of the HVAC system is running, regardless of the rest of the HVAC system), mode of the HVAC (off, heat, cool), temperature set on the thermostat, override the state of the thermostat (i.e., when this feature is set, it maintains the HVAC unit running for a user-defined period of time and, therefore, it can be used to increase the ventilation in the building, or keep the HVAC shutdown in an effort to decrease the levels of noise in the building when the HVAC is off), hold state (i.e., overrides the predefined schedule in the thermostat and the currently set temperature is maintained until the hold state is deactivated), and the method the data was collected (push/pull). The outside temperature was collected once per hour. All the data is timestamped with precision within a second. The sensors in each room reported new measurements whenever there was a difference in the value of at least one reading, compared to the last values sent, or if a 15-minute time span passed. It is to be noted, however, that the difference should be beyond

[1]The dataset has been provided by HyberSense and it has been anonymized.

a certain threshold, which is set by the thermostat manufacturer and, therefore, is beyond the scope of this work. The thermostat is reported every 3 seconds. For all experiments, except the first one, we fed the estimator with eight consecutive measurements, called a "window". We know from our previous work that windows of length less than eight produce inaccurate results, and windows of sixteen or more do not produce accurate results either [72, 73, 4, 3]. The former is explained by the fact that every single reading in a short window of measurements has a significant impact on the estimated f_{ee}. Therefore, when a single reading deviates significantly from the other readings in the window, the prediction is inaccurate. Furthermore, when the window is too long, the predictor cannot adapt to the changes in function f_{ee} quickly enough, resulting again in inaccuracies.

User preference User preference levels were collected as well, following the model, as described in Section 2.4.1: the minimum and the maximum temperature the user tolerates when in the room, as well as the safety boundary temperatures.

For our experiment on energy savings we used sets of requests, each of which covers 3 rooms. Six of those sets were derived from the HiberSense Historical Data dataset. Given the period of the year when the dataset was collected, we created the subsequent 5 sets synthetically in order cover all 12 canonical scheduling cases. The sets of requests follow:

- case (a)—the request for the first room was submitted at 00:10:32 on 2/1/2018, it was a cooling request and the system had 90 min. to reach the target temperature of 78F; the request for the second room was submitted at 00:10:30 on 2/1/2018, it was a cooling request and the system had 90 min. to reach the target temperature of 79F; the request for the third room was submitted at 00:55:33 on 2/1/2018, it was a cooling request and the system had 90 min. to reach the target temperature of 78F.
- cases (d) and (f)—the request for the first room was submitted at 12:00:09 on 2/1/2018, it was a heating request and the system had 60 min. to reach the target temperature of 76F; the request for the second room was submitted at 12:26:11 on 2/1/2018, it was a heating request and the system had 60 min. to reach the target temperature of 76F; the request for the third room was submitted at 12:15:10 on 2/1/2018, it was a heating request and the system had 60 min. to reach the target temperature of 77F.

- case (j)—the request for the first room was submitted at 05:19:05 on 2/2/2018, it was a heating request and the system had 90 min. to reach the target temperature of 67F; the request for the second room was submitted at 06:15:07 on 2/2/2018, it was a cooling request and the system had 90 min. to reach the target temperature of 63F; the request for the third room was submitted at 05:50:06 on 2/2/2018, it was a heating request and the system had 60 min. to reach the target temperature of 66F.
- cases (h) and (k)—the request for the first room was submitted at 20:04:50 on 2/2/2018, it was a heating request and the system had 30 min. to reach the target temperature of 71F; the request for the second room was submitted at 20:05:50 on 2/2/2018, it was a cooling request and the system had 10 min. to reach the target temperature of 71F; the request for the third room was submitted at 20:05:50 on 2/2/2018, it was a heating request and the system had 25 min. to reach the target temperature of 74F.
- cases (e) and (f)—the request for the first room was submitted at 00:00:01 on 2/3/2018, it was a heating request and the system had 90 min. to reach the target temperature of 72F; the request for the second room was submitted at 00:51:04 on 2/3/2018, it was a heating request and the system had 10 min. to reach the target temperature of 63F; the request for the third room was submitted at 00:20:02 on 2/3/2018, it was a heating request and the system had 90 min. to reach the target temperature of 72F.
- case (b)—the request for the first room was submitted at 20:39:07 on 2/3/2018, it was a cooling request and the system had 30 min. to reach the target temperature of 69F; the request for the second room was submitted at 20:40:08 on 2/3/2018, it was a cooling request and the system had 30 min. to reach the target temperature of 71F; the request for the third room was submitted at 20:40:07 on 2/3/2018, it was a cooling request and the system had 30 min. to reach the target temperature of 69F.
- case (c)—the request for the first room was submitted at 20:04:50 on 2/2/2018, it was a cooling request and the system had 30 min. to reach the target temperature of 69F; the request for the second room was submitted at 20:40:50 on 2/2/2018, it was a cooling request and the system had 30 min. to reach the target temperature of 69F; the request for the third room was submitted at 20:40:50 on 2/2/2018, it was a cooling request and the system had 50 min. to reach the target temperature of 69F.

- case (e)—the request for the first room was submitted at 20:39:07 on 2/3/2018, it was a heating request and the system had 30 min. to reach the target temperature of 69F; the request for the second room was submitted at 20:40:08 on 2/3/2018, it was a heating request and the system had 30 min. to reach the target temperature of 71F; the request for the third room was submitted at 20:40:07 on 2/3/2018, it was a cooling request and the system had 50 min. to reach the target temperature of 69F.
- case (g)—the request for the first room was submitted at 05:19:05 on 2/2/2018, it was a cooling request and the system had 90 min. to reach the target temperature of 60F; the request for the second room was submitted at 06:15:07 on 2/2/2018, it was a heating request and the system had 90 min. to reach the target temperature of 63F; the request for the third room was submitted at 05:50:06 on 2/2/2018, it was a cooling request and the system had 90 min. to reach the target temperature of 72F.
- case (i)—the request for the first room was submitted at 20:04:50 on 2/2/2018, it was a cooling request and the system had 30 min. to reach the target temperature of 69F; the request for the second room was submitted at 20:04:50 on 2/2/2018, it was a cooling request and the system had 30 min. to reach the target temperature of 69F; the request for the third room was submitted at 20:50:50 on 2/2/2018, it was a heating request and the system had 40 min. to reach the target temperature of 74F.
- case (l)—the request for the first room was submitted at 20:04:50 on 2/2/2018, it was a heating request and the system had 30 min. to reach the target temperature of 75F; the request for the second room was submitted at 20:04:50 on 2/2/2018, it was a heating request and the system had 30 min. to reach the target temperature of 76F; the request for the third room was submitted at 20:50:50 on 2/2/2018, it was a heating request and the system had 40 min. to reach the target temperature of 69F.

Moreover, for the experiments with ILPSS, the comfort zone for all requests was set to plus/minus 1 degree Fahrenheit deviation from the target temperature. The maximum allowed deviation for the deadline is plus/minus 5 minutes.

Occupancy plays role in temperature change on a per room basis. Each person emits $50W$ of power when still. This number can go to as high as $260W$ when actively exercising

[9]. We incorporated the presence of humans in our energy exchange function as another sensor reading. For our experimental evaluation, we used the dataset produced by scholars at the University of Texas San Antonio [24]. The dataset contains data for June 2014, for three rooms in one house, namely a kitchen, living room, and bedroom. We duplicated the bedroom data from [24] to accommodate the two bedrooms in our dataset. The data granularity is one reading for each room every fifteen minutes, or 96 per day.

3.4 Experimental Evaluation

In this section, we present the results from the experimental evaluation of our framework. The experiments' parameters and their possible values are summarized in Table 7. *ILPSS* operates leveraging *horizons of time*, whereby the horizons are continuous time frames that start now and last for the next 6 hours. In the first experiment, we study the impact of the window length on the accuracy of the estimation. The goal of the second experiment is to quantitatively evaluate the suitability of MLR and LASSO regressions. In our third experiment, we study how our solution scales up with an increase in the number of rooms. In the fourth experiment we measured the estimation accuracy of our two regression techniques when occupancy is added as a dependent valuable to the regression model. In this experiment, we measured the energy savings caused by our ILPSS solution against the commodity solutions available today (we call it Naive as discussed above).

3.4.1 Experiment 1: Estimation Accuracy

In our first experiment, we measured the time difference between our estimation, using our regression estimation model with MLR, and the actual amount of time needed to reach the target temperature. We select sliding windows of different sizes of consecutive sensor measurements in a given room and run the regression estimation model to produce the estimated time. We use 5 different sensors to feed the regression model, specifically the average of the 2 temperature readings from the IoT unit that does not control the vent, the

Table 7: Experimental Parameters

Parameter	**Description**	**Values**
rooms	number of rooms	0, 4, 8, 12, 16, 20 , 24, 28, 32
win	window length	8, 16, 32, 64, 128, 256, 512
hozn	horizon length	6
occup	occupancy	on (1) / off (0)
reg	regression technique	MLR, MPR, Lasso
sched	scheduler	Naive, D-DUAL, ILPSS

pressure and humidity readings of the same unit, the external temperature, and the state of the HVAC (i.e., heating, cooling or off). Occupancy is not considered in this experiment. The results are depicted in Figure 8. Further, we take the most recent record of sensor readings for that particular room (which includes the temperature reading), replace the temperature reading with the temperature we want to reach. The temperature we want to reach is fetched from the subsequent record (i.e., the one following the end of the window), and we derive the estimated time it takes for the target temperature to be reached. We then compare the estimated duration against the actual duration observed in the dataset. We then slide the window by 1 record and repeat the steps until we exhaust the dataset.

We run the experiment for seven different window lengths–specifically, 8, 16, 32, 64, 128, 256, and 512 and for each room independently. It is to be noted, however, that our temperature change function is a function of 6 variables, specifically, the 5 aforementioned sensor readings and time. According to the Theorem for Lagrange interpolation of multivariable functions, we need at least 7 distinct points in order to find a unique polynomial for them. We experimented with window lengths that are powers of 2 and, therefore, 8 is the shortest window that satisfies the requirements for Lagrange interpolation of the temperature change function [16]. In Figure 8, we show the average of the five runs for all rooms and the standard deviation, whereby the standard deviation is calculated for the five runs for all rooms for a single-window length. On the x-axis of the figure, we have the different window lengths.

On the y-axis, we show the average of the differences between the estimated values and the actual values. Our experiment showed that the MLR accuracy consistently decreased for increasing window size. This is not surprising as the most recent values of sensor readings are the most relevant in affecting a current temperature. Furthermore, for our test dataset, a window length of 8 readings predominantly provides the best results. Windows of length less than 8 do not provide accurate results [73, 4, 5]. It is to be noted, however, that given that we use 5 sensors, we cannot have a window of length less than 5, because otherwise, the regression will not work. Similarly, a system of equations with 5 unknown variables will not have a unique solution when solved for less than 5 non-proportional to each other equations. We compared the estimated value for the amount of time needed to reach the target temperature to the ground truth, and we observed that the estimated value consistently deviates by around 100 seconds for all rooms and for a window length of 8.

Moreover, we ran the same experiment for six more buildings. We ran it for a window length of eight consecutive readings. All of the buildings are in Pittsburgh, in Pennsylvania. They have between three and eighteen rooms. The data from these buildings was also provided by HiberSense. The data was collected between 12/18/2018 and 2/23/2019. Each room in each of these houses is equipped with the same sensing units and collects the same type of information as the first building that we got data for. The outside temperature data was downloaded from the website of Wunderground [85]. The outside temperature was measured at least once per hour. We used the last received value for running our estimator. The results are presented in Figures 9 - 14.

If we consider the 58 rooms of these 7 buildings to be a sample size of all rooms in buildings in the United States, we calculate the probability that our estimator is not off by no more than 100 s. Our results show that for exactly 29 rooms the estimation deviates by 100 s at most.

$$Standard\ Error = \sqrt{\frac{(sample\ probability) * (1 - sample\ probability)}{sample\ size}} \tag{3.5}$$

Given that we calculated the *sample probability* to be 0.5 (i.e., 29 of 58 rooms) and our *sample size* is exactly 58 when we used this data in Equation 3.5, we calculate that

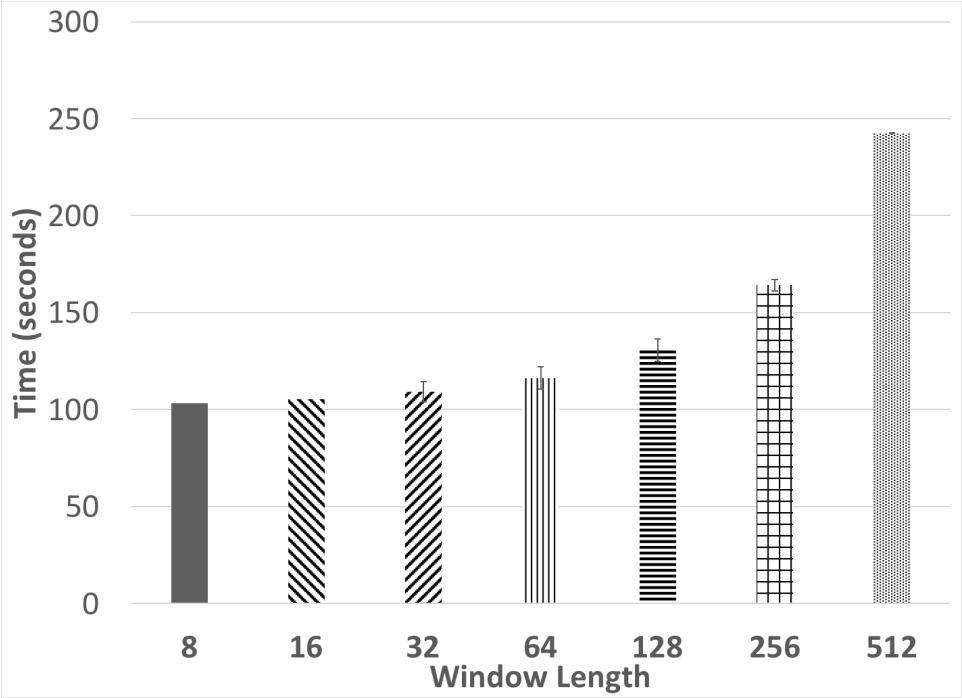

Figure 8: Time difference (accuracy) between estimated and actual amount of time needed to reach the target temperature, with MLR for different window lengths for 4 rooms in Building 1

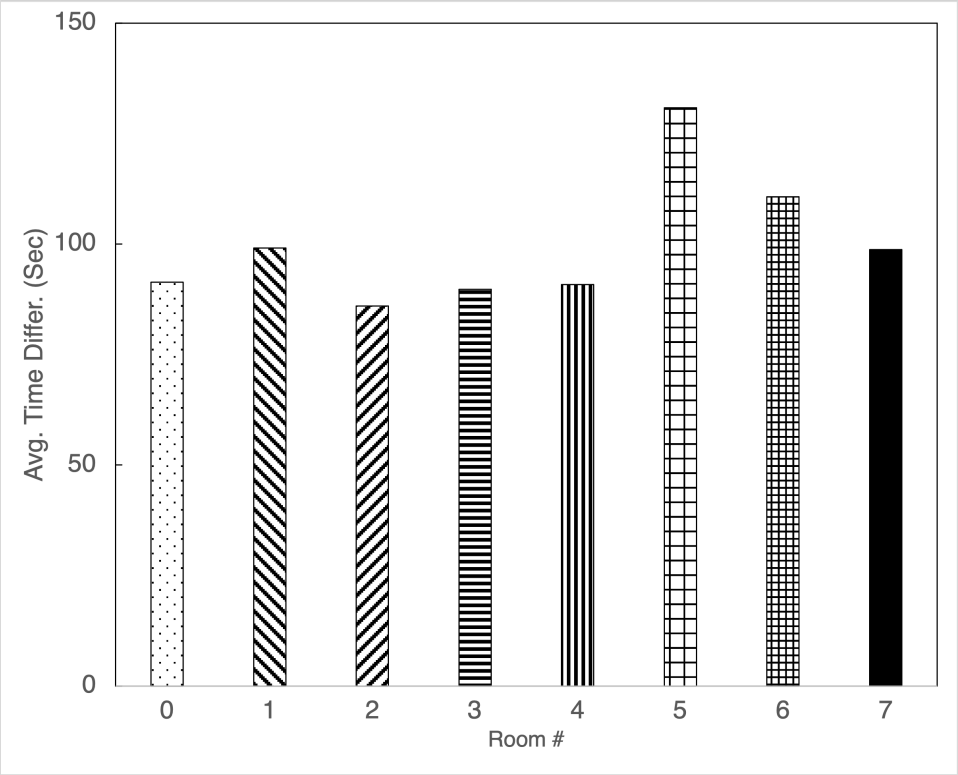

Figure 9: Time difference (accuracy) between estimated and actual amount of time needed to reach the target temperature, with MLR for window length 8 for 8 rooms in Building 2

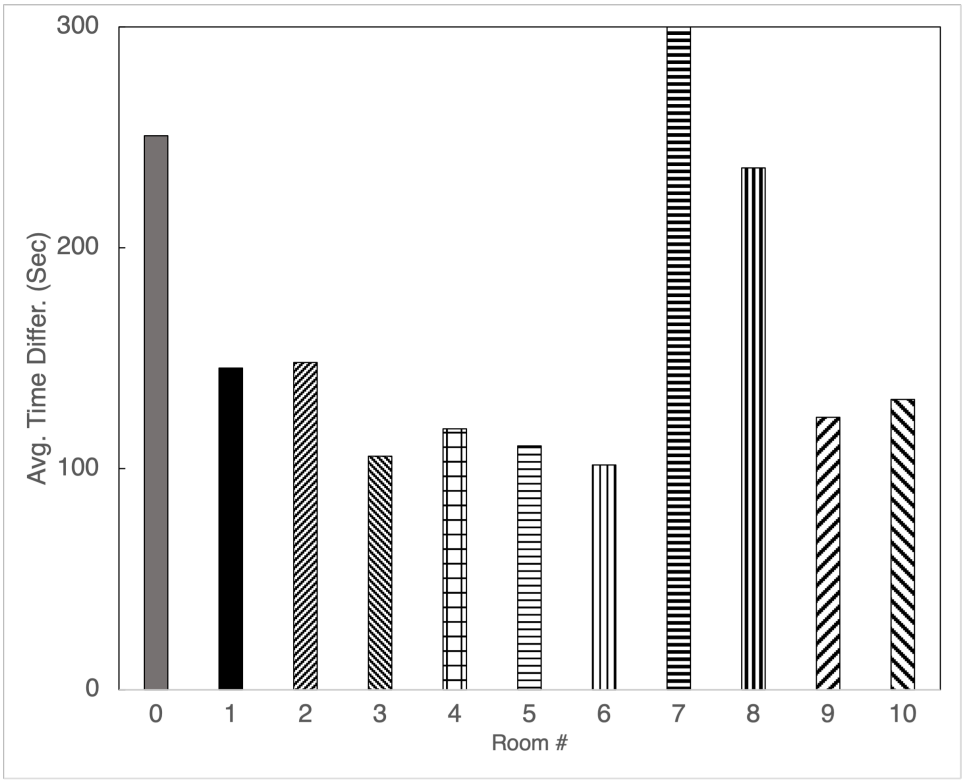

Figure 10: Time difference (accuracy) between estimated and actual amount of time needed to reach the target temperature, with MLR for window length 8 for 11 rooms in Building 3

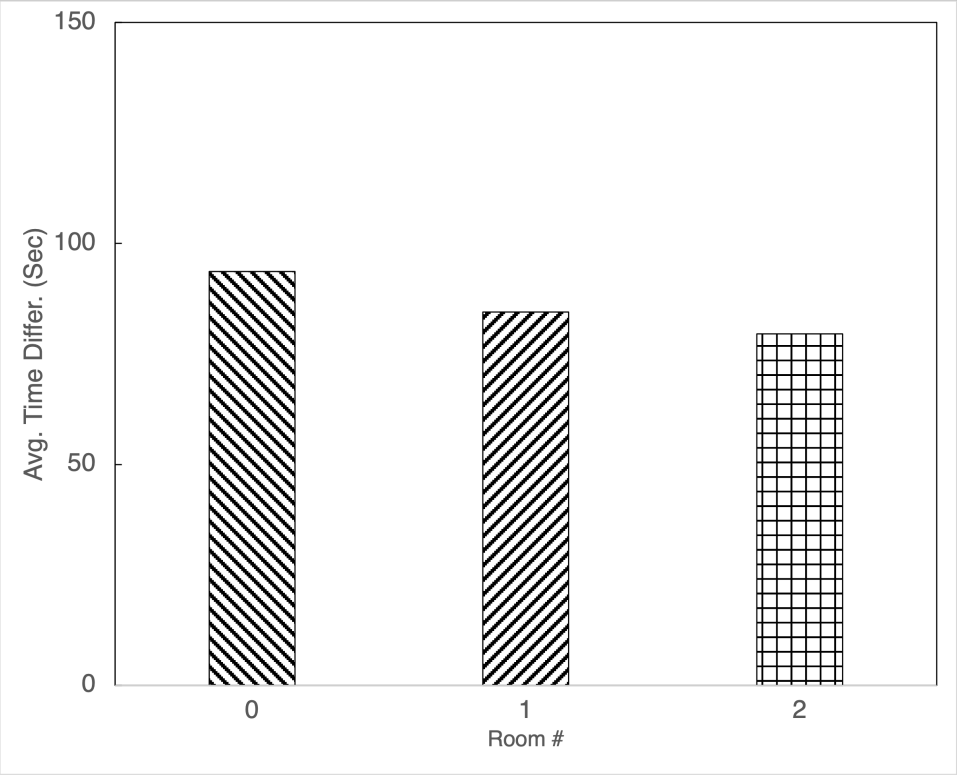

Figure 11: Time difference (accuracy) between estimated and actual amount of time needed to reach the target temperature, with MLR for window length 8 for 3 rooms in Building 4

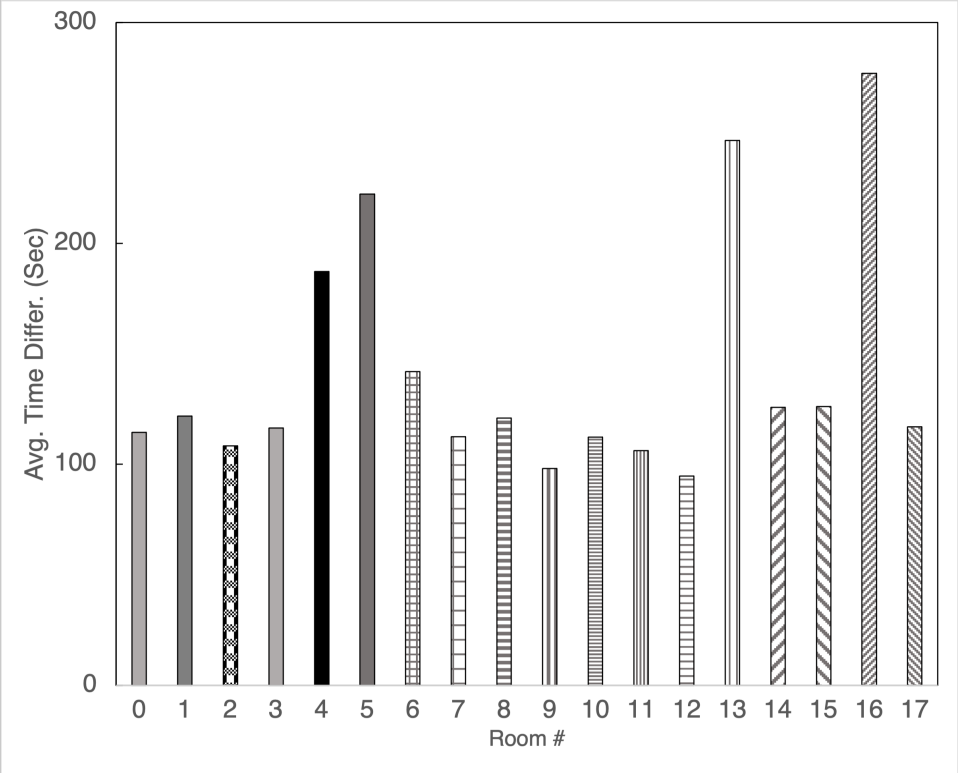

Figure 12: Time difference (accuracy) between estimated and actual amount of time needed to reach the target temperature, with MLR for window length 8 for 18 rooms in Building 5

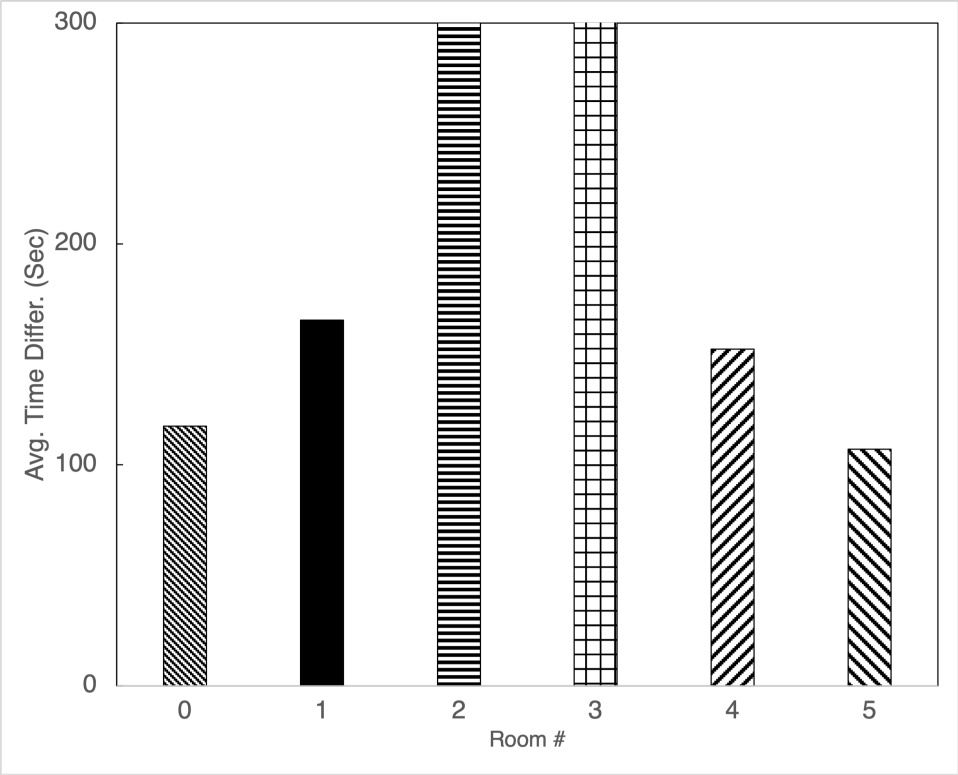

Figure 13: Time difference (accuracy) between estimated and actual amount of time needed to reach the target temperature, with MLR for window length for 6 rooms in Building 6

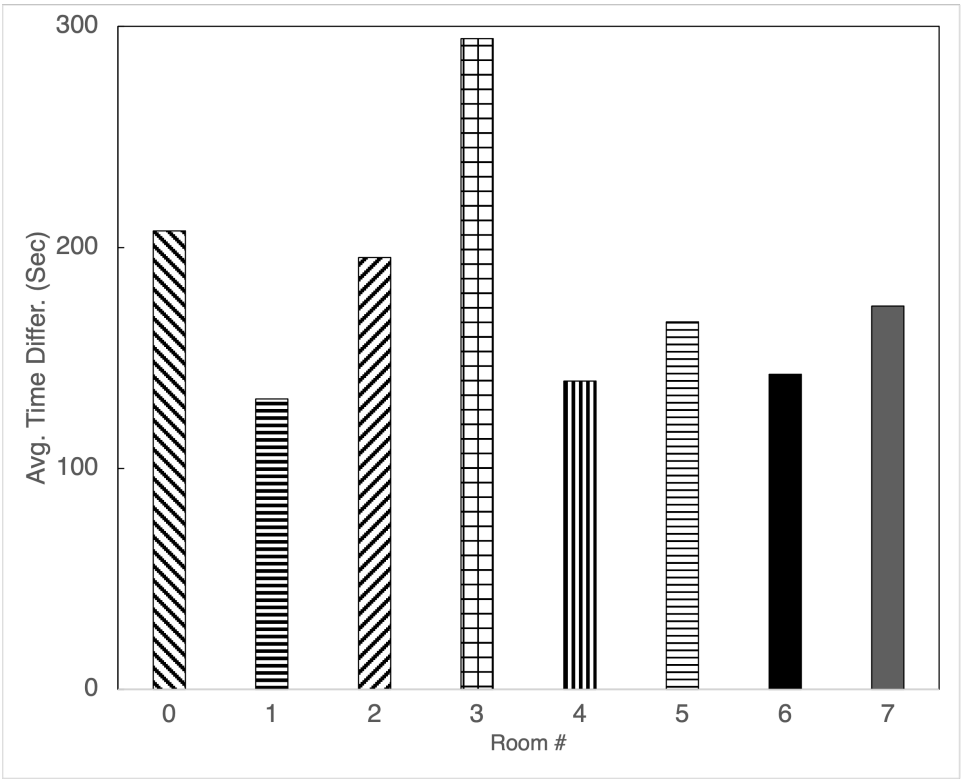

Figure 14: Time difference (accuracy) between estimated and actual amount of time needed to reach the target temperature, with for window length for 8 rooms in Building 7

Standard Error = 0.065653216. Furthermore, the true probability is two standard errors with 95% confidence. Thus, we can conclude:

Given our sample size, with 95% *confidence, the difference between the actual amount of time to reach a given temperature and the amount of time, estimated by our ILPSS estimator is no more than* 100 *s for between* 43% *and* 57% *of the rooms.*

3.4.2 Experiment 2: Regression Methods Comparison

The goal of our second experiment was to quantitatively evaluate the suitability of Multiple Linear Regression (MLR) for our solution. In our second experiment, we measured the accuracy of the two most promising candidate regression methods, MLR and LASSO, to our problem. We use the settings from the previous experiment and we ran it for the building from the first dataset only. We used our temperature change function, as discussed in Chapter 2. Moreover, starting and target temperatures, as well as the time differences, were calculated from the HiberSense dataset, as described in the example in Chapter 2. The results are depicted in Figure 15.

We ran the LASSO regression with multiple shrinkage parameter values (i.e., 0.1, 0.5, 1, 5, 10, 50, and 100). Figure 15 shows the average time difference between the estimated time to reach a certain temperature and the actual time taken to reach that temperature. Results are reported for the same 7 different window sizes (i.e., 8, 16, 32, 64, 128, 256, and 512). Clearly, for our HiberSense dataset, the accuracy of MLR and LASSO is comparable when the window size is the smallest (size 8). When the window size is larger, LASSO produced more accurate results than MLR, the deviation between the estimated time and the actual time to reach the target temperature is smaller. However, the larger the window size, the lower the estimation accuracy is for both MLR and LASSO. This is consistent with the observation in Experiment 1, that small windows containing the most recent values of sensor readings are the most relevant in affecting a current temperature. By including more sensor readings (i.e., larger window size), would reduce the accuracy of both the module estimators. Hence, using MLR with a small window size that yields accurate results and does not require any tuning parameters. It is to be noted, however, that when the shrinkage parameter for

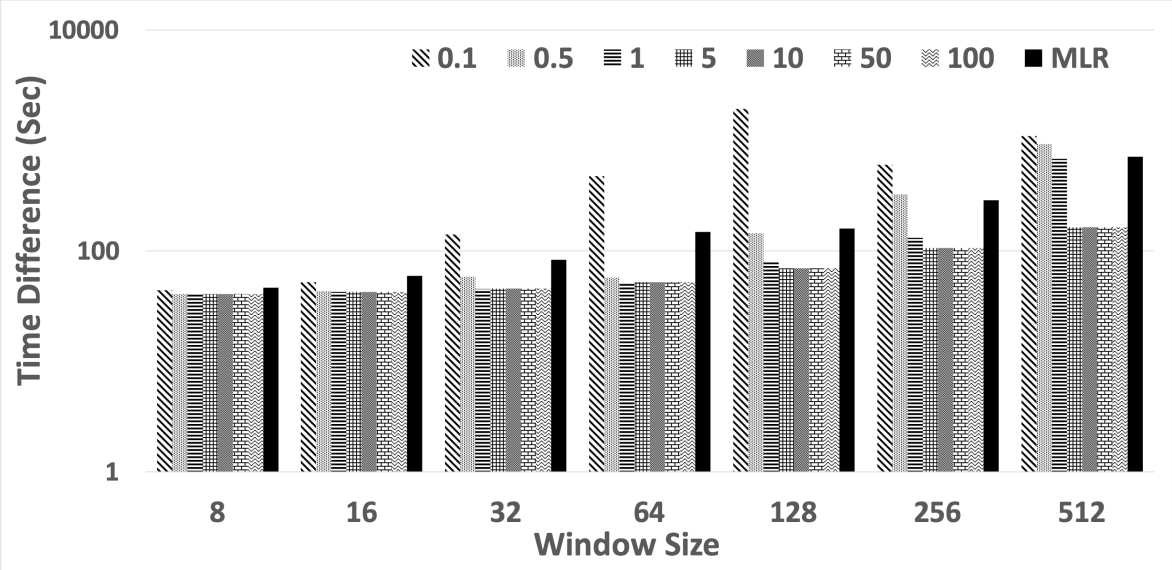

Figure 15: Average time difference between estimated and actual time to reach a temperature for MLR and LASSO with different shrinkage parameter values

LASSO is set to 1 and 5, the results for a window length of 8 are more accurate, but there are no guarantees that the same parameter value will show the same results when used on a different dataset.

3.4.3 Experiment 3: MLR vs MPR Comparison

In this experiment, we measured the estimation accuracy of our two regression techniques, namely MLR and Multiple Polynomial Regression (MPR) for each room when occupancy as a feature into the dataset. We use the experimental setup from the first experiment. The window length is set to 8 because we show that this window length produces the best results in previous experiments. Occupancy is set to ON.

The results are depicted in Figure 16. On the x-axis of the figure, we report the results for each room independently. On the y-axis is the time difference between the estimated value and the actual time for reaching the temperature difference, as discussed in the example in Chapter 2. We ran this experiment with the real dataset values as described in the

previous section. MLR and MPR show comparable results. This is because our dataset has many gaps in reporting the sensor readings (i.e., the time difference in some consecutive timestamps varies significantly—up to fifteen minutes). To assure that this is true, we ran the experiment again after fixing the timestamps and ensuring a constant short gap of 60 seconds is observed between every two consecutive readings. MPR still shows slightly higher inaccuracy compared to MLR (See room 3). This can be explained by "overfitting". When MLR is used, the resulting function g is linear, and it is calculated by minimizing the sum of the squares of the deviations from each data point to the line of the function. This is not the case with polynomial functions. Specifically, for every $n+1$ points there is a unique function of degree n that fits through the points [16]. The non-linear function g is susceptible to higher error because of its higher degree—i.e., non-linear functions typically grow faster, compared to linear functions and, thus, when the trend in the temperature changes from increasing to decreasing or vice versa, the next estimated value will have a larger difference with the actual value. Even though the MPR showed comparable results to MLR when the time gaps are fixed, MLR still showed better performance. Furthermore, we ran the same experiment for MLR without fixing the gaps between the consecutive readings—in the estimation accuracy experiment. The results for MLR with and without fixed gaps do not differ by more than a second. Thus, we conclude that the addition of the occupancy as a feature of the temperature change function improved the accuracy, using MLR. Specifically, the average value for the 4 rooms is 60 *seconds*, compared to 100 *seconds* without considering occupancy (Figure 8).

3.4.4 Experiment 4: Scalability

In our third experiment, we study how our solution scales up with an increase in the number of rooms. We measured the wall-clock time needed by our experimental testbed to run the regression from Experiment 1. For a window length of 8, we set the number of rooms to 0, 4, 8, 12, 16 20, 24, 28, and 32. Furthermore, we ran the experiment for the first building from our HiberSense dataset. We use the setup from the first experiment. Occupancy is OFF.

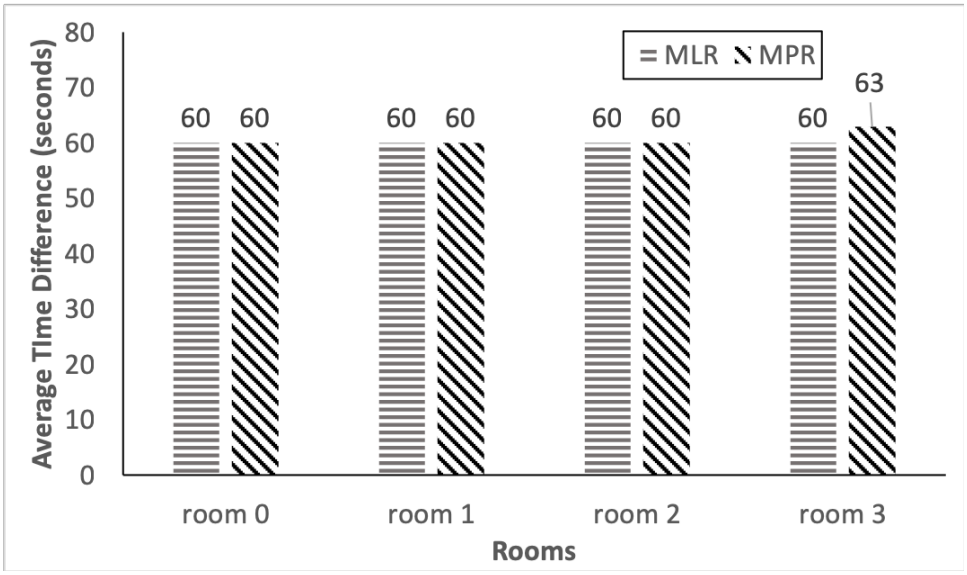

Figure 16: Time difference (accuracy) between estimated and actual amount of time needed to reach the target temperature, with MLR and MPR, with occupancy, for window length 8 for 4 rooms

The average amount of time to execute MLR grows linearly with the number of rooms (Figure 17). It takes less than 7 minutes for the solution to run the regression for 32 rooms. The number of data points that we have for each room differs between rooms, but it is 5 times on average per room. Thus, we run MLR 5000 per room, on average. Note that regression for a window of 8 can be calculated within 1 second or less ($(400/32)/5000 = 0.0025$ *seconds per window*) for up to 32 rooms simultaneously. Moreover, when our solution is run for no rooms, it still takes 60 s to run it even for 0 rooms. This makes our solution when deployed on Raspberry Pi Zero, suitable to manage space conditioning of single-family houses and small office buildings.

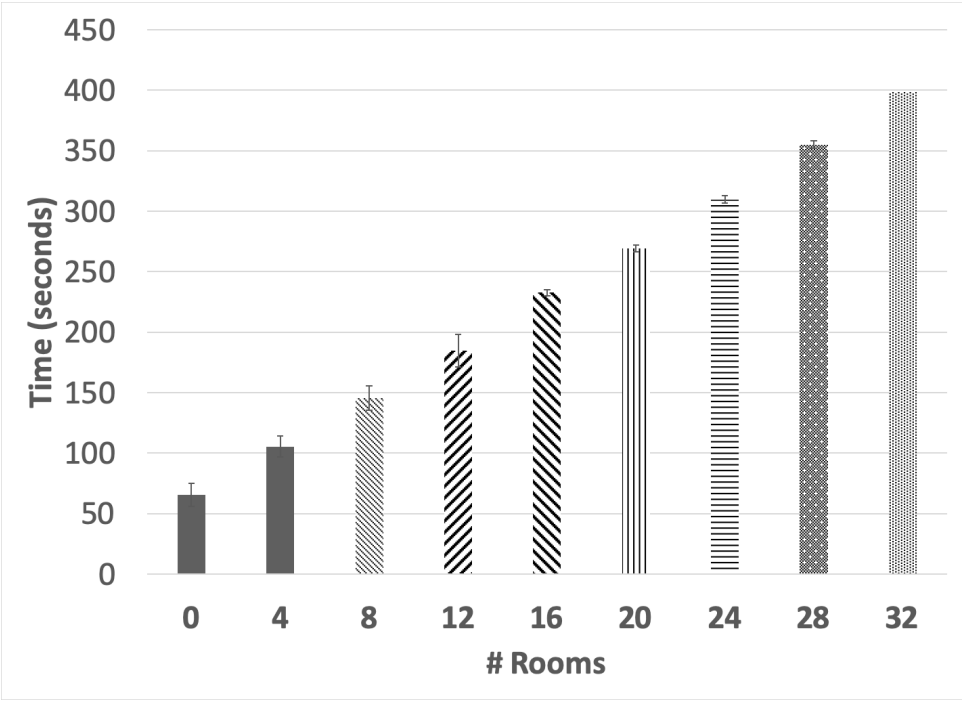

Figure 17: Average time to run MLR for different number of rooms for sliding window length 8

3.4.5 Experiment 5: Energy Savings

In this experiment, we measured the energy savings caused by our *ILPSS* solution against the commodity solutions available today (we call it *Naive* and *D-DUAL*). Specifically, users are given the option to set when the furnace should start and the target temperature of their liking, but they have no guarantees when the temperature will be achieved. The results are depicted in Figure 18. We pick two times a day from the three days of data we have, and we extracted scenarios from the dataset, i.e., the scenarios cover six of the canonical scheduling cases and the deadline for the target temperature is within 90 minutes from the moment the requests are received. Those scenarios fit the canonical scheduling cases discussed earlier. This gave us a total of six scenarios that are diverse in the nature of their requests. Moreover, we wanted to experiment with all canonical scheduling cases, thus we synthetically created five more scenarios that cover the rest of the canonical scheduling cases, which we did not extract from the data we had. Keep in mind that some scenarios cover more than one of the canonical scheduling cases. The dataset was collected during the winter in Pennsylvania and there were not many cases of requests for lowering the temperature. Furthermore, each scenario consists of four requests (one for each room), that vary in the time of arrival and the cooling/heating request.

Figure 18 shows the total time needed to regulate (i.e., cool and heat) the temperature in the four rooms. The figure is on logarithmic scale. The results show that our *ILPSS* scheduling reduces the time needed to regulate the temperature in the four rooms by up to 45% (23% on average). The six cases, namely (a), (d+f), (j), (h+k), (e+f), and (e), cover six of the canonical cases discussed earlier (see Section 2.4.2). For the other canonical cases, we synthesized data in order to evaluate our *ILPSS* solution. Furthermore, we did not want to combine many different canonical cases into one experiment, when possible. Thus, in most cases, whereby heating and cooling were needed, three rooms required temperature adjustment in the same direction and one room in the opposite. The only exceptions to this rule are cases (g) and (i), where it was more natural to warm up two rooms and cool down the other two. This explains why our solution mostly wins for only one of the temperature changes—there is no difference if a single room is scheduled with *Naive* or *ILPSS*. Moreover,

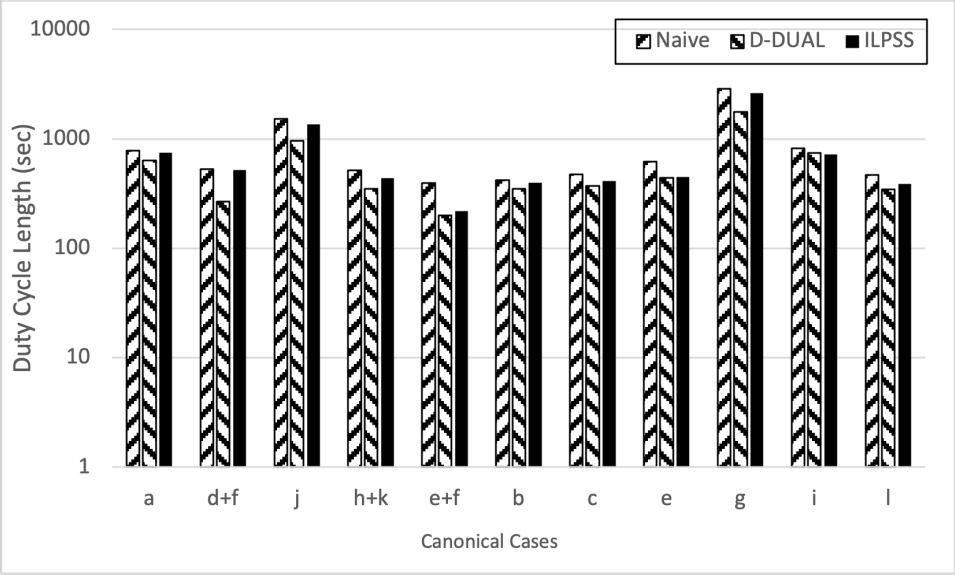

Figure 18: Total durations of HVAC operation for *ILPSS*, *D-DUAL* and *Naive*

we also studied the penalty induced on the schedule by our *ILPSS* solution—it shows by how much the deadlines were not met. For our workloads, the scheduler met all deadlines and there was no penalty.

It is to be noted that our *ILPSS* provides either fewer energy savings or is on par with D-DUAL. However, the latter does not provide any guarantees that the temperature in each room will be within the comfort zone of its users. Furthermore, even when D-DUAL is given shorter deadlines, namely at the start time of the comfort zone, it provides comparable results with the same solution when it is used with the original deadlines, i.e., deadlines that are not shortened in order to start at the beginning of the comfort zone. This shows that the produced schedule for thermal conditioning concludes early enough to not be affected by the change of deadlines. There is only one case, namely (i), whereby D-DUAL and *ILPSS* show comparable results for heating—i.e. this is the case when D-DUAL will meet the comfort zone requirements.

As part of this experiment, we also studied the performance of ILPSS with and without the optimization, using Newton's Law of Cooling. The results are depicted in Figure 19. The

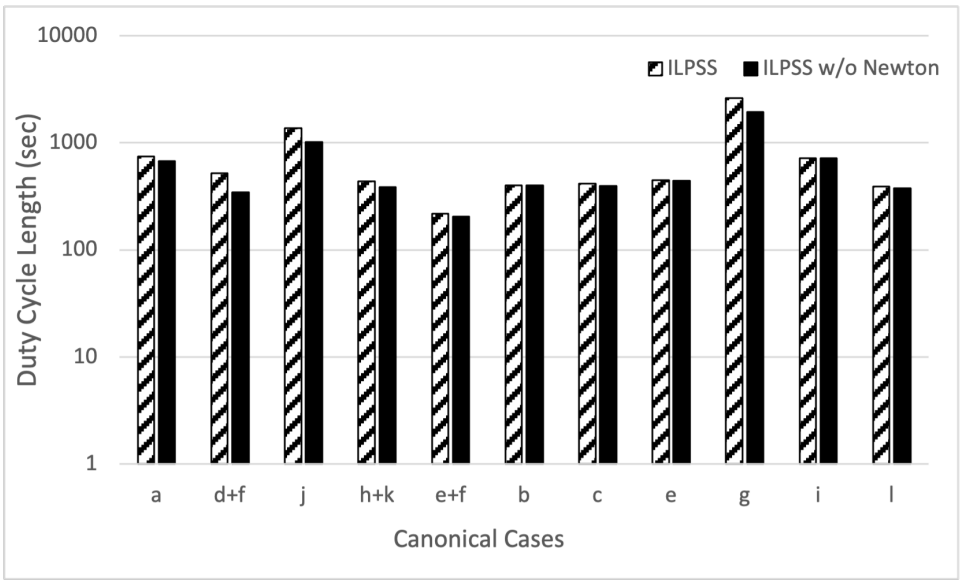

Figure 19: Total durations of HVAC operation for *ILPSS*, and *ILPSS w/o Newton*

figure is on a logarithmic scale. The version of ILPSS without the optimization consistently outperforms ILPSS with the optimization with between 1% and 25%. This is attributed to the fact that when the optimization is not leveraged, the solution does not accurately capture the adversarial effect of the environment.

3.4.6 Experiments Summary

In our first experiment, we found that the most recent sensor readings are most relevant to get an accurate estimation of time, hence, the smaller the window size the more accurate the estimation. In the second experiment, we found that MLR should be used over LASSO regression because it does not require additional tuning and shows similar results. In the third experiment, we showed that our system can be run on a real IoT smart gateway (i.e. Raspberry Pi Zero) for large residential buildings. In the fourth experiment, we found that when occupancy is taken into consideration, the accuracy of the estimator is improved. Moreover, MLR should be used over MPR since it is not susceptible to overfitting. In the fifth experiment, we found that our solution saves energy over the commodity approach and

provides guarantees that the temperature will be within the comfort zone of the users. In the last experiment, we showed that our solution saves energy.

It is to be noted, however, that *D-DUAL* is a corner case of *ILPSS*. The comfort zone in *D-DUAL* collapses to a single point - the exact time when the temperature should be reached and no tolerable deviation from the target temperature. Moreover, *D-DUAL* guarantees that the target temperature for each room will be reached between the time the request is received and the deadline by which the target temperature should be reached. But that does not ensure that the temperature will be the same at the deadline from the request. The weaker time guarantees enable higher energy savings, compared to *ILPSS*. Contrary to that, *ILPSS* provides lower energy savings, but it ensures stronger guarantees. Specifically, the target temperature will be reached at a moment in time within the user's comfort zone. For the initial evaluation of our IoT computational framework, we used *D-DUAL*. Once we demonstrated the feasibility of the framework and the solution, we enhanced our system model and that led to the development of our *ILPSS* solution.

Furthermore, our solution provides not only guarantees, but also provides flexibility to its users. They can submit requests for deadlines arbitrarily far ahead in the future. This flexibility improves the ease of use and overall positive user satisfaction. Contrary to that, the solution has to provide energy savings. Therefore, evaluating and adding requests, whose deadlines are too far ahead in the future, for scheduling will result in energy-wasting, as opposed to energy saving. *ILPSS* operates leveraging *horizons of time*, whereby the horizons are continuous time frames that start now and last for the next 6 hours. Only requests, whose deadlines are within the horizon are assessed for scheduling. Our *comfort zone* guardian ensures that the temperature in the room will be within the limitations of the comfort zone at the deadline. The horizons leverage sliding windows as opposed to tumbling or jumping windows.

When we did an analysis on our dataset, our main goal was to identify requests representing all 12 canonical scheduling cases, as depicted in Figure 6. The dataset contained data for cases (a), (d), (e), (f), (h), (j) and (k). The lack of the rest of the cases is explained by the fact that the data was collected during the winter in Pittsburgh (during the month of February). The harsh weather conditions outside do no stimulate the users to submit

requests for cooling down their rooms and all of the missing canonical cases include cooling for at least one of the rooms. Moreover, we identified that the deadlines for all requests are within 90 minutes of the submission of the requests. This inspired us to investigate further and develop the *horizons of time*, as discussed in the previous paragraph.

It is to be noted, however, that the *horizons of time* are not a limiting factor for our system. As shown in Algorithm 1, our solution recalculates the schedule every time when either new data arrives from the sensors or when a new request is received from users. An unaccounted change in context may affect negatively the accuracy of the estimator and may lead to running a schedule, which will not meet the deadlines, specified by the users. Contrary to that, our solution is self-driving—it consumes the newly received data as it arrives, re-runs the *estimator* in order to re-adjust the *temperature change function* for each of the rooms, and, therefore, improve the accuracy of the *estimator*. Subsequently, it recalculates the *scheduler*. This event-driven invocation of the algorithms provides the necessary adaptivity to meet the requirements for temperature in each room by the user-defined deadlines. Moreover, the adoption of the sliding window concept for the *horizons of time* enables our solution to quickly adopt newly received requests and adapt the schedule accordingly.

3.5 Summary

In this chapter, we presented our *ILPSS*, an IoT solution that schedules the duty cycles of HVAC systems, that considers both energy-saving and user' comfort. The solution aims at optimizing the time of operation while meeting users comfort requirements and specific requests for target temperature by a certain deadline on a per-room basis in residential buildings.

Our solution combines scheduling and two ML techniques, namely MLR and MPR, that take into consideration temperature, deadline, sensor readings, and occupancy from each room. This information is delivered to a "smart" gateway, which prepares a schedule that controls the duty cycle of the HVAC. We modeled a comfort zone, whereby the users can tolerate the temperature changes in the rooms they submitted requests for a period of time until the deadlines that are part of their requests.

Our experimental evaluation with real data showed that our approach achieves energy savings up to 49% (26% on average), compared to the baseline commodity HVAC. Furthermore, we demonstrated that our computationally cheap solution can be deployed on low-cost commodity hardware, such as Raspberry Pi Zero, and it is capable of addressing the demands for HVAC control of real-world residential buildings.

4.0 Virtual Bus Lanes: Computational Framework

In this chapter, we present a solution that establishes *on-demand, virtual bus lanes* to prioritize public transportation over other traffic and provide shortest path guidelines for other drivers, while causing insignificant delay penalties to them. Our solution called Environment Protective Traffic Orchestration (EPTrOn) leverages priority scheduling, and Dijkstra's shortest path algorithm to shape and detour traffic.

In Section 4.1 we present the background needed for our solution. We formulate the problem in Section 4.2, followed by the system model in Section 4.3. Our EPTrOn solution is presented in Section 4.4, followed by a comparison with the state-of-the-art in Section 4.5. We summarize our findings in Section 4.6. We present the results from the experimental evaluation of our solution in Chapter 5.

4.1 Background

Most internal combustion engine vehicles (ICEVs) have their engines idling when they are not in motion, i.e., when they stop at traffic lights and crosswalks, or when they are in traffic jams [18]. Furthermore, most public transit buses are equipped with diesel engines. These engines not only produce greenhouse gas emissions, but their exhaust contains a significant amount of fine particulate matter (FPM), the inhalation of which has a negative impact on human health. A plethora of diseases is attributed to FPM—asthma and lung cancer, to name a few.

The current state-of-the-art solution for pollution reduction in urban environments is EkoMark2.0 [6]. The underlying idea is to study fuel consumption by mass transit vehicles and use the information to steer cars away from congested road segments, where the buses consume a lot of fuel and thus produce a lot of pollution. The road network is modeled as a graph, whereby the streets are edges and the intersections are vertices. The fuel consumption of public transportation vehicles is calculated on a per-edge basis and the obtained

information is used to calculate eco-weights for the edges. The eco-weights are used by cars to calculate their routes. The higher the eco-weight for an edge, the less preferred it is by cars.

This solution does not take into consideration the topography of the cities as a cause for high fuel consumption by buses, i.e., buses going uphill typically consume more fuel per meter, compared to the same bus driving on a non-inclined street. Therefore, based on the fuel consumption of buses, *EkoMark2.0* assigns high-value eco-weights for cars to edges that are used by buses. These edges are less probable to get on cars' paths. Subsequently, the road infrastructure gets underutilized because these edges attract even less car traffic, without being necessarily congested in the first place. The underutilization of some edges causes higher utilization of other edges in order for the road network to handle the same amount of traffic.

Furthermore, this solution does not mitigate the negative impact of cars—calculating routes, based on eco-weights may extend the paths of cars significantly and thus have a bifold negative impact. The total pollution of cars and buses may surpass the total pollution of cars and buses that do not use eco-weights, as well as the additional travel time for cars is not mitigated.

The rapid proliferation of smart mobile devices that are equipped with positioning sensors (e.g., GPS and Galileo), and ubiquitous Internet connectivity, facilitated the growth of the near real-time traffic analysis necessary for effective solutions to traffic jams. We assume that internal combustion engine cars and buses are equipped with a mobile computing device that has a global navigation satellite system and Internet connectivity capabilities. These devices report the current location and the speed of the vehicle periodically, but *not* its destination. Leveraging on these smart mobile computing devices, our solution EPTrOn overcomes the shortcomings of the state-of-the-art work *EkoMark 2.0*.

4.2 Problem Formulation

In this section, we formulate the problem of creating dynamic bus lanes on-demand as an optimization problem. The notation, which we adopted for the rest of the chapter is summarized in Table 8.

4.2.1 Road Network, Paths & Trajectories

We model a road network as a graph in a similar manner as *EkoMark 2.0*.

Definition 11. *(Road Network) The road network of a city, including bus stops and facilities, is represented by a semantically enriched graph $G = (V, E, M)$, whereby the intersections are the vertices in V, the streets are the edges in E, and the semantic information for each vertex and each edge are vectors in M.*

A vector $m_i = (w, \gamma, sem_1, sem_2, sem_3, ...), m_i \in M$ and $i \in V \Cup E$, is of varying length with type-specific parameters: w is the weight of the edge/vertex, which represents how busy the road/intersection is, and γ is the velocity threshold for an edge, which defines when the edge is congested.

Example 1: An example of edge semantics of *the edge of 5th Avenue in New York City, right in front of the Public Library* is $m_{e_5} = \{0.342,\ 8,\ 40.753486, -73.980888, 40.752184, -73.981843,\ 1,\ 5,\ 25,\ 1,\ 1,\ o\}$; the first two numbers are the weight w and velocity threshold γ, followed by the latitude and longitude of the northern end of the segment, and the same coordinates of the southern end. The next value 1 means that the edge is one way, 5 is the number of lanes, and 25 is the speed limit in mph. The next two parameters denote the fact that there are sidewalks on each side of the edge. The last parameter o denotes the number of buses on that edge.

Example 2: An example of vector semantics at the intersection of the NYC Public Library is $m_{v_{1324}} = \{0.2412,\ o,\ 40.753486,\ -73.980888,\ 4\}$; the first element of the vector is the weight, followed by the cumulative number of buses o that approach the intersection on the edges connected to the vertex, and the coordinates of the vertex, as well as the number of edges it connects.

Table 8: Notation for Virtual Bus Lanes

G	road network graph
V	set of vertices in G
E	set of edges in G
M	set of semantic vectors in G
m_i	semantic vector, $m_i \in M$
w	weight of edge / vertex
γ	velocity threshold for edge
e	edge, $e \in E$
sem_i	element of vector m
τ	epoch of time
v	vertex, $v \in V$
VH	set of vehicles in G
vh	vehicle, $vh \in VH$
TR	set of traffic lights
tr	a traffic light at intersection, $tr \in TR$
P	set of paths in G
p	path, $p \in P$
D	set of diversions of trajectories in G
d	diversion of a trajectory, $d \in D$
I	timestamp interval of time
l	length of I
C	batch of vrp records
ir	deadline to process C

For our definition, we use an undirected graph. It is clear that a directed graph may be a more accurate model of the road networks of different cities. However, the extension from undirected to directed graph is trivial and the directed graph model does not impact our approach to solving the problem.

The terms "edge" and "road segment" for $e \in G$ will be used interchangeably. Given the above definitions for the city road network, we can formally define traffic congestion as:

Definition 12. *(Road Segment) A road segment is congested iff the average speed of the vehicles, passing through it over a given epoch of time τ, is below a specified threshold of γ miles per hour.*

Furthermore, we make a clear distinction between paths and trajectories of vehicles in our model of the city road network.

Definition 13. *(Path) A path p, from a starting point $s \in G$ to an end point $t \in G$, is a sequence of edges (road segments) connecting the points (vertices) s and t in G. P is a subgraph of G that consists of all paths p from s to t.*

Definition 14. *(Trajectory) A trajectory of a vehicle is defined as a path p in G, whereby each of its road segments is semantically enriched with one or more timestamps that show the moment(s) in time when the vehicle was on that particular road segment.*
Each trajectory has a directionality *property (i.e., from s to t) and* diversion *property that is defined as follows.*

Definition 15. *(Diversion of Trajectory) The diversion of a trajectory D is a set of points $d_i \in G$ that do not extend the trajectory by more than a given threshold of* $\max_{div}$ *miles when added to it.*
The vehicle diversion controls a car's rerouting by preventing it from diverging too far away from its initial trajectory. The directionality of a trajectory is not changed if and only if the rerouting morphs the trajectory within the set D.

The bus routes are also paths in the road network G. The buses and their trajectories (i.e., locations of these buses at different moments in time) are known in real-time, as many cities worldwide now provide this information in real-time (e.g., Busgazer [71]).

Definition 16. *(Neighborhood) A neighborhood is a subset $G' = (V', E', M')$ of G, whereby each two intersections v'_1 and v'_2 in V' are connected with an edge e' in E' and all edges from E' end in vertices in V'. Neighboring neighborhoods have edges in common, connecting them, but not vertices in common.*

The idea of the neighborhoods loosely reflects the concept of neighborhoods in cities and is inspired by the concept of *autonomous systems* in computer network routing.

4.2.2 Problem Definition

Given a road network $G(V, E, M)$, its current state as captured by the semantics M, an epoch of time τ, the set of vehicles VH, the set of traffic lights TR and the trajectories of the mass transit vehicles, calculate paths p' for the cars such that:

- they ameliorate the traffic in the way of public transportation vehicles at the next epoch of time τ_{next}, and
- they do not change the directionality nor violate diversion $\max_{div}$ of the trajectories of the ICEVs, and
- the traffic lights are dynamically adjusted to mitigate the slowdown of mass transit vehicles.

The objective of our solution is to find a subset of paths that reroute cars with minimal impact on their traveling time/distance and these routes to be communicated to the drivers. The optimization criteria are the alleviation of traffic in the way of buses effectively establishing on-demand virtual bus lanes.

4.3 System Model

We primarily focus on cars and buses with internal combustion engines. The presence of electric cars or buses does not affect our solution in minimizing congestion on the bus routes. It only reduces pollution even further.

As mentioned above, we assume that both cars and buses are equipped with mobile computing devices that report the current location and speed of the vehicle periodically. Specifically, each data report point is a tuple in the following format:

Definition 17. *(Vehicle Report Point) A data report point, of a vehicle, is called vrp:*

$$vrp(vehcID; ts; long; lat; a) \tag{4.1}$$

where $vehcID$ is a unique vehicle identifier, ts is the timestamp of the vrp, long and lat encode the geo-location of the vehicles and a is the velocity.

The timestamp captures the moment in time when the vrp was produced and is denoted in global time. The type of vehicle can be derived from the unique vehicle ID, i.e., car or bus. Consecutive tuples for a given vehicle form its trajectory in the time epoch e.
Furthermore, we provide the following definitions for our infrastructure:

Definition 18. *(Spout) Every vehicle which generates vrp tuples (mass transit bus or car) is called spout.*

Definition 19. *(Bolt) All data stream management system entities which ingest and process the spouts are called bolts.*

The terminology is adopted by many of the available data stream management systems (DSMSs) like Apache Flink [17], Heron [46], Apache Spark [97], Drizzle [90], and Merlin [87].

Moreover, we assume that each major intersection which has traffic lights is equipped with a light board that is used to display information to drivers. Some cities and highways already use such boards to provide traffic and weather updates, and details about detours and points of interest (POI).

Furthermore, there are computers that are present at each intersection. These computers act like bolts and they have a network interface, which receives the tuples *vrp* from all vehicles (spouts) that are heading towards the intersection at the current epoch. Given the geo location of these vehicles, the computer at each intersection can build the leaves of the index, which we discuss later in the section. Once the data is ingested and processed, the computer sends the summarized load of its edges to the next layer (of the index), whereby a designated computer at one of the intersections at the neighborhood accumulates all of the data and

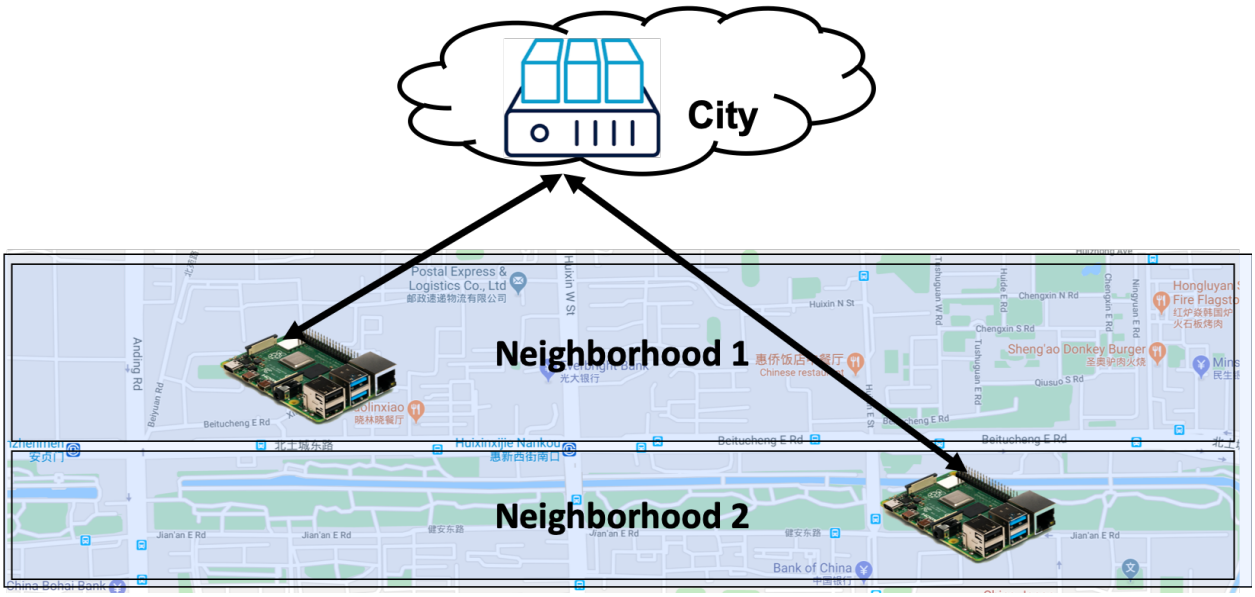

Figure 20: Monitoring System

processes it further. These computers build the next layer of the index and once they are done, they have two tasks. The first one is to send the index for the neighborhood to the computer that is in charge of building the index for the city. The second task is to push back the trees that they built back down to the computers in their neighborhood. The computer that is in charge of building the system for the city is located in the cloud. It receives the information for all neighborhoods and builds the complete index for the city. It then disseminates it back to the lower layer, namely one computer per neighborhood. Subsequently, these computers disseminate the information to all computers in their neighborhoods.

All computers send to the upper layer all of the tuples that they received by mass transit vehicles. This task is parallel to and independent from building the index. This assures that the systems that use the information from buses to estimate their arrivals to the stops on their way have up-to-date information. Moreover, when buses approach an intersection that is controlled by traffic lights, the EPTrOn solution can enable virtual bus lanes.

This approach is inspired by well-studied dynamic routing protocols in computer net-

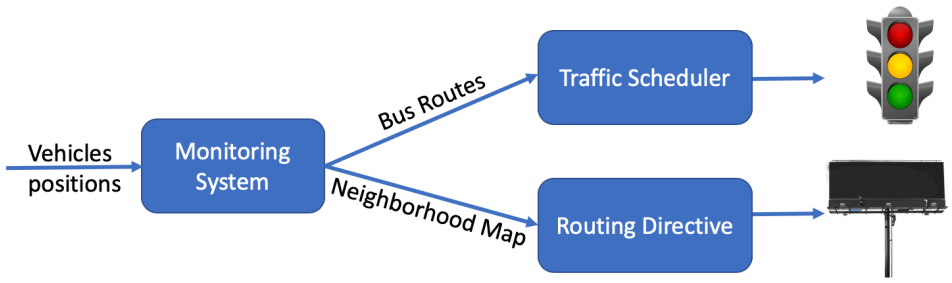

Figure 21: EPTrOn Solutions

works and the way cellular networks are designed and built. Each computer at an intersection can be considered to have a role, similar to the role of a router in an Open Shortest Path First (OSPF) [62] dynamic routing protocol enabled networks, whereby the edges are network links. All intersections in the neighborhood form what area is in OSPF. Furthermore, each intersection can be considered similar to a cell in a cellular network. Additionally, a neighborhood in our system is what a traffic/cellular/routing area is in cellular networks. All neighborhoods in a city are similar to a cellular network. Our design is synchronized with the advancements of 4G(LTE) and 5G and the effort to bring computation and content closer to the users, specifically at the network edge—at the eNodeB (for LTE) and at the 5G nodeB (for 5G). A similar layered approach is proposed in Merlin [87]. Unlike Merlin, we assume that there are no data centers at each intersection, but we assume that there is a computer. Moreover, we assume that the computer is capable of handling the amounts of data generated by the vehicles that can fit on the edges that are connected to the intersection. A diagram is depicted in Figure 20. We discuss our EPTrOn solution next.

4.4 EPTrOn Solution

Our virtual bus lanes solution is based on the hypothesis that an efficient implementation of virtual bus lanes does not delay or have a significant negative impact on the rest of

the traffic. This can be achieved by controlling traffic on the road segments, as well as controlling the traffic lights. Moreover, our solution carries out the necessary computations at the network edge and eliminates the necessity for a centralized entity that synchronizes the traffic scheduling at the intersections. Furthermore, *EPTrOn* achieves privacy by relying on drivers who know what neighborhoods of the city they have to go through in order to reach their ultimate destination from their source / current location without revealing their ultimate destination.

Our solution is depicted in Figure 21 and it consists of three logical components, specifically, *monitoring system*, *traffic scheduler,* and *routing director*. The monitoring system collects information about the traffic and maintains an up-to-date snapshot of all vehicles. The traffic scheduler adjusts the phases of the traffic lights at the intersections in order to shorten the waiting time of buses as much as possible. The routing directive generates the recommendations for drivers to be displayed on information boards, as well as handles the creation of virtual bus lanes.

4.4.1 Monitoring System

The location data (*vrp*) from cars and mass transit vehicles (buses) is produced at high velocity—a new record is generated once between every 1 and 5 seconds [96]. The *vrp* records are small—not more than 100 bytes each. For a set of one record per vehicle and 100000 vehicles, this sums up to 10 megabytes. The monitoring system receives and ingests all tuples *vrp* from both cars and mass transit vehicles within a batch C_{curr}.

Definition 20. *(Batch) A batch C is a group of tuples vrp subsequences, $vrp \in C$, over a set of data streams defined by a timestamp interval I of length l. The inter-arrival time of two consecutive batches specifies the maximum computational time for processing a batch.*

Definition 21. *(Deadline) The inter-arrival time is the delay target, or deadline ir, by which the last result can be produced while analyzing a batch. The cumulative length of ir to process a batch and the time needed to schedule the traffic lights define the duration of the epoch τ.*

The monitoring system is layered and distributed. The real-time analytical processing

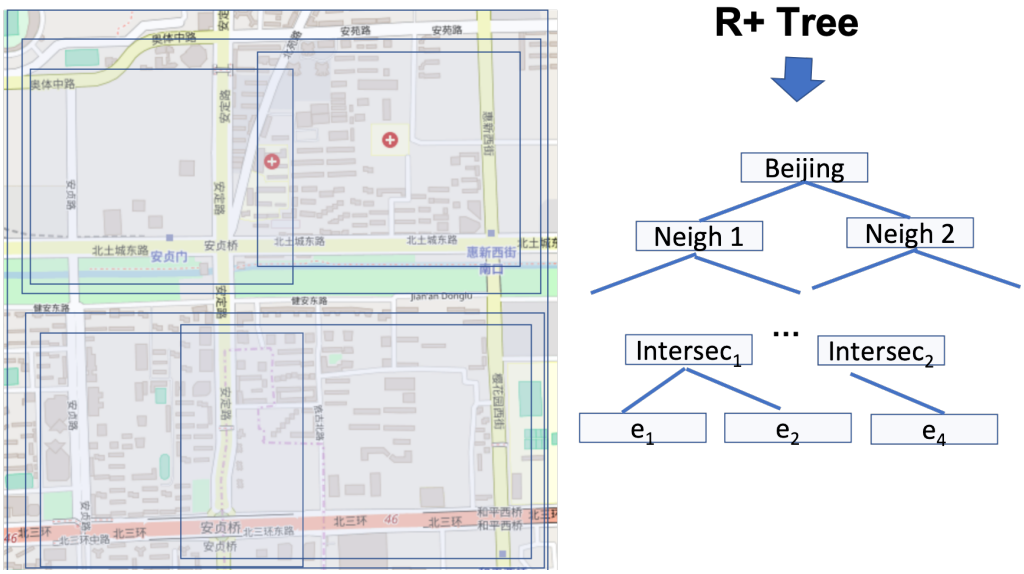

Figure 22: R^+ Tree

to identify congested road segments is performed in batches. The monitoring system uses a two-dimensional hashing R, specifically R^+ tree, and associates the position of the cars and buses with the respective edges of G. It also uses incremental sliding window aggregation techniques [79] to calculate the average speed of each vehicle and the average speed of the vehicles on each edge. Subsequently, it updates the weights w of all edges and vertices in G and updates the semantic information o about the number of buses that are located on each edge and the cumulative number of buses that approach each vertex. The city-level part of the monitoring system traverses the R^+ tree and builds a priority queue Q that contains the vertices sorted in decreasing order of the number of buses and congested edges. The pseudo-code of the algorithm is shown on Algorithm 2.

The tree is depicted in Figure 22. As discussed earlier in the section, the edges that are connected to an intersection form leaves in G. A neighborhood is the next level of the tree and a city is a top level. Furthermore, keep in mind that the neighborhoods may not be the same in size. Moreover, each neighborhood may have a number of intersections and edges that differ from the number of edges and intersections of other neighborhoods. By the

Algorithm 2 Monitoring System

Input: G, R, C

Output: Q

1: *Update the weights of all edges, based on the position of each car or bus*
2: **for each** $vrp \in C$ **do**
3: $\quad R.CarNext++(vehcID_{vrp}, lon_{vrp}, lat_{vrp}) = e_{curr}$
4: $\quad R.w_{e_{prev}} -= 1$
5: $\quad R.w_{e_{curr}} += 1$
6: **end for**
7: *Traverse the tree and place all vertices in* Q
8: $R.traverse(Q)$ R, Q

end of an interval, all $vrp \in C_{curr}$ are processed and the updated graph G and the priority queue Q are passed to the other two logical components, namely, the Traffic Scheduler, which schedules the traffic lights at each intersection, and the Routing Directive, which displays guidelines how drivers can get to the next neighborhood on their way. While the traffic lights are scheduled, the next batch C_{next} is generated and sent to the Monitoring System.

4.4.2 Dynamic Traffic Lights and Virtual Bus Lanes

A naive approach to enable virtual bus lanes is to dynamically adjust the programs of the traffic lights. We experimented with two different flavors of this approach when we were conduction our initial research on the topic. We call these single-level scheduling methods *DTL-Green* and *DTL-Red*. They are both explained later in the section. Moreover, our solution for dynamic virtual bus lanes, called *DTL-BeSPi* is two-level scheduling. We implemented it as a single-level scheduler at first, but that was proven ineffective, when experimented with—the experimental results are presented in the next chapter. Moreover, in order to overcome the requirement for centralization of our *DTL-BeSPi*, we implemented a different two-level scheduler, which uses *DTL-Green* as a springboard and combines it with

Algorithm 3 Dynamic Traffic Lights and Virtual Bus Lanes

Input: G,

Output: *Virtual Bus Lanes*

```
1: for each bus do
2:   bus.identifyNextIntersection() nIn
3:   if nIn.hasTrafficLights() then
4:     bus.notify(nIn)
5:     if nIn.onFinal(bus) then
6:       nIn.changeTrafficLights()
7:     end if
8:   end if
9: end for
```

transient detouring of the cars right ahead of mass transit vehicles in order to create virtual bus lanes. The latter relaxes the requirements to the underlying infrastructure, as it does not require costly computation. Our approach for the control of traffic lights is presented in Algorithm 3.

Each *bus* sends its *vrp* information to the next intersection it is headed to. This is done in the *notify()* procedure. Furthermore, the local computing device, as discussed in the model above, changes the phases of the traffic lights. This is completed in the *changeTrafficLights()* procedure. In the *Dynamic Traffic Lights—Red* (DTL-Red) version, the phase of the traffic lights is changed to red for all cars that go to the edge that the bus should go when it goes through the intersection. The red-light signal is kept for as long as the edge is empty, which will ensure that the bus can go in it. The approach is presented in Figure 23.

In the *DTL-Green* version, the traffic light is set to green for the lane the bus is on for as long as needed for the bus to go through the intersection. After that, the traffic lights schedule is set back to its usual schedule. The underlying intuition is to prioritize the buses over other traffic and to ensure that they can go through the intersection to the next road segment. The approach is presented in Figure 24. [1]

[1] Both of the latter two diagrams have been created with the Accident Scene Diagram Tool [1].

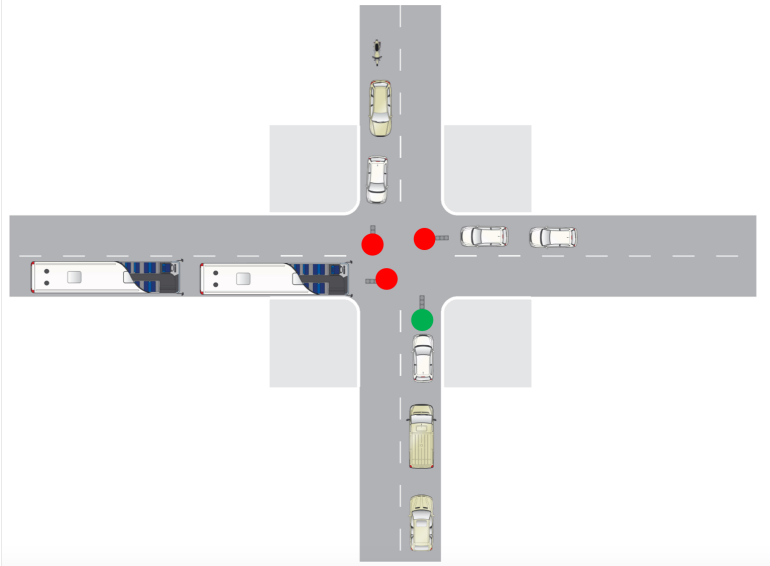

Figure 23: Dynamic Traffic Lights Control - Red

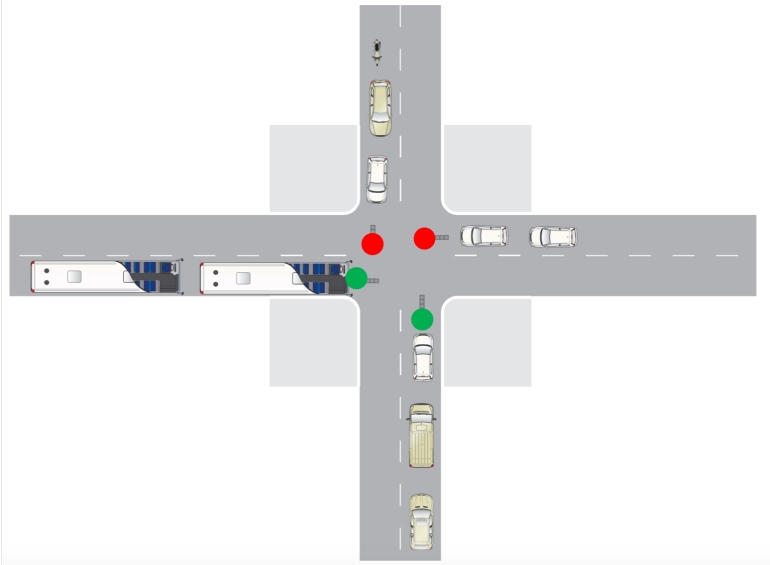

Figure 24: Dynamic Traffic Lights Control - Green

Algorithm 4 Traffic Scheduler

Input: G, R, Q

Output: $Traffic\ lights\ scheduling$

1: *Initialization:*
2: $Q = \emptyset$, $Q^I = \emptyset$, $Q^{II} = \emptyset$, $Q^{III} = \emptyset$, $Q^{IV} = \emptyset$, $Q^V = \emptyset$
3: *Place vertices to queues*
4: **for each** $v \in Q$ **do**
5: **if** $Congested(w_v)\ AND\ q_m \in Q$ **then**
6: $Q^I = Q^I \cup v$ /* vertices that connect congested segments with buses on them */
7: **else**
8: **if** $Congested(w_v)\ AND\ not(q_m \in Q)\ ANDonBusRoute(v)$ **then**
9: $Q^{II} = Q^{II} \cup v$ /* vertices that connect congested segments on bus router */
10: **else**
11: **if** $Congested(w_v)\ AND\ not(q_m \in Q)\ ANDnot(onBusRoute(v))$ **then**
12: $Q^{III} = Q^{III} \cup v$ /* vertices that connect congested segments not on bus router */
13: **else**
14: **if** $not(Congested(w_v))\ AND\ q_m \in Q$ **then**
15: $Q^{IV} = Q^{IV} \cup v$ /* vertices that connect non-congested segments with buses on them */
16: **else**
17: $Q^V = Q^V \cup v$ /* all other vertices */
18: **end if**
19: **end if**
20: **end if**
21: **end if**
22: **end for**

23: **for each** $Q_{temop} \in \{Q^I,\ Q^{II},\ Q^{III},\ Q^{IV},\ Q^V\}$ **do**

24: **for each** $v \in Q_{temop}$ **do**

25: $runTrafficScheduling(v)$

26: **end for**

27: **end for**

The *DTL-BeSPi* approach is deployed on a per-neighborhood basis and operates at two levels. At the top level, the global scheduler controls the order of processing at intersections within the current interval using five priority queues. It initiates these queues by traversing the priority queue Q received from the monitoring system, and it distributes the vertices amongst five local priority queues, Q^I to Q^V. Queue Q^I maintains the vertices that contain at least one congested road segment with buses on them, Q^{II} maintains the vertices that are ends of edges that are on bus routes and are congested but do not currently have buses, Q^{III} is the priority queue that maintains the intersections that are ends of at least one edge that is congested but is not on bus routes, Q^{IV} has the vertices that connect segments that are on bus routes but are not congested, and Q^V has the information about other vertices (i.e., connecting edges that are not on bus routes, not congested).

At each intersection, a local scheduler controls the green and red-light signal interval lengths. Our solution runs a priority scheduling, whereby the edge with the highest priority is the one with the highest number of buses on it. The interval length of the green signal is for as long as cars and buses from that edge can go through the intersection. Once the edge is empty, or no vehicles can move because the edge is saturated by a traffic jam, the next edge by the number of buses at the same intersection gets a green light signal. The operation gets repeated until all edges get a green signal, or until the interval is over. If no edges at the intersection have buses, the more congested ones get a green light first. When two edges have the same number of buses, their congestion is used as a tiebreaker to schedule one before the other. The pseudo-code for the algorithm is presented in Algorithm 4.

Our *virtual bus lanes* solution is presented in Algorithm 5. The computing device in the intersection identifies the set of cars VH_{curr} that are on the same road segment where the *bus* is. Then the intersection computing device instructs the cars that are within the virtual

Algorithm 5 EPTrOn Virtual Bus Lanes

Input: G, L

Output: $Virtual\ Bus\ Lanes$

```
1: for each bus ∈ L do
2:   bus.identifyOwnPosition() /* identify bus edge */
3:   bus.identifyNextIntersection() nIn /* identify next intersection */
4:   if nIn.hasTrafficLights() then
5:     bus.notify(nIn) /* if the intersection has traffic lights */
6:     C = nIn.identifyCars() /* get all cars that approach that intersection */
7:     for each car ∈ C do
8:       if car.inVirtualLane() then
9:         nIn.clearVirtualLane(car) /* instruct car to clear virtual bus lane */
10:      end if
11:    end for
12:    if nIn.onFinal(bus) then
13:      nIn.changeTrafficLights() /* adjust traffic lights signal */
14:    end if
15:  end if
16: end for
```

bus lane to move out of it—in the next lane when the street has more than one lane or to split the lane with the bus, if possible. This is done in the *clearVirtualLane()* primitive. In case the bus is approaching the intersection, the local computing device changes the phases of the traffic lights in order to accommodate the green signal for the bus and avoid a slowdown of the bus. This is completed in the *changeTrafficLights()* procedure. The operations for each bus affect the next intersection on its way that is equipped with traffic lights and the cars in close proximity ahead of it that are in its lane. The virtual bus lanes are created and eliminated dynamically, asynchronously between buses and the computations are carried out at the computers at each intersection.

4.4.3 Routing Directive

This logical component operates in neighborhoods and uses the well-studied Dijkstra shortest path algorithm [86] to calculate the optimal detour routes of the cars. *EPTrOn* has a two-fold gain by operating at the neighborhood level: (1) the optimal routing can be calculated for each neighborhood independently; and (2) the calculations for the optimal routing are dependent on the number of vertices, and the notion of neighborhood bounds this number to a small enough value that makes calculations possible in real-time. Hence, *EPTrOn* can efficiently provide directives/guidance for the drivers by informing them of the optimal way to the next neighborhood on the way to their ultimate destination. Typically, drivers have a trajectory in their mind that they would like to follow on their journeys from their source to their destination. The trajectory selected by a driver can follow the shortest or fastest path in their mind or can be selected for other reasons—taking into consideration driver's familiarity with the city, they may not know other possible paths. The trajectory typically goes through different neighborhoods. This is where our *EPTrOn* solution helps the drivers as it provides guidance on the light boards for the shortest paths to the surrounding neighborhoods of the current neighborhood. If a driver decides to follow the guidance, she will get her way out of the current neighborhood and into the next neighborhood, following the shortest path. This process is repeated until the driver reaches the neighborhood of their ultimate destination. Once the driver is there, they drive to their destination. It is to be noted, however, that the drivers of cars decide whether they would like to follow the guidelines or not. The system does not enforce rerouting, unlike *EkoMark2.0*. Drivers from outside the city typically use the navigation to get to their destination in the city. If these drivers decide to follow the guidelines of our system, they will only improve the traffic and experience less stress driving. The pseudo-code of the algorithm is shown in Algorithm 6.

Algorithm 6 Routing Directive

Input: G, R

Output: *Detour Guidance*

Obtain information for detours
for each $n \in N$ **do**
  for each $v \in n$ **do**
    $type = icev$
    $S_q^i = Dijkstra(G, M, type, v)$
    $type = ev$
    $S_q^e = Dijkstra(G, M, type, v)$
  end for
end for

4.5 Comparison with State-of-the-art

Some cities already implement smart traffic lights that adaptively steer the traffic in an effort to mitigate congestion [68, 80]. Another such example is the work of Smith et al. [81]. This work proposes a model, whereby the length of the phases of traffic lights at each intersection is adjusted, based on the expected amounts of traffic for each direction. Furthermore, the updated phase lengths are propagated to the downstream intersection (i.e., the subsequent intersections in each direction), so that those intersections factor in this information in their own phase length planning. The system was deployed on Center Ave. in the neighborhood of Shadyside, in Pittsburgh. We have the first-hand experience over a period of several years that Center Ave. was always more congested, as compared to the parallel to its streets—Baum blvd., Ellsworth Ave., and Fifth Ave. Either the amounts of traffic got miscalculated or some other component of the system needs fine-tuning. It is to be noted, however, that this system does not prioritize mass transit vehicles over other traffic. Several of the bus lines whose routes include that part of Center Ave. are served by articulated buses, which are much longer than the other vehicles in the traffic mix. Moreover, studies show that the additional infrastructure built does not solve the problem

with traffic jams. It only attracts new traffic and changes the scale of the problem—the phenomenon of "induced traffic" [61, 25, 94]. This suggests that *we need a balanced solution that promotes the use of public transportation while reducing the idling of both cars and buses.*

The approach taken in [93] argues that graph weights, based on a single factor, are inefficient when a traffic congestion mitigation solution is provided. The work proposes edge weights that incorporate the three main factors that influence congestion mitigation, namely edge length, road conditions, and average velocity of the vehicles on that edge. Our work uses edge weights that are based on the number of vehicles on it. We use this information further to select the optimal path, assessing the cumulative number of cars on each path. Moreover, our solution proposes dynamic bus lanes that ensure that mass transit vehicles are not slowed down by other traffic. At the same time, the road network's capacity is not decreased by permanent bus lanes.

Another concept that our solution is built upon is bi-objective optimal path selection. In [28] the proposed approach differs from previous work as it does not rely on the affine combination of the weights for distance and crime risk to amalgamate hybrid weights. Instead, the work uses the concept of skyline routes to identify a subset of paths that strike a balance between distance and crime risk. A similar bi-objective optimization problem is tackled in [66]. Our work differs from [28, 66] in getting data online and providing solutions based on the analysis of the data. Furthermore, our system provides personalized solutions for multiple actors in the traffic simultaneously rather than focusing on a single user.

Several algorithms have been proposed to update the optimal route for a given vehicle in real-time. In [51, 50], the authors propose a scheme that balances the traffic on the road network and thus mitigates traffic congestions. The scheme relies on an ad-hoc network whereby vehicles share their velocity with other vehicles and are clustered on a per-edge basis. Furthermore, the weights of the different edges of the road network graph are calculated taking into consideration average vehicle velocity, fuel consumption, and vehicle density. The work in [51, 50] does not address the prioritization of mass transit vehicles either.

The focus on reducing emissions from mass transit vehicles has been partially addressed in [6]. The work proposes a system that annotates open street map [70] road graphs with eco-weights that are based on the amount of fuel consumed by buses. The idea is extended

into a framework named *EkoMark* 2.0 that evaluates environmental impact models that are used for defining eco-weights. The framework uses 2D spatial network, GPS trajectories, and actual fuel consumption data from a case study to evaluate the models. Our solution has the bi-objective of creating on-demand virtual bus lanes and providing guidelines for optimal paths to drivers [33, 34]. Moreover, our solution has minimal to a negative impact on cars, both in terms of additional travel time and additional distance.

4.6 Summary

In this chapter, we presented our *EPTrOn* solution for the on-demand, dynamic creation of virtual bus lanes in congested urban environments. It follows a multi-fold approach, whereby priority scheduling and optimal routing techniques are interwoven to mitigate the pollution caused by internal combustion engine vehicles by clearing the way of the biggest polluters, namely the mass transit diesel engine buses. Our solution has the following advantages:

- The buses are not slowed down by other cars;
- The negative impact on cars is minimal in terms of both additional travel time and distance;
- The computations for it are carried out at the network edge on the extent possible.

In Chapter 5 we present the experimental evaluation, which confirms the advantages of our virtual bus lanes solution.

5.0 Virtual Bus Lanes: Experimental Evaluation

In this chapter, we present the results from the experimental evaluation of EPTrOn, our virtual bus lanes solution presented in the previous chapter. For our evaluation, we developed our testbed, as well as we use one of the most commonly used traffic simulators.

We first discuss the metrics in Section 5.1. The experimental testbeds are presented in Section 5.2, followed by the datasets, used for the evaluation in Section 5.3. The workloads for our simulators are presented in Section 5.4. The results from the experiments are discussed in Section 5.5. We summarize our findings in this chapter in Section 5.6.

5.1 Metrics

We evaluated the performance of the systems in terms of *lost time*, *waiting time*, *emissions*, *fuel consumption*, *speed-up*, and *detour penalty*.

Lost time: This metric shows the time lost due to vehicles driving below the ideal speed. The ideal speed is calculated for each vehicle individually. The individual speed factor, as defined in the vehicle type, slowdowns due to intersections, other participants in the traffic (vehicles and pedestrians) that are on the way of the vehicles and slow it down are considered, when the metric is calculated. Additionally, the specific restrictions of the road segments that the vehicle travels on, specifically the number of lanes, maximum speed, curves, and turns, stop signs and others are also taken into account for calculating the values of the metric. Scheduled bus stops are no counted towards lost time. We report the metric separately for buses and the rest of the vehicles. For this metric, we show both the average and the median values.

Waiting time: This metric shows the amount of time during a vehicle's trip whereby the speed of the vehicle was below 0.1 m/s. Such speed can be caused by traffic jams, waiting for a green light on a traffic light, stopping at pedestrian crossings, waiting behind other

vehicles, and others. Moreover, scheduled stops are no counted towards lost time, such as mass transit vehicle stop. We report the metric separately for buses and the rest of the vehicles. Similarly to the lost time, we report the average and the median values.

It is to be noted, however, that there is no simple algebraic relation between the two metrics. The waiting time is not equal to the difference of the lost time decreased with the ideal time. We use the metrics, as defined in [37].

Speed-up: We measure the performance of our solution by the *speed-up in terms of the number of epochs* it took for all buses to conclude their trips. An epoch is the amount of time it takes for the traffic lights of one intersection to complete one iteration over their program, i.e., the time to execute all phases of the traffic lights program.

Detour penalty: This metric shows the average detour taken by the cars (in $meters$). Our solution is bi-objective and we studied the penalty that cars had to pay in order to assure the mass transit vehicles do not get delayed.

Emissions: This metric shows the pollution generated by vehicles. We report the amounts of carbon monoxide (CO), carbon dioxide (CO_2), fine particulate matter (PM_x), and hydrocarbons (HC)—this is a common name for a group of organic compounds that consist entirely of carbon and hydrogen atoms. The hydrocarbons can be saturated (methane, ethane, propane, etc.), unsaturated—alkenes (ethene, propene, butene, etc.), alkynes (ethyne, propyne, etc.), cycloalkanes (cyclopropane, etc.), and alkadienes (propadiene, butadiene, etc.). The amounts of emissions produced for the trip of each vehicle are dependent on the type of the vehicle, the traffic conditions, as well as the route of the vehicle.

Fuel consumption: This metric shows the amount of fuel consumed by vehicles for their trips. The metric is the same for both diesel and gasoline engine vehicles. We report the metric separately for buses and the rest of the vehicles. For all emissions, we report the average and the median values.

5.2 Experimental Testbed

When we researched ways to experimentally evaluate our solution, we considered multiple traffic simulators, specifically CORSIM [2], VISSIM [57], SUMO [56], MITSIMLab [11], and DynaMIT [22]. We were interested in a microscopic simulator—one that simulates the movement to each individual car, which made DynaMIT unsuitable—it is mesoscopic (simulated vehicles are handled in small groups of homogeneous vehicles). The rest of the aforementioned simulators fail short to address the detour of vehicles using light boards, which is part of our solution. This led us to develop our *BeSPi* simulator. In order to test our virtual bus lanes in a more realistic simulation, which supports multiple car-following models, different traffic lights scheduling capabilities, and considering vehicle emissions, we carried out additional experiments in SUMO. More details about the two simulators follow.

BeSPi: Our *Beijing Simulator at Pitt* is a microscopic, event-based simulator. It is developed in C++ 11 and we tested our light boards detouring hypothesis in it. For those experiments, we used the dataset for the city of Beijing in China, which is described in the next section.

SUMO: The Simulation for Urban Mobility (SUMO) simulator [56]. SUMO is an open-source highly portable, microscopic, and continuous simulation package. It is designed by the Institute of Transportation Systems at the German Aerospace Center (DLR). SUMO is licensed under GPL. It is developed in C++ and supports a number of different application programmable interfaces (APIs) in Java, Python, Go, and others. We implemented our solution and its algorithms in Java. Furthermore, we used the TraCi Java API to connect it to the SUMO simulator [92]. We carried out the experiments using SUMO v1.2 and we used the *LuST Scenario*, which is discussed next.

Testbed: We ran the experiments on a computer with a quad-core Intel CPU, running at 3.66Gh, The computer has 16 GB of RAM memory, 256GB SSD drive, and an additional 1 TB HDD. It is running operating system Ubuntu 18.04 LTS.

5.3 Datasets

For our experimented evaluation we used two different datasets, specifically the *Beijing Dataset* [96] and the *LuST Scenario* [19]:

- *Beijing Dataset*: This dataset is collected by Microsoft Asia and covers the routes of more than 12000 taxis in Beijing. The dataset contains more than 15 million data points and covers more than 9 million kilometers (5.6 million miles) [95, 96]. Each tuple in the dataset contains the unique ID of the car that generated the tuple, the timestamp when the tuple was generated, as well as the geographical coordinates. The average distance between two consecutive data points is 600 meters (656 yards). The dataset was collected over a period of 3 months. Moreover, the dataset lacks additional details about the vehicle types.

  We downloaded the map of the city of Beijing from OpenStreetMap [70], and we converted it into a graph of vertices, using the OSMnx tool [13]. Each intersection is represented as a vertex, and the street that connects two intersections is represented as an edge. We used a subset of the city that contains 2100 vertices and 2600 edges. From the meta-data, available on OpenStreetMap, we extracted the bus routes of the public transportation. Our graph contained 74 different routes. The dataset has 127 bus stops.
- *LuST Scenario*: This dataset is the Luxembourg SUMO Traffic (LuST) Scenario [21, 20]. The scenario includes the traffic in the city of Luxembourg for a period of twenty-four hours and contains various vehicle types. It includes a topology of 155.95 square km. The total length of non-highway roads is 930.11 km. It also includes 88.79 km of highways, 561 bus stops that are used by 38 different bus lines. The number of intersections on the map is 4473 and 203 of them have traffic lights. It is to be noted, however, that for our experimental evaluation we consider that light boards are available at the intersections which have traffic lights only—this is 4.5% of all intersections. Moreover, the traffic lights are connected to 3157 inductive loops for dynamic control of the length of the green light signals. The city has 175 parking spots and 13553 buildings. The map for the scenario was created, using data from [70], similarly to our *Beijing Dataset*. The scenario has been generated with SUMO v 0.26 and it has been validated with real data.

Table 9: BeSPi Workloads Parameters

Parameter	Unit	Value
epoch length	seconds	30
car speed	km/h	35
car length	meters	4.5
bus speed	km/h	25
bus length	mmeters	13.5
bus spacing	mmeters	2000
bus delay	epoch	3
bus priority	-	0
$traffic\ distribution$	-	Gaussian, Uniform, Center, Peripheral
scale	%	30

5.4 Workloads

In this section, we discuss the workloads for each of our datasets and simulators.

5.4.1 BeSPi Parameters

As mentioned above, we developed *BeSPi* to evaluate EPTrOn's performance when both traffic lights and light boards are used. We summarize the parameters of *BeSPi* in Table 9.

Epoch length: *BeSPi* is based on a fixed-length time interval for scheduling all edges that connect to the same intersection. A device at each intersection schedules the green light length for each edge. We consider all edges to be of the same width—one lane and that all vehicles move unidirectionally. Each direction of a street is presented with a separate edge. Assuming that the majority of the intersections are on the crossing of two streets, the traffic lights run the red and green cycle for at most four different directions. At each epoch, all directions of the traffic on an intersection get scheduled. Typically, the total time is 30 *sec*.

Car speed: We calculate that on average 60 cars (15 cars per edge in a 4-edge vertex) can go through each intersection within an epoch of time by approximating the average car speed at 35 km/h at the intersection.

Car length: The typical car size at 4.5 meters long.

Bus Speed: Typically, the average speed of buses is lower, compared to the average speed of cars. We set it to 25 km/h.

Bus length: Typically buses are three times as long as a car. We set the bus length to 13.5 m.

Bus spacing: Some routes are served by more than one bus. Buses that serve the same bus routes are spaced ten minutes apart, or 4100 $meters$ (4.1 km).

Bus delay: To mimic the situation whereby some bus stops are on the side of the street to avoid blocking the traffic behind the buses when they stop, we maintain a tunable parameter *bus delay*, that specifies how many epochs will pass before buses can get back on the road after they stopped at a bus stop. The parameter is set to 3 epochs.

Bus priority: To specify bus priority in crossing an intersection during a green light, we maintain this parameter. It marks how many bus spots will be reserved on the edges of the route of the bus. This parameter takes into consideration the length of each edge. Given the length of a car and a bus, we can calculate how many cars can fit on the edge. We ignore the distance between cars when they are on the street, mimicking a stop-and-go traffic jam (keep in mind that each edge is one lane only). By reserving enough space on an edge for a bus to fit on it, we mitigate the cases, whereby a street is full and even though the bus is the first vehicle at the intersection, it cannot cross it, because the next edge on its way is fully congested. The parameter is set to 0 (i.e., no space is reserved for buses).

Traffic distribution: This parameter controls the traffic distribution with respect to the position of cars' destinations from their source location. Typically, vehicles start their trips on small neighborhood streets, go through several main streets and end the trips on small neighborhood streets. As stated earlier, cars are not required to disclose their destination. In order to simulate traffic flow or drive behavior at each intersection, the cars exhibit

traffic distributions for the different directions: West (left), NW, North (Straight), NE, East (right), and no U-turns. For example, a uniform/center traffic distribution where an equal percentage of cars go in all directions will be {02, 02, 02, 02, 02}. We assume that the predominant distribution of destinations with respect to the source of each car is a Gaussian distribution, whereby 40% of the cars go west of their current location, 10% go south, 10% go north, 20% go southwest and the other 20% go northwest, i.e., $\{0.1, 0.2, 0.4, 0.2, 0.1\}$.

In addition to Gaussian distribution, we studied three different (violation) traffic distributions, namely *uniform*, whereby an equal percentage of cars goes in all five directions $\{0.2, 0.2, 0.2, 0.2, 0.2\}$, and the two extremes: *center*, whereby all the traffic focuses in one direction $\{0.0, 0.0, 1.0, 0.0, 0.0\}$ and *peripheral* $\{0.25, 0.25, 0.0, 0.25, 0.25\}$. Comparing these three distributions to the normal Gaussian distribution, the number of violators is 40%, 60%, and 80%, respectively. For example, for the case of uniform distribution, 40% of the cars should have a destination in the west, but it is only 20%, who have it, thus we have 20% violators in that direction only. The other 20% come from the drivers, whose destinations are north or south. The number of violators sums up to 40%.

Scale: This parameter controls the amount of traffic in the road network. Each road segment has a length and using the car length parameter, we calculated the number of cars that fit on it. The scale is calculated as a percentage of the capacity the road network has. We set it to 30%.

5.4.2 SUMO Parameters

The details about the used parameters are explained as follows:

Vehicle type: In the LuST Scenario, there are six classes of passenger vehicles: sedan, wagon, van, delivery vehicle, and two types of hatchbacks. The six types of vehicles are mapped to six virtual types of vehicles, namely "passenger1", "passenger2a", "passenger2b", "passenger3", "passenger4", and "passenger5". All six types of vehicles are equipped with gasoline engines that meet Euro 4 emissions standards, as specified in HBEFA, namely "PC_G_EU4" [67]. There is only one type of buses and it is based on vehicle type "Bus", as defined in HBEFA. It is an average urban bus of all fuel types. Moreover, the HBEFA

types are customized for the case of Luxembourg with the following additional parameters: maximum speed, color, acceleration, deceleration, sigma, length, minimum gap, speed deviation and shape for the GUI version of the simulator. The description of the parameters is summarized in Table 10.

SpeedDev: All classes of vehicles have the same values set for the *speedDev* parameter, namely 0.1 $meters\ per\ second(m/s)$.

MinGap: The minimum gap distance that they maintain to the car they follow differs, whereas the minimum is 1.0 m for of class "passenger2b" and the maximum is for buses, 3 m.

maxSpeed: The maximum speed *maxSpeed* is measured in m/s and the values vary between 30 and 70 for the different classes of vehicles. The bus and the delivery vehicles ($passenger5$) have the lowest maximum speed, 30 m/s, which translates to 108 km/h.

Vehicle distribution: The scenario is built on statistical data from the city of Luxembourg and the proportion of the vehicles is built upon them. Specifically, 40% of the vehicles are sedans, each of the two hatchback classes has 20% of the total number of vehicles in the city of Luxembourg, 10% of the cars are wagons and delivery vehicles and vans are 5% each. The values used in the models are summarized in Table 11 and Table 12.

The scenario includes 2240 diesel buses. Furthermore, there are two types of vehicles with respect to where they start and complete their trips. Those who start and complete their trips in the city are called "local mobility" vehicles and there are 215526 of them in the scenario. The vehicles, which pass through the city, but their trips start or end outside the city are called "transit mobility" and there are 70484 of them. The distribution of vehicles based on their vehicle type is summarized in Table 13.

Car-following models: The car-following models are a tenet of microscopic traffic simulators. They define the parameters that all vehicles in the simulation obey in order to avoid collisions and deadlocks. Such parameters are the distance to the followed car, the frequency of adjusting the speed of a vehicle, the frequency of observing the surrounding environment, and others. For our evaluation, we experimented with 7 different car-following models, namely *Krauss, IDM, IDMM, ACC, CACC, BKerner*. We wanted to confirm our

Table 10: SUMO Vehicle Type Parameters

Parameter	**Description**
vClass	abstract vehicle class, as defined in SUMO
id	unique identifier of the class
color	RGB color for the vehicles of this class
accel	acceleration ability of vehicles of this type
decel	deceleration ability of vehicles of this type
sigma	driver imperfection factor
length	vehicle's nett length
minGap	length of the empty space to the followed car
maxSpeed	maximum velocity of vehicles of this type
probability	the probability to insert new vehicle into the simulation each second
speedDev	deviation from the speed factor
guiShape	shape to be used for the GUI version of SUMO

Table 11: Vehicle Types Definition

Id	**vClass**	**color**	**accel**	**decel**	**sigma**	**length**	**minGap**
*passenger*1	passenger	.8,.2,.2	2.6	4.5	0.5	5.0	1.5
*passenger*2*a*	passenger	.8,.8,.8	3.0	4.5	0.5	4.5	1.5
*passenger*2*b*	passenger	.2,.2,.8	2.8	4.5	0.5	4.5	1.0
*passenger*3	passenger	.3,.3,.3	2.7	4.5	0.5	6.0	1.5
*passenger*4	passenger	.9,.9,.9	2.4	4.5	0.5	5.5	1.5
*passenger*5	passenger	8,.8,.0	2.3	4.5	0.5	7.0	2.5
bus	bus	-	2.6	4.5	0.5	12.0	3

Table 12: Vehicle Types Definition

Id	**maxSpeed**	**probability**	**speedDev**	**guiShape**
*passenger*1	70	0.4	0.1	passenger/sedan
*passenger*2*a*	50	0.2	0.1	passenger/hatchback
*passenger*2*b*	50	0.2	0.1	passenger/hatchback
*passenger*3	70	0.1	0.1	passenger/wagon
*passenger*4	30	0.05	0.1	passenger/van
*passenger*5	30	0.05	0.1	delivery
bus	30	-	0.1	bus

Table 13: Number of Vehicles per Vehicle Type

Id	**Local**	**Transit**
*passenger*1	86447	27981
*passenger*2*a*	43244	14023
*passenger*2*b*	42800	14241
*passenger*3	21559	7157
*passenger*4	10798	3560
*passenger*5	10678	3522
bus	2240	-

hypothesis that our solution works for all of the most widely adopted car-following models.

Gipps is one of the simplest car-following models [30] and some of the models we experimented with are based on it. Thus, we will start the discussion with it. The model defines the maximum acceleration a vehicle will need at a given point in time in order to overtake the vehicle in front of it. The parameters used are only the distance between the two vehicles and the difference in their current velocities. This model is not realistic and is not implemented in SUMO.

Krauss is the original collision-free car-following model that was developed for SUMO. It is also the default car-following model for the simulator. It was proposed in 1998 [45]. Krauss is a safe distance model and it is a variant of the Gipps model. Furthermore, it is a stochastic model and all the parameters are optimized similarly to the Gipps model, specifically response time, braking rate, and maximum desired speed [76]. The speed of a vehicle at each time step is calculated using the measured speed of the followed vehicle in the previous time step. Moreover, the speed is calculated with respect to the vehicle acceleration, deceleration, length, driver's imperfection in holding the desired speed, and the minimum desired net distance to the followed vehicle.

IDM stands for "intelligent driver model" and it is a time-continuous car-following model [89]. It is applicable both to highway traffic, as well as to urban traffic. *IDM* overcomes the realistic properties that the Gipps model lost being deterministic. The desired velocity of a vehicle is calculated, using the minimum gap to the followed vehicle, the desired time gap to the followed vehicle, the vehicle acceleration and deceleration ratios.

IDMM is an enhanced version of the "intelligent driver model" [89, 88]. It is based on the observation that the driver's behavior changes when the traffic conditions change. The model introduces a new variable, specifically "subjective level of service" and its value for each driver is calculated as the exponential moving average of the instantaneous level of service experienced in the past. Moreover, the instantaneous level of service is calculated as a function of the actual velocity of the vehicle. While *IDMM* is a more realistic car-following model, compared to *IDM*, the theoretical comparison calculations show that *IDMM* has a lower theoretical flow of vehicles compared to *IDM*.

The "adaptive cruise control" (ACC) is one of the advanced driver assistance systems

available to drivers [75]. The *ACC* system uses radar, lidar, and/or cameras to measure the distance and the relative velocity to the followed vehicle. Moreover, the *ACC* system engages the actuators to accelerate the vehicle (throttle) or slow it down (breaks), depending on the measured distance and velocity of the followed vehicle. In the cases of lack of a vehicle to be followed, the *ACC* system maintains a preset by the driver speed. The acceleration in the gap control mode is modeled as a second-order transfer function, based on the gap and speed deviations. The gap deviation is calculated using the current position of the followed vehicle, the position of the vehicle, and the desired time gap to the followed vehicle. The gap closing control mode is triggered when the distance to the followed vehicle is less than 100 meters [58, 55, 53, 39, 54, 52, 14].

CACC stands for "cooperative adaptive cruise control" and it is an enhanced version of the *ACC* model [75]. The speed control mode is an enhanced version of the *ACC* model and it is activated whenever the time gap between the vehicles is at least 2 seconds. The speed of the vehicles is modeled in the gap control mode as a first-order transfer function, based on the gap and speed deviation. The gap closing control mode is triggered when the time gap becomes less than 1.5 seconds [58, 55, 53, 39, 54, 52, 14].

BKerner is a car-following model developed by Boris Kerner [42, 44, 43]. Kerner's three phases of traffic theory are used as a springboard for the model. The fundamental diagram of traffic flow has two phases: "free flow" and "congested traffic". Kerner splits the congested traffic phase further into two phases, namely "synchronized flow" and "wide moving jam". The speed is calculated similarly to the *IDM* model. The difference is that drivers maintain a gap between the minimum safe gap and a "synchronized space gap" that is dynamically calculated during the "synchronized flow" phase. When the gap between a vehicle is more than the "synchronized space gap", the driver accelerates. On the other hand, when the gap is smaller than the minimum safe gap, the vehicle decelerates.

Time-to-teleport: This parameter defines the amount of time a vehicle will spend stuck at before the simulator teleports to the next edge on the vehicle's path. The underlying idea is to avoid deadlocks of the simulation. In reality, the vehicles can significantly decrease their following models and avoid deadlock situations. This is not the case for simulators. The *LuST* scenario set the time-to-teleport parameter to 600 seconds.

Traffic orchestration: This parameter defines the overall traffic orchestration solution, used for the experiments. It supports three options, namely *Naive, EPTrOn and EPTrOn-DTL*. The way traffic is orchestrated traditionally is what we refer to as *Naive*. It is the default orchestration for the Lust scenario. Furthermore, *EPTrOn* is our virtual bus lanes solution and *EPTrOn-DTL* is the case whereby the traffic lights are controlled, using the *DTL-Red* and *DTL-Green* approaches.

Scale: This parameters control the amount of traffic as a percentage of the original *LuST* scenario. The values that we experimented with are 30%, 60%, 90%, 100%, and 200%.

Rerouting devices: This parameter controls the percentage of vehicles that are equipped with rerouting devices. This parameter is used in conjunction with the DUE user assignment. Only vehicles that are equipped with rerouting devices can update on the shortest path to their destination, taking into consideration the current traffic in the road network. Mass transit vehicles are excluded from the set of vehicles that are equipped with rerouting devices. The *LuST* scenario set this value to 70%.

Ignore junction blocker: This parameter controls the amount of time a vehicle can spend at an intersection before it gets teleported to the next edge on its path. The parameter prevents the creation of deadlocks in the simulations.

User assignment: SUMO supports two different types of user assignments, namely *Dynamic User Assignment* (DUA) and *Dynamic User Equilibrium* (DUE). When DUA is used, the shortest path between the starting and the ending point is calculated in an "empty" road network, i.e., there are no other vehicles in the road network. Moreover, the shortest path is calculated with respect to time, rather than distance. DUE is used for cars that are equipped with rerouting devices. The shortest path for these cars is recalculated when they are in the simulation, taking into consideration the current snapshot of the traffic in the road network. Furthermore, the paths are recalculated for the shortest traveling time.

Traffic lights schedule: SUMO supports two different traffic lights programs, namely *Static* and *Actuated*. When *Static* programs are used, the traffic lights have predefined programs, i.e., all phases of the traffic lights programs are statically defined, and they do not get changed over time. The *Actuated* programs use road detectors in order to dynamically

adjust the lengths of the different phases of the traffic lights. The road detectors are installed on each lane at the intersections that are controlled by traffic lights. They use electric induction to count the number of cars that go through the lane per unit of time. Additionally, they calculate the average time between two consecutive vehicles. Thus, when a lane becomes less busy, as detected by the detector, the computer of the traffic lights can change the signal for that lane to red.

For all experiments, conducted in the SUMO simulator, we use combinations of *user assignments* and *traffic lights schedule*. For the rest of the chapter, we adopt the following notation:

- dua.static (DUA ST)—this is a combination of dynamic user assignment and static traffic lights schedules;
- due.static (DUE ST)—this is a combination of dynamic user equilibrium and static traffic lights schedules;
- dua.actuated (DUA ACT)—this is a combination of dynamic user assignment and actuated traffic lights schedules;
- due.actuated (DUE ACT)—this is a combination of dynamic user equilibrium and actuated traffic lights schedules.

The values for all parameters in our experiments are summarized in Table 14.

5.5 Experiments

In this subsection, we present the results of the experimental evaluation of our virtual bus lanes approach. We use SUMO to answer the following questions:

- Is our EPTrOn solution applicable to realistic scenarios;
- Is EPTrOn effective when light boards are not used to steer traffic;
- How much pollution reduction do we achieve using EPTrOn;
- Does EPTrOn work with the most widely used car-following models;
- Is EPTroN more efficient, compared to dedicated bus lanes.

Table 14: SUMO Workloads Parameters

Parameter	**Values**
user assignment	DUA, DUE
traffic lights schedule	Static, Actuated
car following model	Krauss, IDM, IDMM, ACC, CACC, BKerner, PWagner2009
$time - to - teleport$	600
traffic orchestration	Naive, EPTrOn, EPTrOn-DTL
scale	30%, 60%, 90%, 100% and 200%
rerouting device	70%
ignore junction blocker	20

We use BeSPi to answer the following questions:

- How effective is EPTrOn;
- How does the usage of light boards for steering traffic mitigate the negative impact on the on-time performance of mass transit vehicles.

For all experiments, which are run in the SUMO simulator, we adopted a uniform notation. The notation is summarized in Table 15.

5.5.1 Experiment 1: On-Time Performance Improvements and Atmospheric Pollution

In this experiment, we will evaluate experimentally our hypothesis that the novel scheduling scheme that we propose whereby virtual bus lanes are created on-demand actually improves the on-time performance of buses. We run the experiment in SUMO. For this experiment we set up the simulator to run the experiment for the 4 different options that are available in the Luxembourg scenario—namely DUA ST, DUE ST, DUA ACT, DUE ACT. Moreover, 70% of the vehicles are equipped with rerouting devices and the rerouting is recalculated every 5 minutes. The car-following model is set to *Krauss*. The ignore junction

Table 15: SUMO Experiments Notation

Abbreviation	**Description**
Naive DUA ST	no virtual bus lanes, DUA ST
Naive DUE ST	no virtual bus lanes, DUE ST
Naive DUA ACT	no virtual bus lanes, DUA ACT
Naive DUE ACT	no virtual bus lanes, DUE ACT
EPTrOn DUA ST	virtual bus lanes, DUA ST
EPTrOn DUE ST	virtual bus lanes, DUE ST
EPTrOn DUA ACT	virtual bus lanes, DUA ACT
EPTrOn DUE ACT	virtual bus lanes, DUE ACT

blocker parameter is set to 20 seconds and the time to teleport vehicles is set to 10 minutes. Furthermore, the maximum departure delay for vehicles is set to 10 minutes. We wanted to experiment with a realistic scenario. The selected values are the default values for the LuST Scenario. The varying parameters are our novel EPTrOn approach and the currently wildly spread solution (i.e., naive). The parameters are summarized in Table 16.

Lost time: The results for the lost time are depicted in Figure 25. The base case line is *Naive* and the best case is no lost time. Our EPTrOn solution consistently outperforms the naive approach, with respect to buses, shortening the lost time to between 38% and 53% of the same metric for naive on average. The best performing scenario is DUE ST, whereby EPTrOn buses exhibit only 38% of the lost time on average, compared to naive. Moreover, the values for DUA ST, DUA ACT, and DUE ACT are 53%, 53%, and 47%, respectively. This behavior is consistent for both the average and the median values. The values for the median case are 50%, 43%, 50%, and 42% for DUA ST, DUE ST, DUA ACT, and DUE ST, respectively. Furthermore, the metric, with respect to cars, does not show a significant increase in the lost time—it is less than 4% for all cases and it is consistent for both average and median cases. It is to be noted, however, that the case for DUE ST Naive shows significantly higher time lost for both buses and cars, with or without the use

Table 16: SUMO Experiments Notation

Parameter	**Unit**	**Value**
Traffic lights programs	-	Static, Actuated
User Assignment	-	DUA, DUE
Rerouting device	%	70
Junction blocker time	seconds	20
Teleport time	minutes	10
Max dep. delay	minutes	10
Car following model	-	Krauss

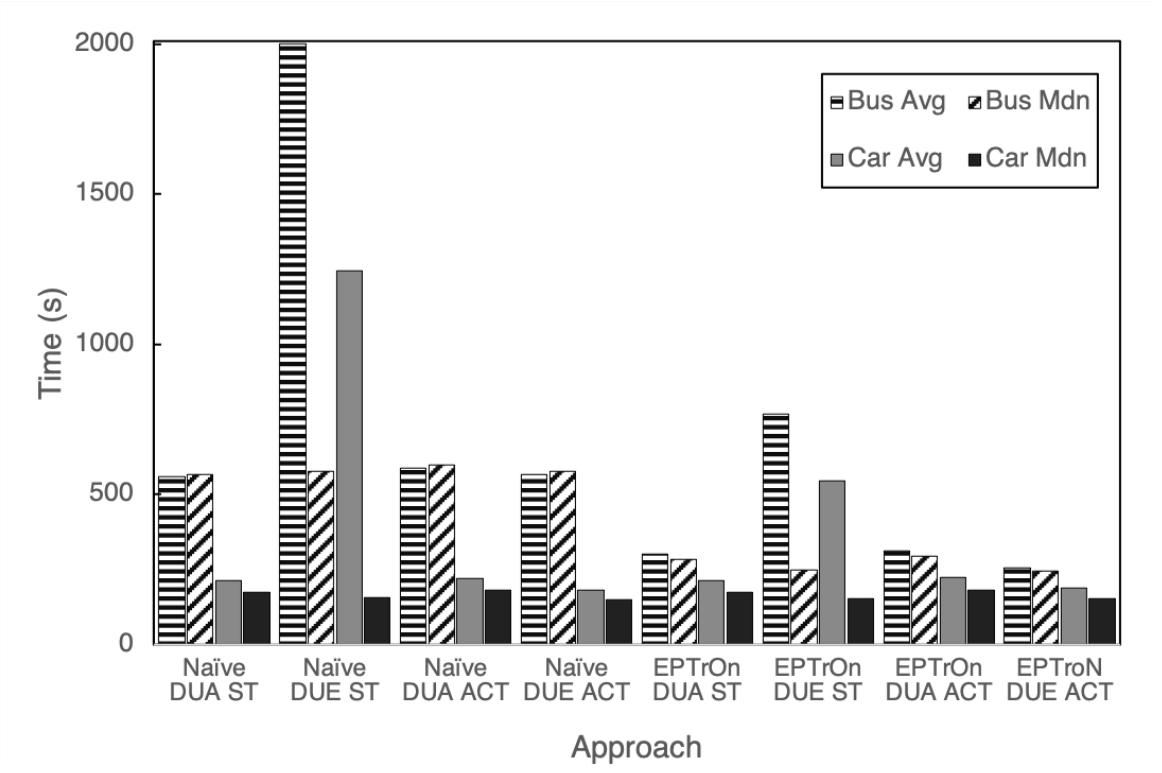

Figure 25: Time Lost for Naive and EPTrOn

of virtual bus lanes, compared to the other cases of the scenario. These results show that using blindly navigation as a way to avoid traffic jams and having infrastructure that relies on static traffic lights schedules, increases the time spent in the car rather than decreasing it. Moreover, there is a significant difference between the average and the median values for both buses and cars. This is caused by the vehicles that run in rush hours—they are subject to higher significant delays, compared to the vehicles during the rest of the day.

Waiting time: The results for the time waiting are depicted in Figure 26. Similar to the other metric, EPTrOn consistently outperforms the Naive approach, with respect to buses, shortening the waiting time to between 25% and 36.3% on average. The biggest gain is for the case, whereby actuated traffic lights and dynamic user equilibrium are used, namely DUE ACT. The smallest gain is for DUA ST. The values for DUE ST and DUA ACT are 33% and 36.1%, respectively. The waiting time for cars does not get increased by more than 5%. The exhibited behavior is consistent for both the average and the median values. Furthermore, similarly to the lost time, the case of dynamic vehicle allocation equilibrium (DUE) and static schedules of the traffic lights show significantly higher values, for both Naive and EPTrOn. The exhibited behavior reconfirms the earlier observation that using the navigation, in this case has a negative impact on travel times in rush hours. Additionally, as seen for the other metric too, when EPTrOn is used in conjunction with DUE Static, the waiting time for cars is cut by 50%. By prioritizing the buses in this case, the cars' travel time is shortened too. The latter reconfirms the reality that both cases are valid—buses slow down cars and cars slow down buses.

For this experiment, we present the atmospheric pollution of one gas (of the family of gases) per plot as we do not want to clutter the graphs. The results for the CO emissions are depicted in Figure 27. Our novel EPTrOn solution consistently outperforms the Naive approach. The decrease in the emitted carbon oxide between 20% and 50%. Typically engines are idling when vehicles are not moving—i.e., they are run at low RPM and thus consume less fuel. This explains why the proportional decrease of CO emissions is lower, compared to the decrease for time lost and time waiting, discussed in Experiment 1. It is to be noted that for the case of DUE Static without virtual bus lanes (i.e., Naive), the average amount of CO emissions for cars was 200 g, but we changed the maximum value on the scale

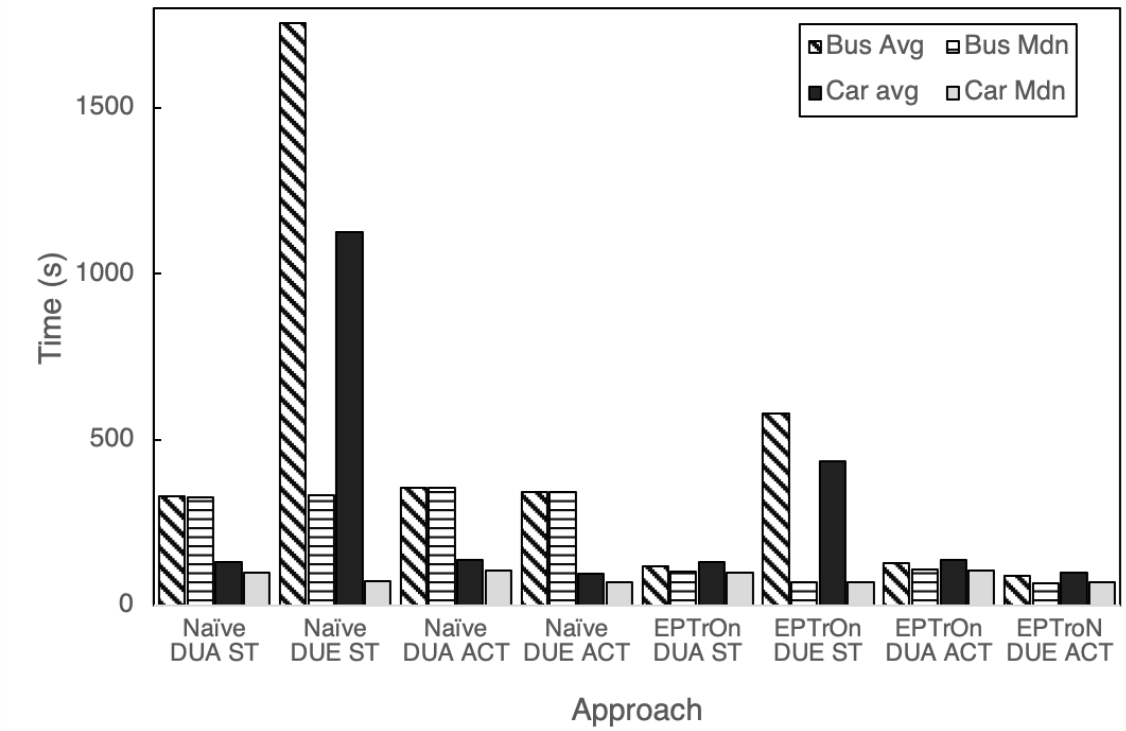

Figure 26: Waiting Time for Naive and EPTrOn

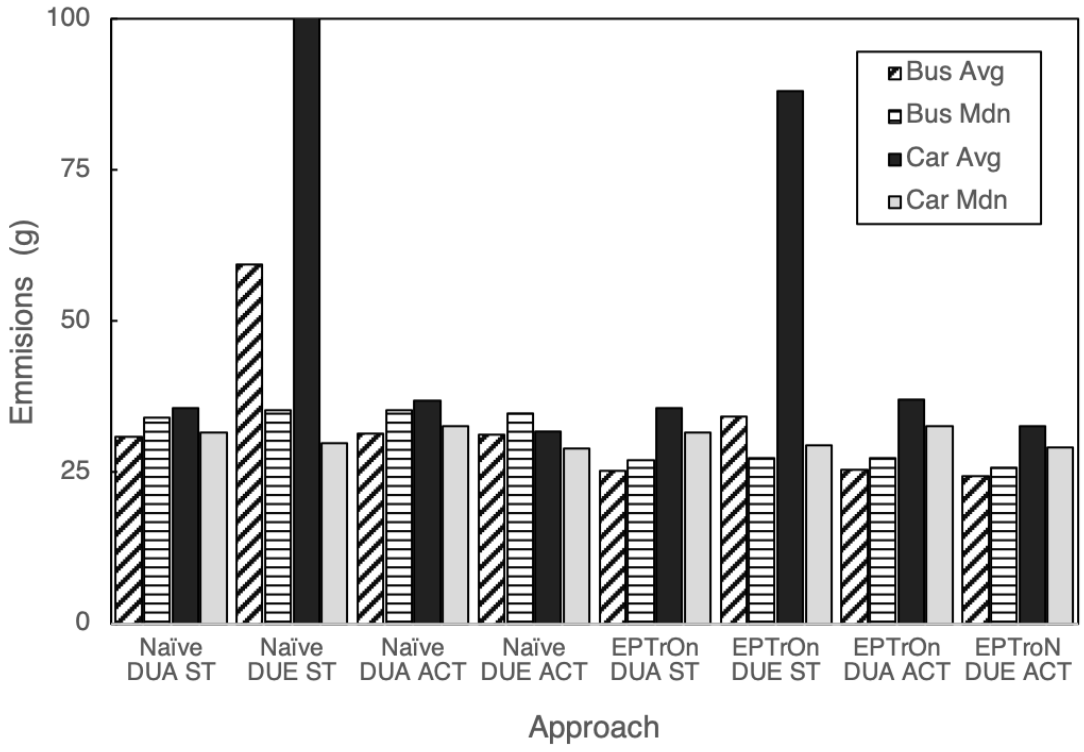

Figure 27: Emissions CO for Naive and EPTrOn

to 100 in order to make the values for the other cases more visible. Given the median value of 29.6 g for the same case, it is to be noted that a small number of outliers skewed the average value. This confirms once again, the hypothesis that using the navigation in road networks, whereby the traffic lights have fixed schedules, increases the travel time, instead of decreasing it.

CO2 emissions: The results for CO2 emissions are depicted in Figure 28. We changed the unit to kg because the buses are big polluters as typically, they produce almost a kilogram of CO2 emissions per kilometer traveled. Our novel EPTrOn solution consistently outperforms the Naive approach. The savings in produced CO2 emissions are between 15% and 30%. Similar to the case of CO emissions, the pollution savings are not proportional to the time savings. This is attributed to the fact that when the buses are not moving, their engines are idling and thus produce fewer emissions, compared to when they are in motion. It is to be noted, however, that the median values are slightly higher than the averages. This shows that the data is skewed for the buses. Contrary to that, the average and median values for the cars are equal for both approaches—with and without virtual bus lanes. This shows that cars have a normal distribution of CO2 pollution. Taking into consideration the phenomenon of high CO emissions for cars in the case of the Naive approach for DUE Static, this means the cars often accelerated - this is the case when fuel is not burned completely and CO is emitted instead of CO2, which happens when the engine is not forced. Furthermore, cars produce an order of magnitude gramfewer CO2 emissions than buses (N.B. cars' travel distance is 80% of buses travel distance on average). For the cases whereby the objective is the reduction of CO2 emissions, in particular, buses should be used only if they carry at least 10 *times* more passengers, on average.

HC emissions: The results for HC emissions are depicted in Figure 29. The results are plotted on a logarithmic scale as there is an order of magnitude difference between the amounts of emissions produced by cars and buses. When EPTrOn is used, buses emit between 20% and 44% fewer HC emissions. Moreover, the average and the median values do not differ significantly—the HC emissions are generated mostly when the vehicles are moving. Furthermore, the average and the median values for the HC emissions, produced by cars, do not differ. Given that the waiting and lost times of cars do not differ significantly

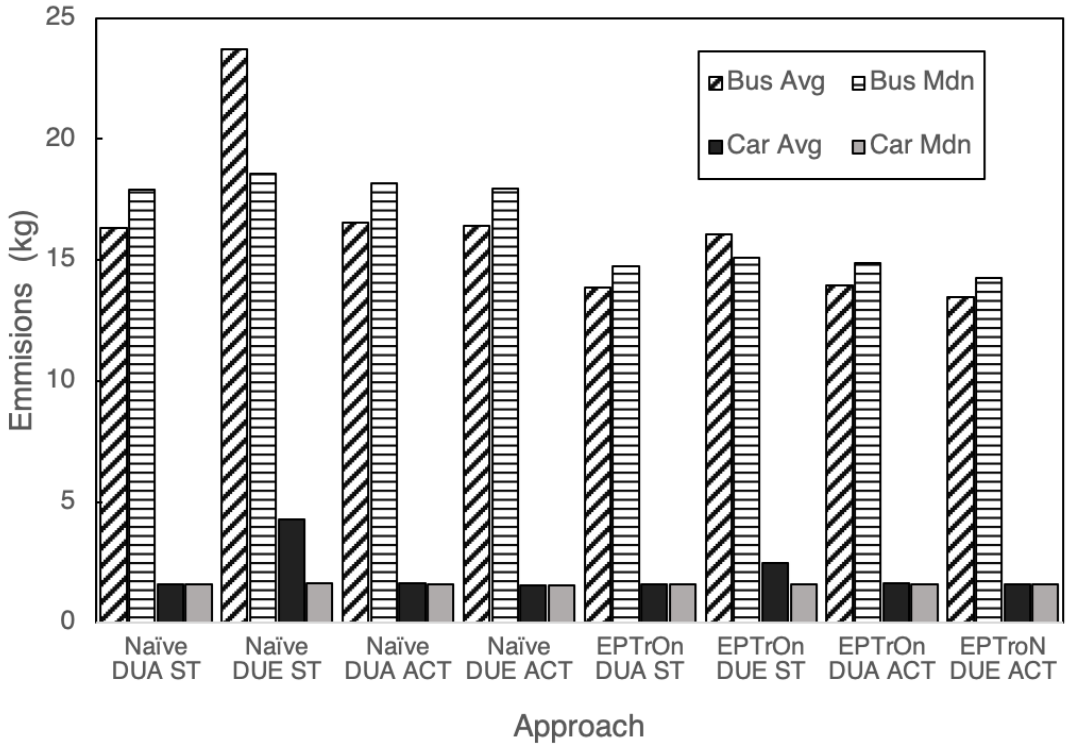

Figure 28: Emissions CO2 for Naive and EPTrOn

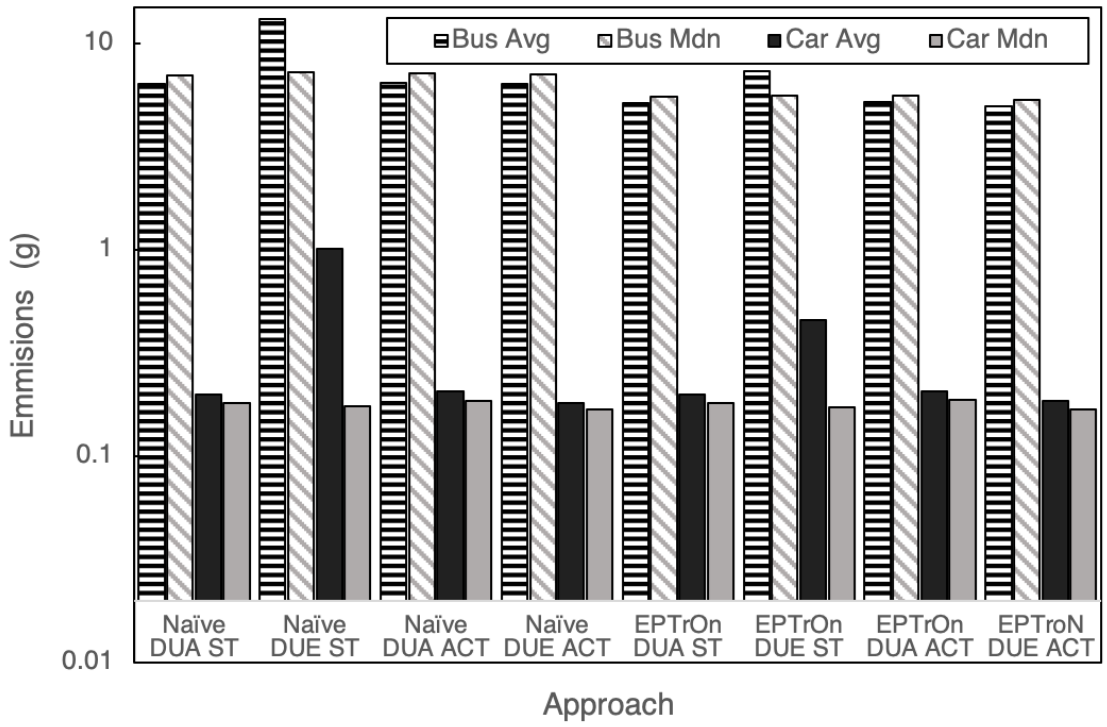

Figure 29: Emissions HC for Naive and EPTrOn

with and without virtual bus lanes, it is expected that the HC emissions will not differ either. It is to be noted, however, that buses produce an order of magnitude more HC emissions on average, compared to cars. For the cases whereby the objective is the reduction of HC emissions, in particular, buses should be used only if they carry at least 10 *times* more passengers, on average.

PMx pollution: The results for PMx emissions are depicted in Figure 30. The results are plotted on a logarithmic scale as there is an order of magnitude difference between the amounts of emissions produced by cars and buses. When EPTrOn is used, buses emit between 18% and 44% fewer PMx emissions. Moreover, the average and the grammedian values do not differ significantly—the PMx emissions are generated mostly when the vehicles are moving. The only exception is the case of DUE Static without virtual bus lanes. The behavior is consistent with the HC and CO2 emissions. Furthermore, the average and the median values for the PMx emissions, produced by cars, do not differ. Given that the waiting and lost times of cars do not differ significantly with and without virtual bus lanes, it is expected that the HC emissions will not differ either. It is to be noted, however, that buses produce an order of magnitude more HC emissions on average, compared to cars. For the cases whereby the objective is the reduction of HC emissions, in particular, buses should be used only if they carry at least 10 *times* more passengers, on average.

NOx emissions: The results for NOx emissions are depicted in Figure 31. The results are plotted on a logarithmic scale as there are two orders of magnitude difference between the amounts of emissions produced by cars and buses. When EPTrOn is used, buses emit between 18% and 44% fewer NOx emissions. Moreover, the average and the median values do not differ significantly—the NOx emissions are generated mostly when the vehicles are moving. The only exception is the case of DUE Static without virtual bus lanes. The behavior is consistent with the PMx, HC, and CO2 emissions. Furthermore, the average and the median values for the PMx emissions, produced by cars, do not differ. Given that the waiting and lost times of cars do not differ significantly with and without virtual bus lanes, it is expected that the HC emissions will not differ either. It is to be noted, however, that buses produce an order of magnitude more HC emissions on average, compared to cars.

Fuel consumption: The results for fuel consumption are depicted in Figure 32. The

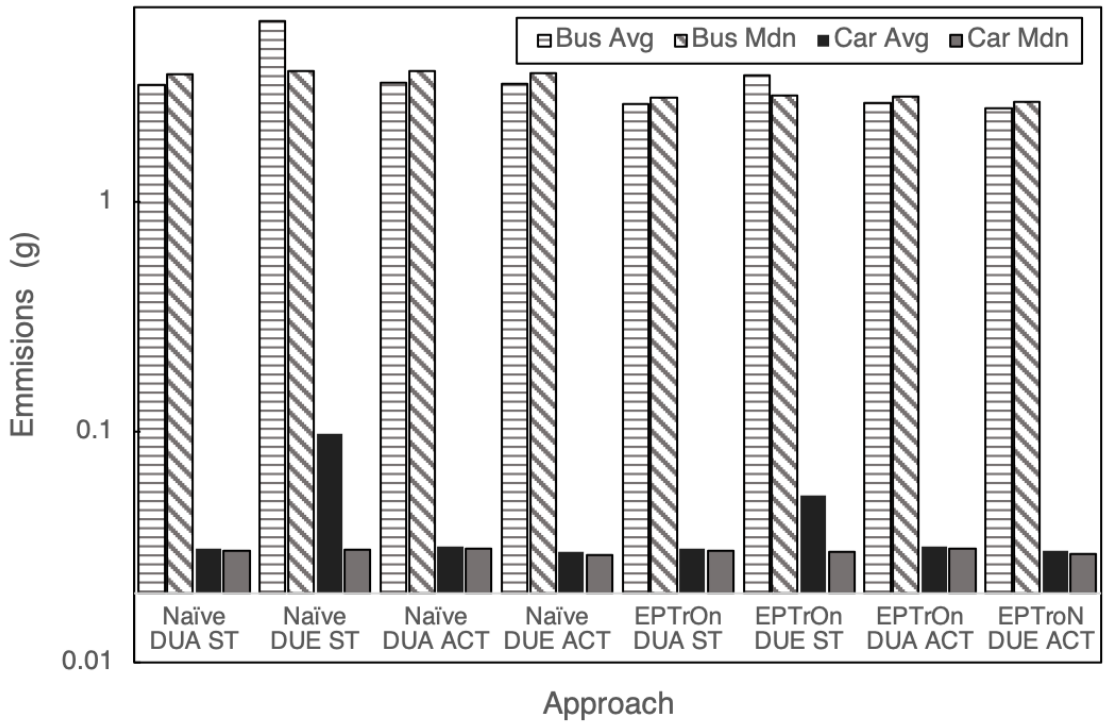

Figure 30: Emissions PMx for Naive and EPTrOn

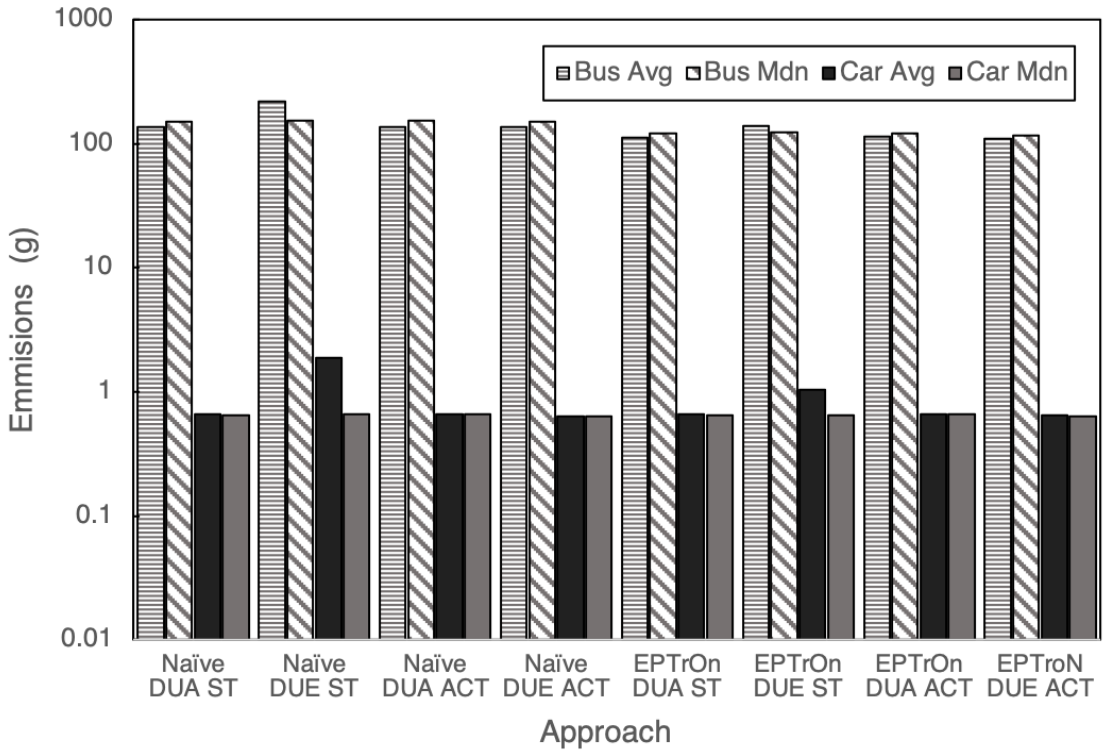

Figure 31: Emissions NOx for Naive and EPTrOn

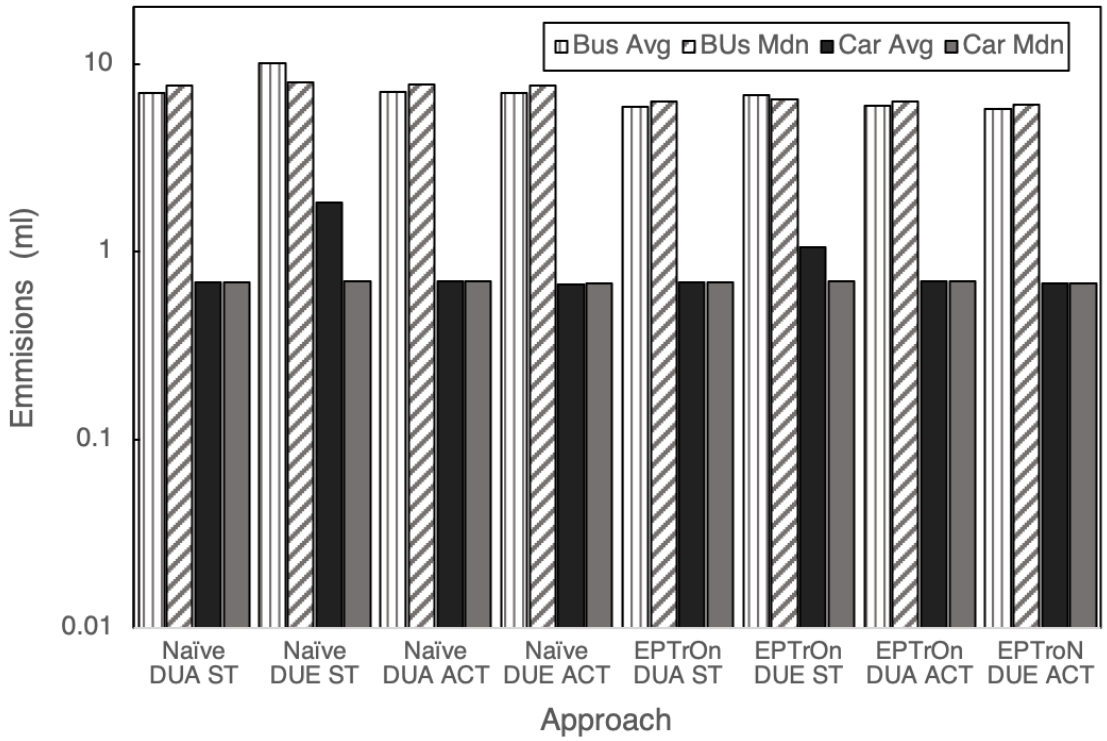

Figure 32: Fuel Consumption for Naive and EPTrOn

results are plotted in logarithmic scale as there is an order of magnitude difference between the amounts of emissions produced by cars and buses. When EPTrOn is used, buses consume 25% gramfewer fuel on average. Moreover, the average and the median values do not differ significantly—the fuel consumption when buses are in motion is significantly higher compared to when they are steady and the engines are idling. The only exception is the case of DUE Static without virtual bus lanes. The behavior is consistent with the PMx, HC, and CO2 emissions. Furthermore, the average and the median values for the fuel consumption of cars, do not differ. Given that the waiting and lost times of cars do not differ significantly with and without virtual bus lanes, it is expected that the HC emissions will not differ either. It is to be noted, however, that buses consume an order of magnitude more fuel on average, compared to cars. For the cases whereby the objective is reducing fuel consumption, in particular, buses should be used only if they carry at least 10 *times* more passengers, on average.

Conclusions: Implementing virtual bus lanes significantly decreases both the time lost and waiting time for buses. Moreover, the behavior is consistent, regardless of whether the traffic lights operate with static or actuated programs. Additionally, it does not matter whether the cars recalculate their shortest path over time. This shows that the approach can be implemented in all 4 permutations of the aforementioned two scenarios. Moreover, the amounts of greenhouse emissions and FPM pollution, generated by buses, are significantly decreased when EPTrOn is used. This behavior is consistent with the findings from the previous experiments in this chapter. It is to be noted, however, that for CO2 the emissions generated by a bus on average are an order of magnitude more compared to those of cars. The increase in the temperature on the planet is attributed to multiple factors, one of which is the amounts of CO2 in the atmosphere. This makes the mitigation of pollution, generated by buses, even more important.

5.5.2 Experiment 2: Dedicated Bus Lanes Comparison

In this experiment, we study how our model compares to dedicated bus lanes. We run the experiment in SUMO. Moreover, we reused the experimental settings from the first

experiment. The varying parameters are our novel EPTrOn approach and the currently wildly spread solution (i.e., Naive), as well as two dedicated bus lanes approaches. The first dedicated bus lanes approach *(BUS2)* defines the far-most right lane of a road segment, that is on the way of a bus, to be dedicated as bus lanes when the road segment has at least two lanes in that direction. The other approach *(BUS3)* is similar to the previous, but it has a stronger requirement—there should be at least three lanes per direction, rather than two.

Lost time: The results for the lost time for BUS2 are depicted in Figure 33. The results are plotted on a logarithmic scale. There is no significant difference between the average and the median values for buses for both approaches. Furthermore, there is no difference between the values for cars either. This is attributed to the nature of the scenario. As mentioned above, the city of Luxembourg is an old city and thus the streets in it are narrow and predominantly, the streets have one lane per direction. Moreover, we did not want to modify the organization of the traffic within the premises of the city, thus all bus lanes end before the intersections, allowing cars to be able to do right turns when necessary. Similarly to *BUS2*, the bus lines are not optimized to avoid left turns.

Waiting time: The results for the waiting time are depicted in Figure 34. The results are plotted on a logarithmic scale. Our EPTrOn solution consistently outperforms Naive and the usage of dedicated bus lanes on streets that have at least three lanes per direction (BUS3).

There is no significant difference between the average and the median values for buses for Naive and BUS3 approached. Furthermore, there is no difference between the values for cars either. This is attributed to the nature of the scenario. The city of Luxembourg is an old city and thus the streets in it are narrow. Predominantly, the streets have one lane per direction. Moreover, we did not want to modify the organization of the traffic within the premises of the city, thus all bus lanes end before the intersections, allowing cars to be able to do right turns when necessary. Moreover, the bus lines are not optimized to avoid left turns. This translates to buses leave the bus lanes in order to merge into the lanes allowing left turn when necessary. It is to be noted, however, that the approach of using dedicated bus lanes may prove efficient for cities that differ from Luxembourg in terms of buses using multi-lane streets and avoiding left turns.

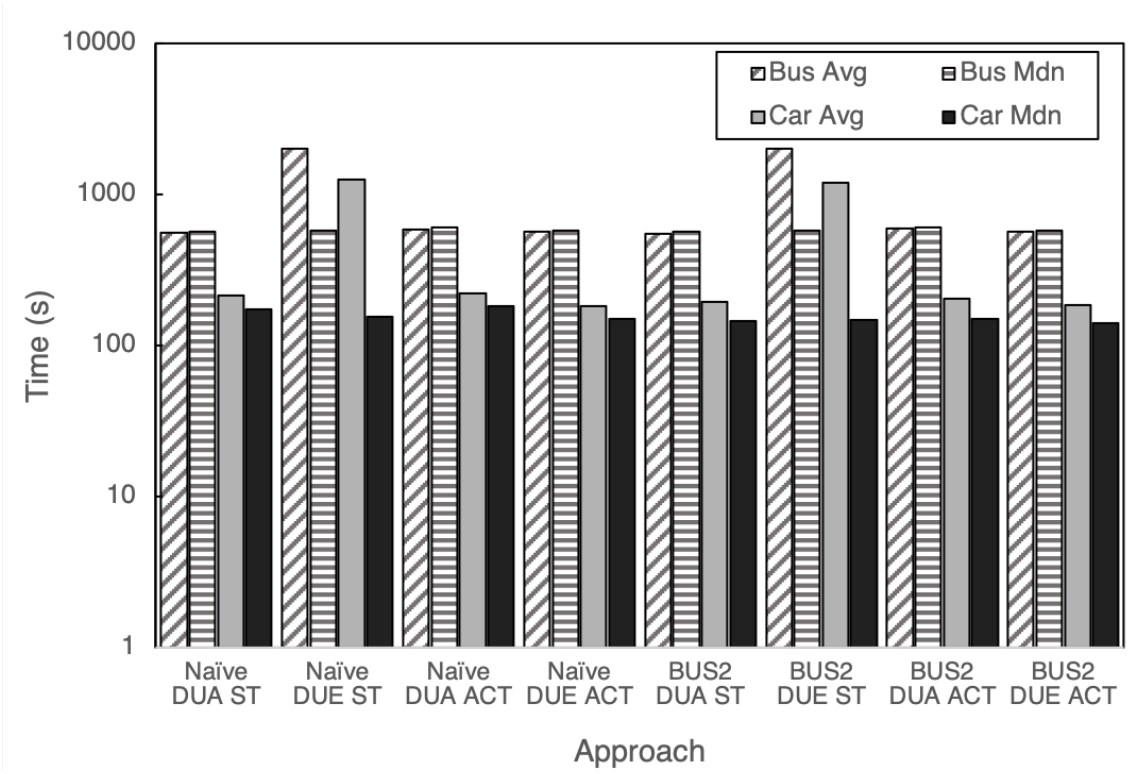

Figure 33: Time Lost for Naive and BUS2

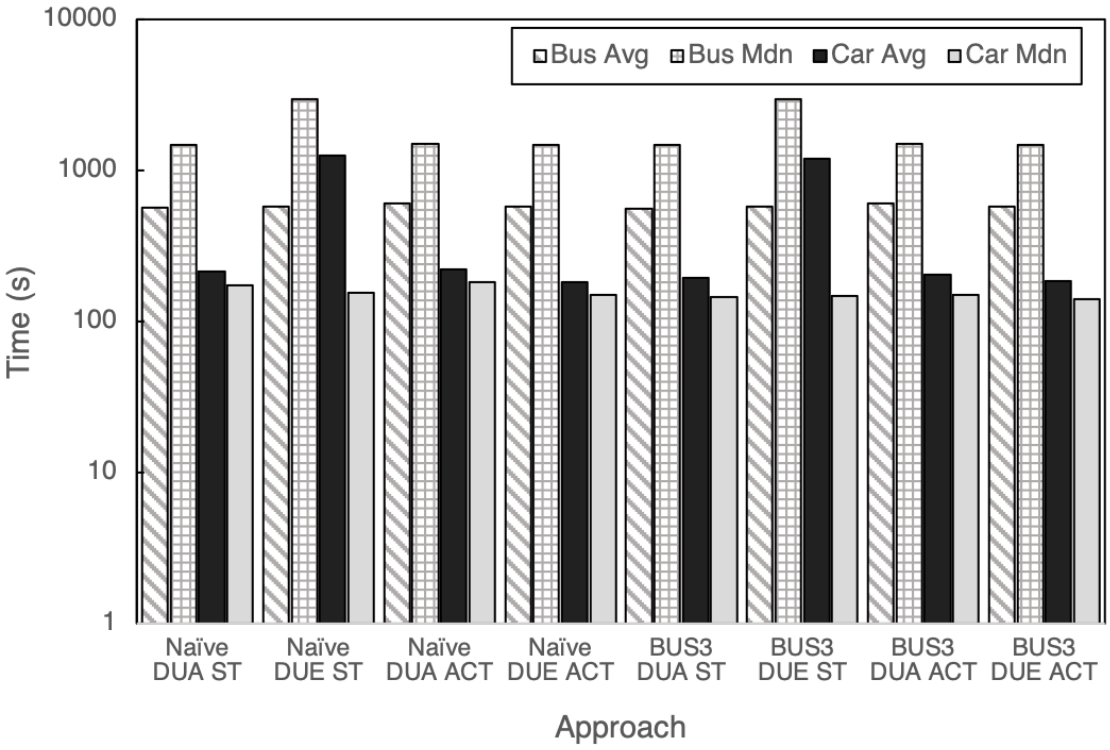

Figure 34: Time Lost for Naive and BUS3

5.5.3 Experiment 3: Scalability

We run this experiment in the SUMO simulator. We reused the parameters from the previous experiment. Moreover, we ran the experiment for six different values of the amount of traffic, specifically 30%, 60%, 90%, and 200%. The case of 200% amount of traffic mimics a deadlock of the road network. Furthermore, for the case of a 200% traffic, the maximum departure delay is deactivated—the simulation can handle as many vehicles as physical space is available, regardless of the departure delay.

Lost time: The results for the lost time are depicted in Figure 35, Figure 37, Figure 39, and Figure 41. Our EPTrOn solution consistently outperforms the naive approach, with respect to buses, shortening the lost time by 60% on average (between the different cases). This behavior is consistent for both the average and the median values. Furthermore, the metric, with respect to cars, does not show a significant increase in the lost time—it is less than 0.5% for all cases and it is consistent for both average and median cases. It is to be noted, however, that there is no difference in the average and median values for buses for all cases, with and without virtual bus lanes. This shows that the values follow a normal distribution and there is no skewness of the results. This is not the case for cars. The median value is slightly lower than the average value, by about 15%. This shows that despite the light load of streets, cars are slowed down by other cars, as buses do not show the same behavior.

Waiting time: The results for the waiting time are depicted in Figure 36, Figure 38, Figure 40, and Figure 42. Similar to the other metric, EPTrOn consistently outperforms the Naive approach, with respect to buses, by between 80% and 90%. The waiting time for cars does not get increased by more than 1 s. The exhibited behavior is consistent for both the average and the median values. Furthermore, taking into consideration the asymptotic improvement in the waiting time, compared to the improvement on the lost time for buses, we derive the conclusion that the amount of time, for which the velocity of the buses was more than 0.1 m/s is significantly improved. The difference between the average and the median values is maintained for both approaches—with and without virtual bus lanes. It is between 30% and 45% difference. Unlike the results that we discussed earlier—for the full

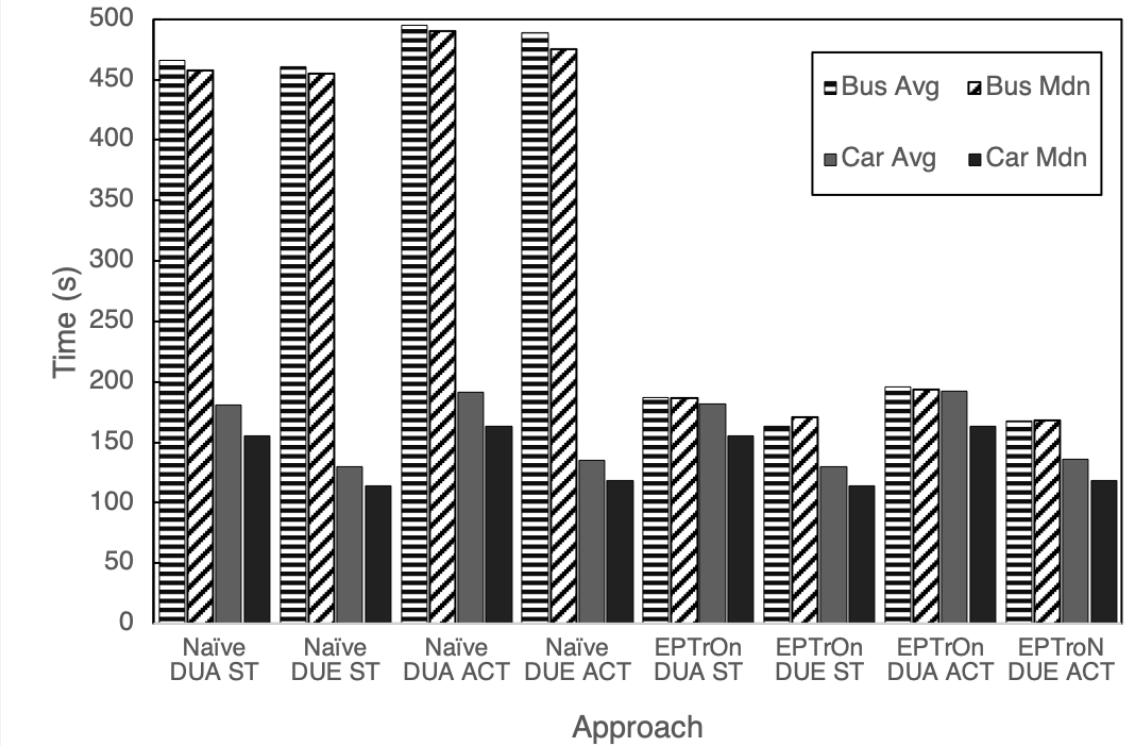

Figure 35: Time Lost for Naive and EPTrOn 30% scale

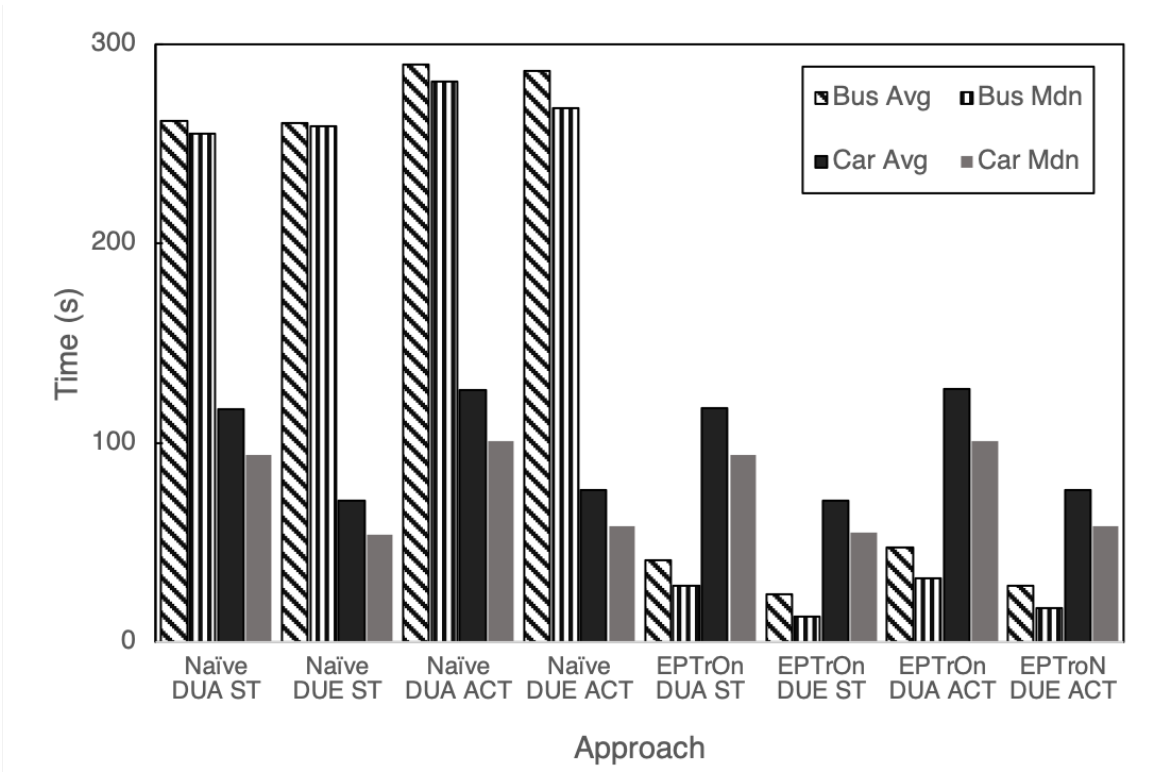

Figure 36: Waiting Time for Naive and EPTrOn 30% scale

amount of traffic in the scenario when the traffic is only 30% of that amount, the decrease of waiting time for buses is the highest amongst all four cases when the cars are in dynamic equilibrium state and the traffic lights have static schedules of their phases.

Conclusions: Our EPTrOn solution is applicable to both lightly loaded and heavily congested road networks. Furthermore, the biggest gain on maintaining the buses on time is achieved for heavily congested road infrastructure. This shows that the solution can be implemented even in very congested cities.

5.5.4 Experiment 4: Load sensitivity with Light Boards

We run this experiment in our *BeSPi* simulator. In this experiment, we study the sensitivity of our EPTrOn solution to the amount of traffic (load factor) on the streets.

The amount of traffic is calculated as a percentage of the spots for cars on all streets in the network. When we say that the road network has a 30% load, that means that the number of cars, distributed on the streets, is 30% of the total amount of car spots. The cars are distributed randomly and do not exceed the capacity of each road segment.

We experimented with 4 different values: 0%, 30%, 60%, and 90%. The traffic distribution based on directionality is Gaussian: $\{0.1, 0.2, 0.4, 0.2, 0.1\}$, the reserved space for buses is set to 1 (bp=1), and bus delay is set to 2 epochs (bd=2).

The results are depicted in Figure 43 and they show no difference between the three algorithms for 0% traffic, and that is expected. This is the case whereby there are only buses on the streets. Neither approach managed to complete even a single bus trip when there is a 90% traffic load. This is an indication that very high street loads mean gridlock for the public buses. For both 30% and 60% traffic, the buses are "delayed" by cars. Our $EPTrOn$ solution outperforms the baseline by up to 590% for the load of 30% and by 384% for 60%.

5.5.5 Experiment 5: Car-following Model Impact

In this experiment, we study how the car-following model of vehicles affects the effectiveness of our approach (EPTrOn). Our hypothesis is that the lack of middle lanes will decrease the effectiveness of the proposed novel solution. For this experiment, we set up the SUMO

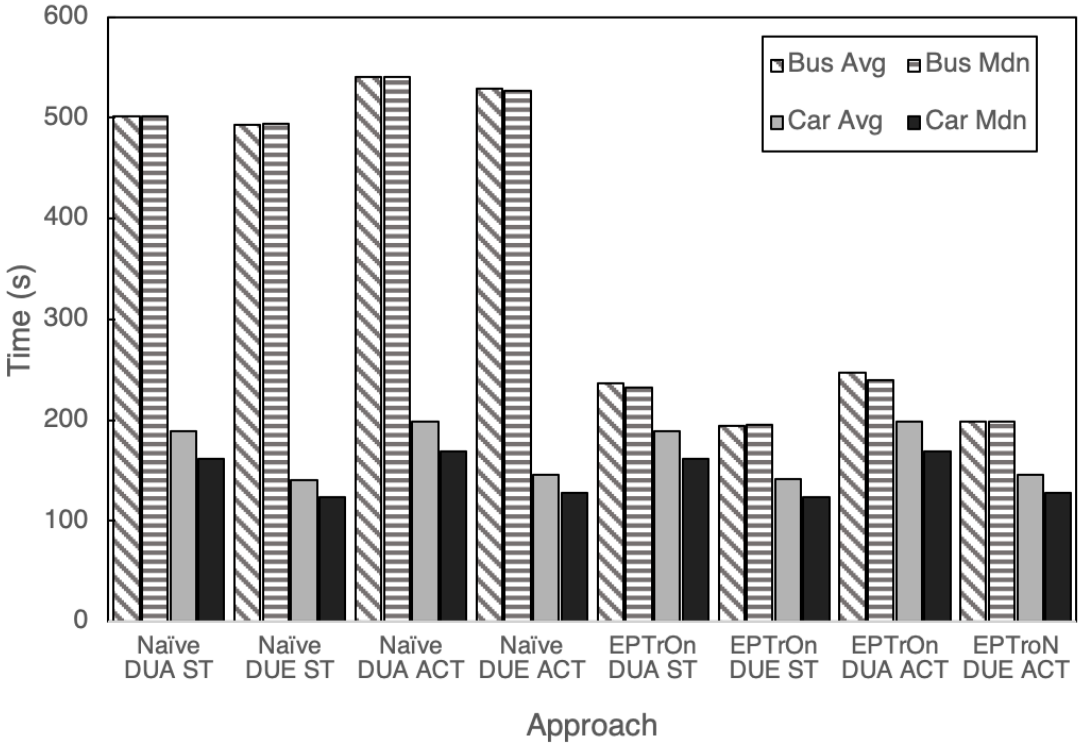

Figure 37: Time Lost for Naive and EPTrOn 60% scale

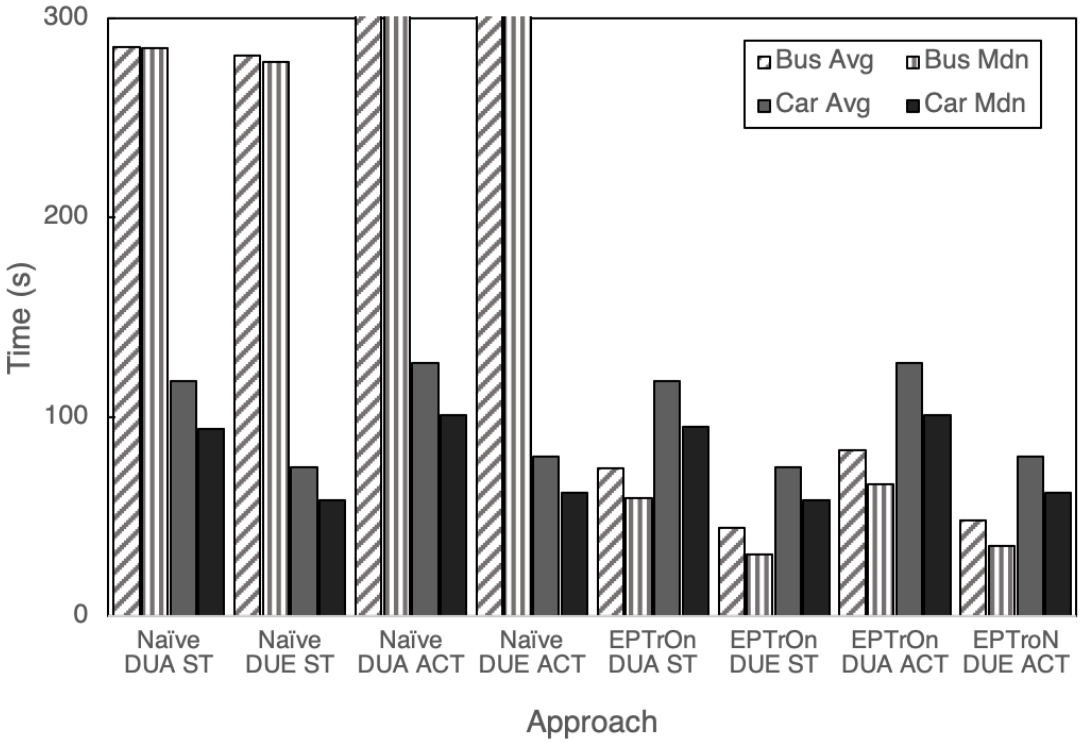

Figure 38: Waiting Time for Naive and EPTrOn 60% scale

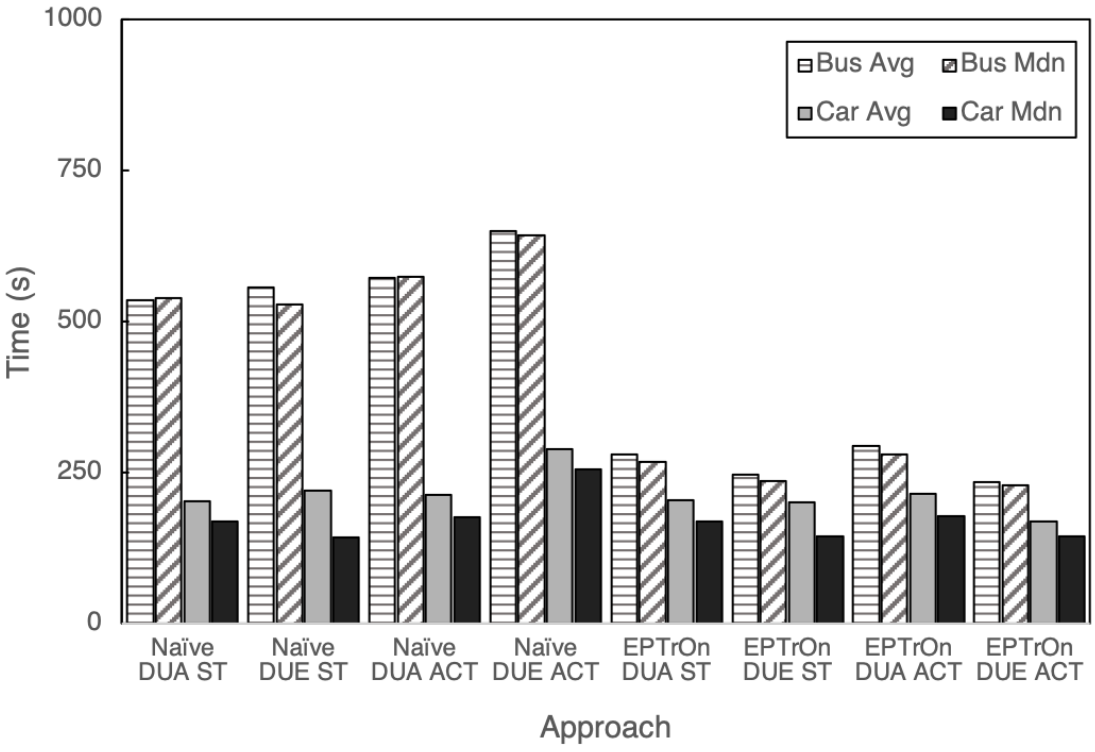

Figure 39: Time Lost for Naive and EPTrOn 90% scale

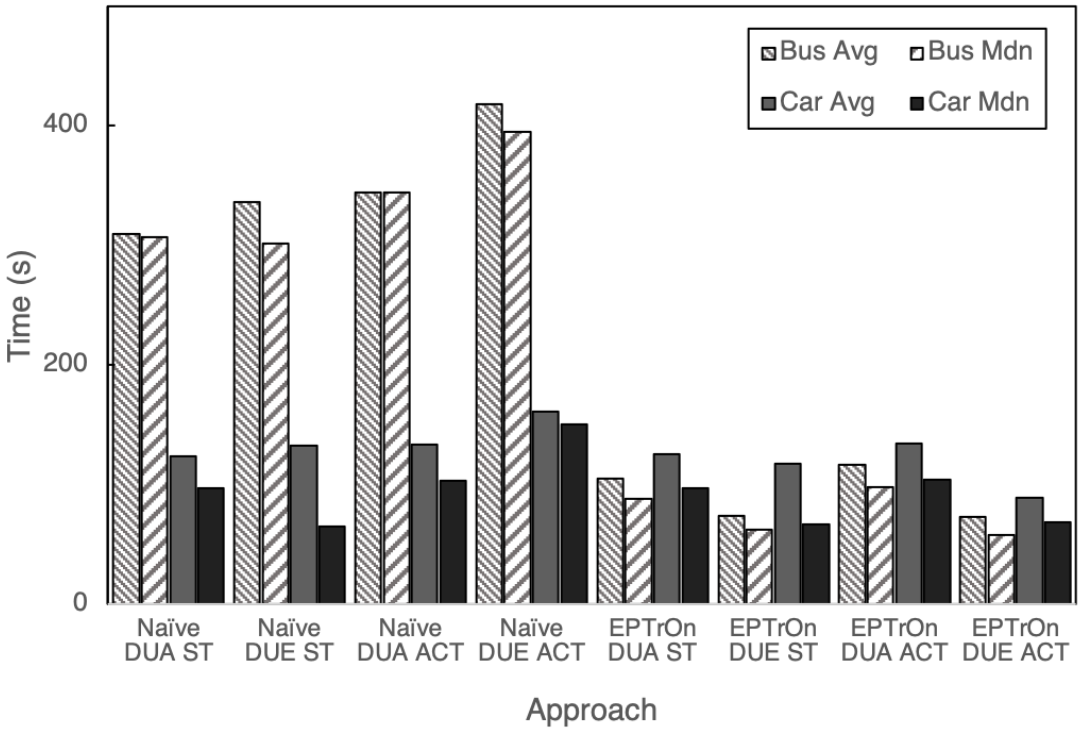

Figure 40: Waiting Time for Naive and EPTrOn 90% scale

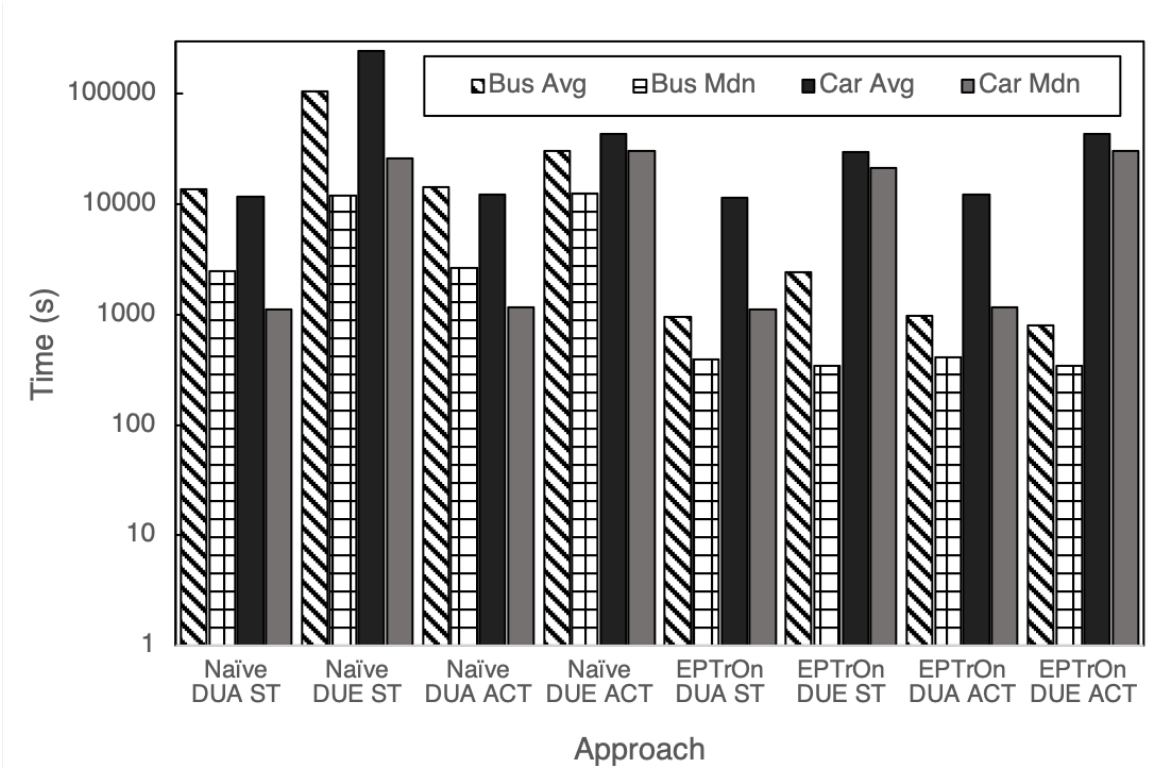

Figure 41: Time Lost for Naive and EPTrOn 200% scale

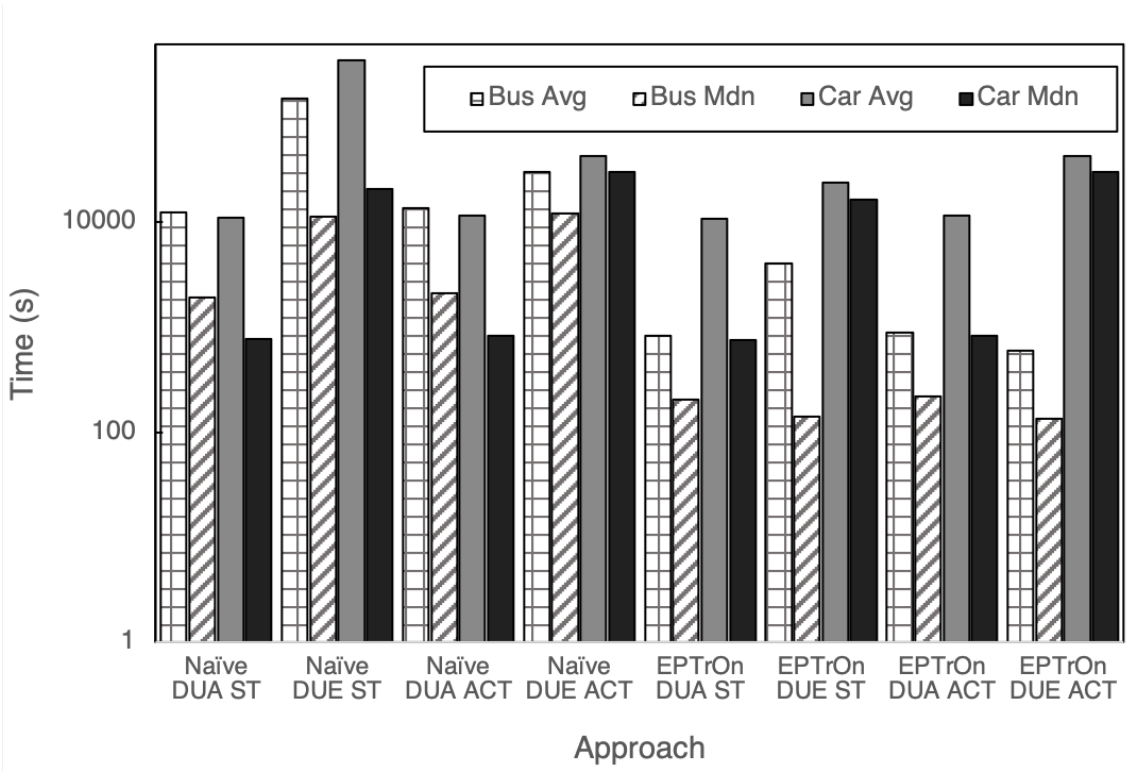

Figure 42: Waiting Time for Naive and EPTrOn 200% scale

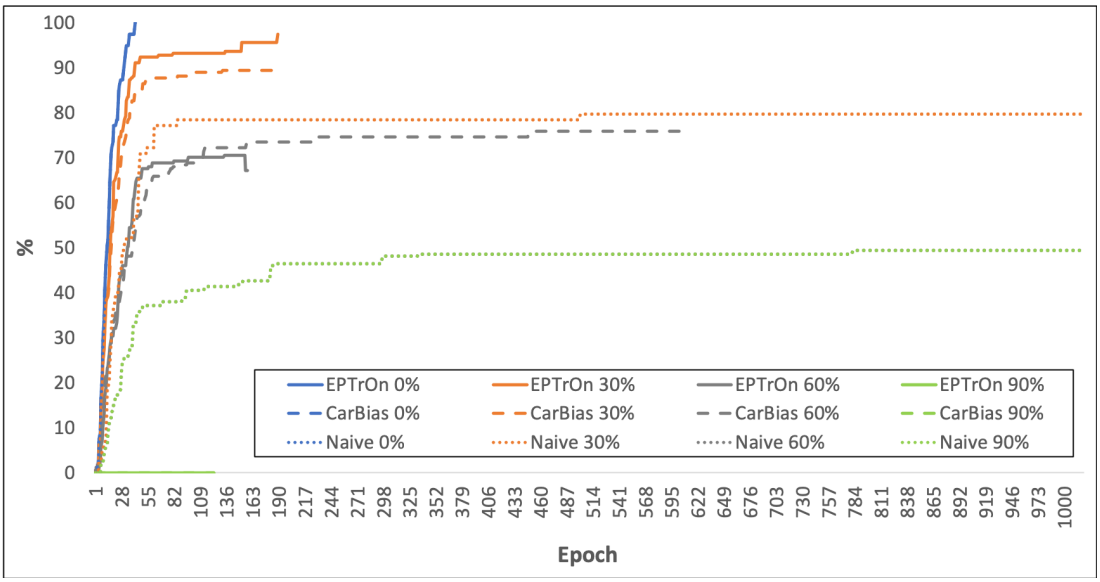

Figure 43: The percentage of bus trips completed in epochs for [0%, 90%] traffic, *EPTrOn*, and naive approaches.

simulator to run the experiment. We reused the settings from the first experiment (see Table 16). The varying parameters are our novel EPTrOn approach and the currently wildly spread solution (i.e., nave). We ran the experiment for 6 different car-following models, namely *Krauss*, *PWagner2009*, *BKerner*, *IDM*, *IDMM*, *ACC*, and *CACC*.

Krauss is the default car-following model for SUMO and the results for it are presented in Experiment 1, in Figure 25.

Lost time: The results for the lost time for the "intelligent driver model" car-following model are depicted in Figure 44. Our novel virtual bus lanes approach, namely EPTrOn, consistently outperforms the Naive approach by shortening the time lost by between 45% and 55%. Moreover, the time lost values do not change for the cars for both of the cases of static traffic lights—with and without dynamic equilibrium. The values for the two cases whereby the traffic lights use dynamic adjustment of the traffic lights phases are decreased by 8.5% when dynamic user equilibrium is not used and by 16.5% for the case when dynamic user equilibrium is used. It is to be noted, however, that the overall values of the time lost for

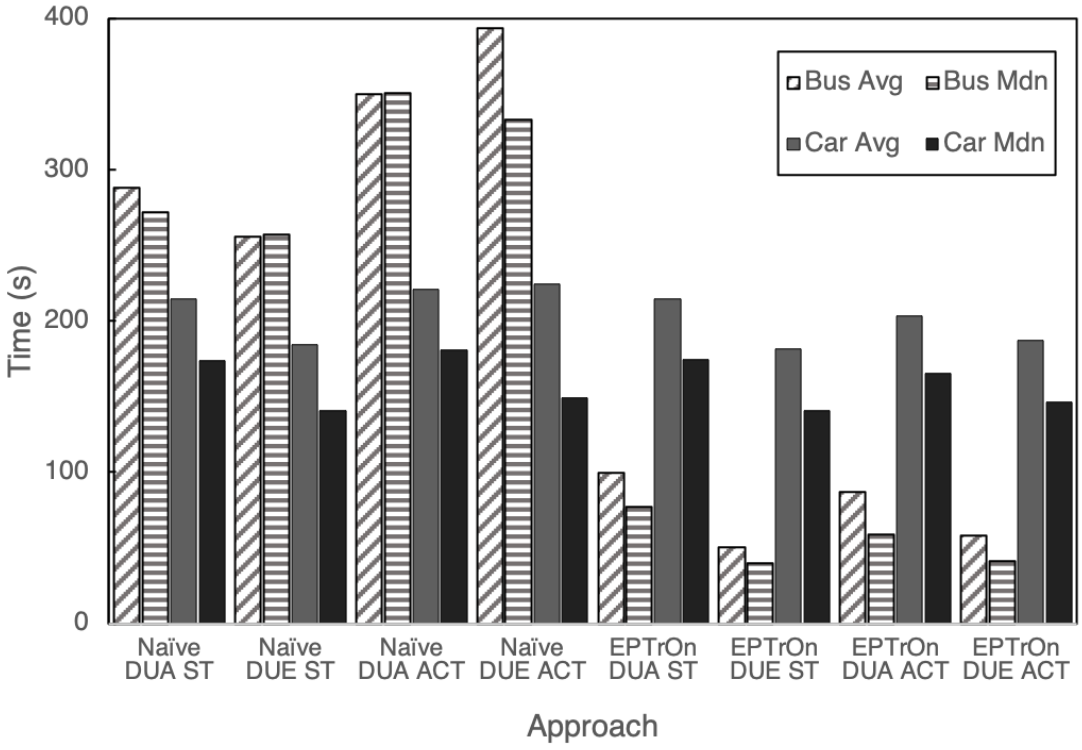

Figure 44: Time Lost for Naive and EPTrOn - IDM

buses is consistently lower, compared to the default car-following model, namely $Krauss$. The decrease is around 40% on average. This shows the importance of the car-following model for the overall performance of the implemented traffic orchestration algorithms and it should be taken into consideration when autonomous driving vehicles takeover. The biggest difference between the average and the median values for both buses and cars is for the case when dynamic user equilibrium is combined with dynamic adjustment of the traffic lights. The average value is higher than the median value which shows positive skewness. Also, the values are higher than the rest of the cases. This makes this car-following model less appealing for usage during rush hours.

Lost time: The results for the lost time for the "intelligent driver model with memory" car-following model are depicted in Figure 45. Our EPTrOn solution consistently outperforms the naive approach. It shortens the time lost for buses by between 48% and 64% (55% on average). The time lost for cars increases with 7 s at most, which is only 3%. Furthermore, these changes do not come at the cost of additional traveled distance. The values for buses are consistently higher, compared to both IDM and $Krauss$ car-following models. This shows that when drivers are exposed to an unrelaxed surrounding environment, they make more mistakes and this affects the overall performance of all vehicles. This observation confirms the intuition that tired and/or frustrated drivers make more mistakes and effectively slow down the traffic. It is to be noted, however, that for the cases whereby dynamic user equilibrium (DUE) is combined with dynamic actuation of the phases of the traffic lights and the buses use virtual bus lanes, the lost time for cars goes down about 3%. This shows that the prioritization of buses actually improves the overall travel time for cars.

Lost time: The results for the lost time for the "adaptive cruise control" car-following model are depicted in Figure 46. Our novel virtual bus lanes approach, namely EPTrOn, consistently outperforms the Naive approach by shortening the time lost by between 72% and 85%. The time lost for cars increases by between 3 s and 10 s at most, which is 10%. It is to be noted, however, that both the average and the median values for buses are lower for ACC than all other car-following models we have discussed already, specifically $Krauss$, IDM, and $IDMM$. This is attributed to the fact that ACC uses sensors, radars, and lidars to continuously monitor the surrounding environment and maintain shorter save

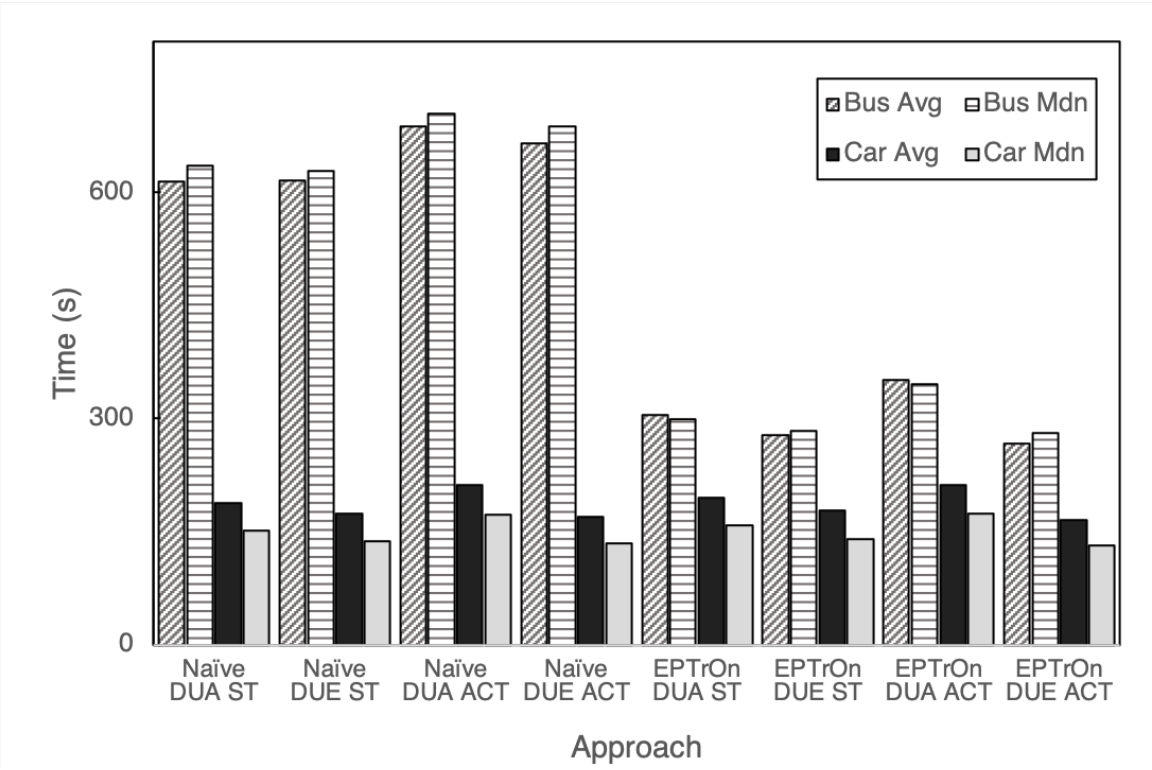

Figure 45: Time Lost for Naive and EPTrOn - IDMM

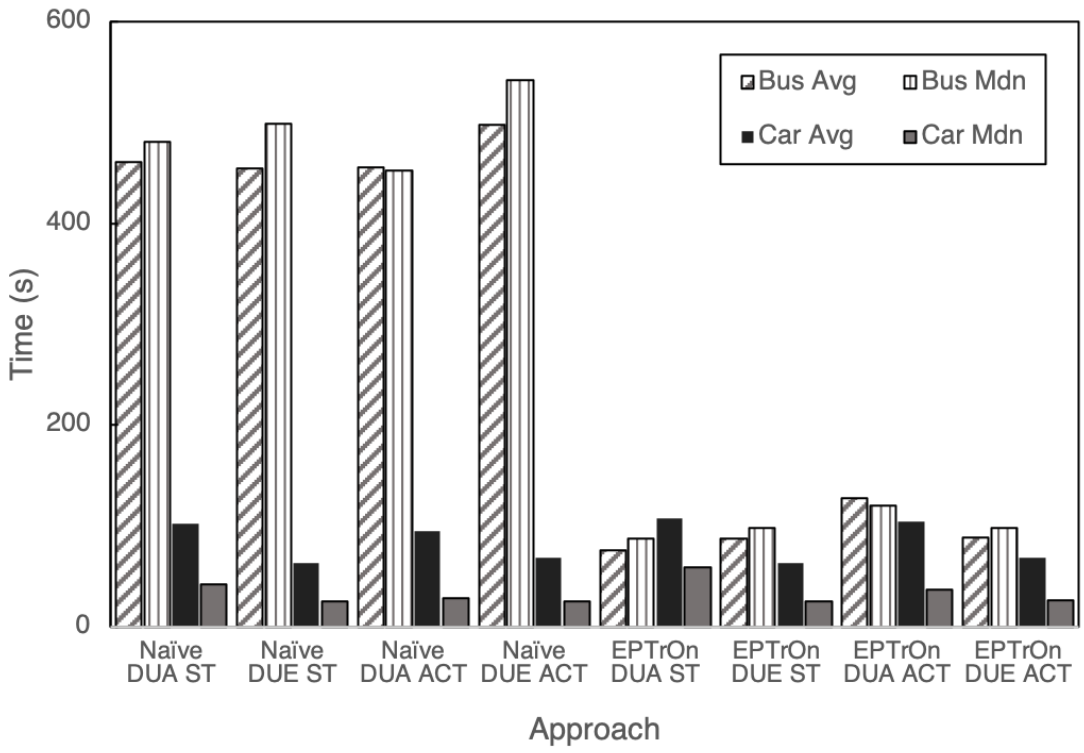

Figure 46: Time Lost for Naive and EPTrOn - ACC

distances to the followed cars compared to the aforementioned models. Furthermore, there is a significant difference between the average and the median values for the metric for cars. The median is consistently lower than the average, which shows that there is a positive skewness of the results. This behavior is attributed to the relatively small number of cars that had significantly higher values during rush hours. The latter is expected behavior.

Lost time: The results for the lost time for the "cooperative adaptive cruise control" car-following model are depicted in Figure 47. Our EPTrOn approach consistently outperforms the Naive approach by shortening the time lost by between 70% and 80%. The time lost for cars increases with 10 s at most, which is 10%. It is to be noted, however, that both the average and the median values for buses are lower for $CACC$ than all other car-following models we have discussed already, specifically ACC, $Krauss$, IDM, and $IDMM$. This is attributed to the fact that ACC uses sensors, radars, and lidars to continuously monitor the surrounding environment and maintain shorter save distances to the followed cars, compared to the aforementioned models. Furthermore, there is a significant difference between the average and the median values for the metric for cars. The median is consistently lower than the average, which shows that there is a positive skewness of the results. This behavior is attributed to the relatively small number of cars that had significantly higher values during rush hours. The latter is expected behavior.

Lost time: The results for the lost time for the car-following model developed by Boris Kerner are depicted in Figure 48. Our EPTrOn approach consistently outperforms the Naive approach by shortening the time lost by 40% on average. The time lost for cars increases by 3 s at most, which is 0.5%. It is to be noted, however, that both the average and the median values for buses are higher for $BKerner$ than all other car-following models we have discussed already, specifically ACC, $CACC$, $Krauss$, IDM, and $IDMM$. This is attributed to the fact that actuation of the vehicle's velocity are discrete, even though the monitoring of the surrounding environment is continuous.

Conclusions: This experiment shows us that our EPTrOn solution consistently outperforms the naive approach for all car-following models, which we experimented with. Moreover, the gain is higher for the more advanced models, where the vehicles are equipped with more sophisticated means to measure, maintain and adjust the distance to the vehicle in

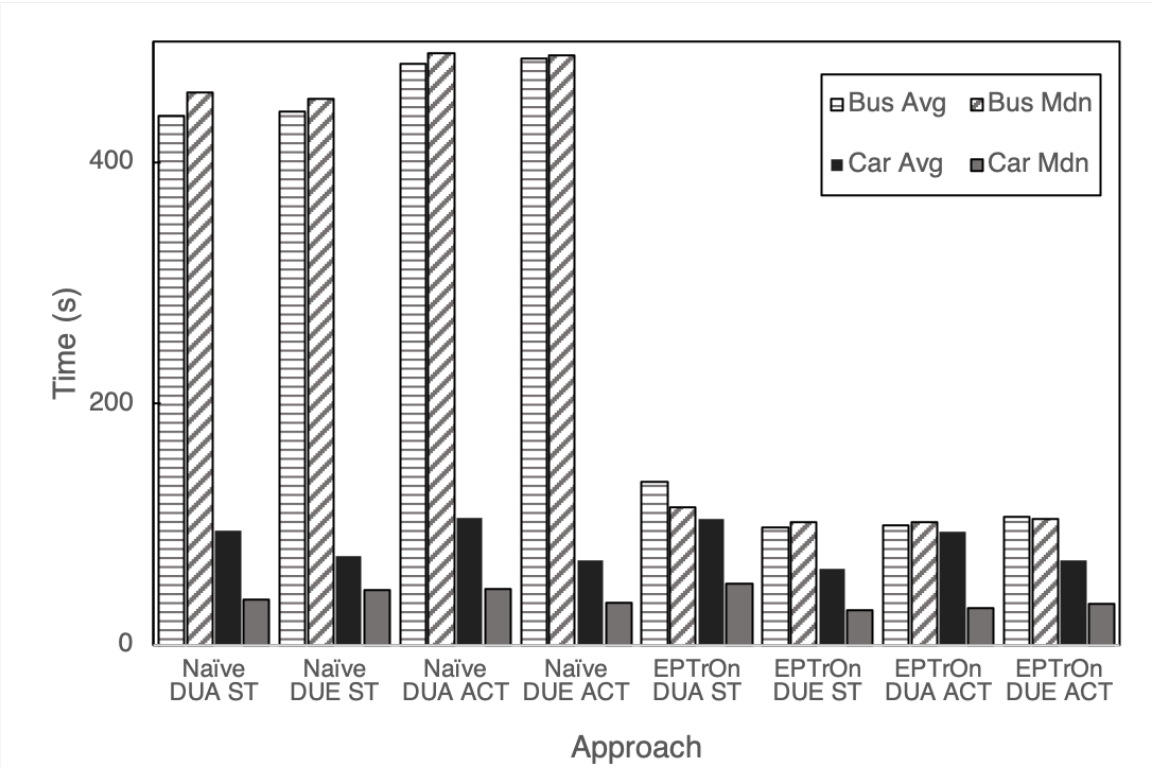

Figure 47: Time Lost for Naive and EPTrOn - CACC

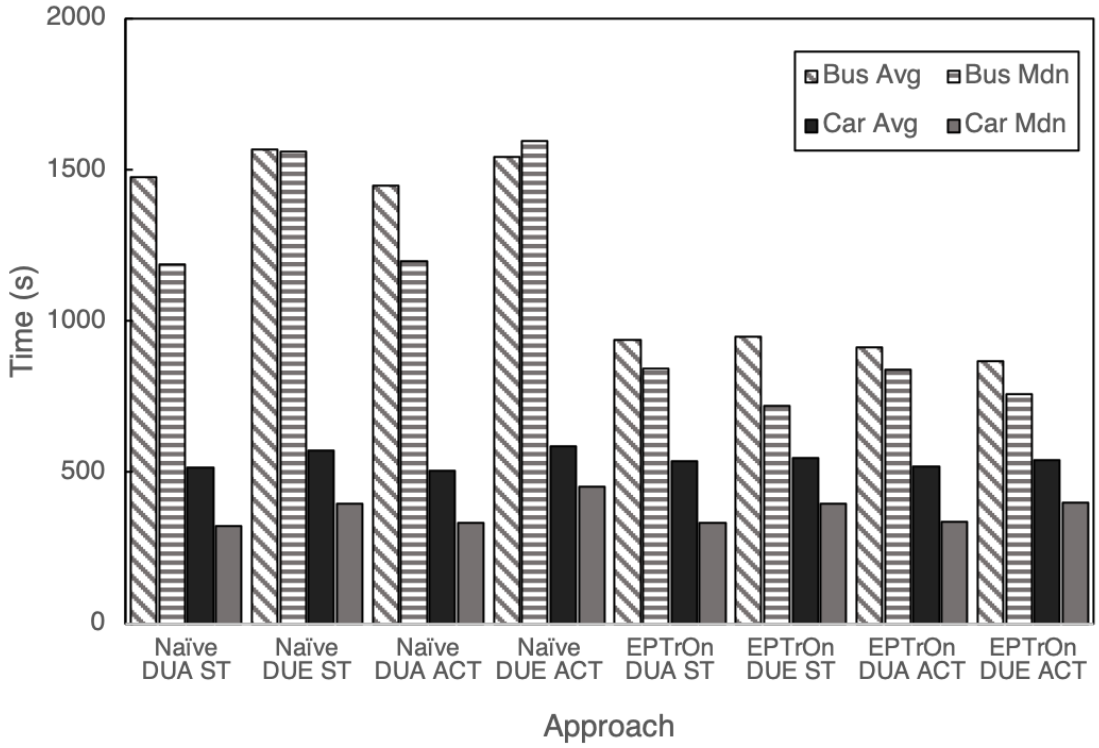

Figure 48: Time Lost for Naive and EPTrOn - BKerner

front of them. Our EPTrOn solution will be applicable to the usage of autonomous driving vehicles in the future, which rely on these technologies.

5.5.6 Experiment 6: Dynamic Traffic Lights Approach

Our hypothesis for this experiment is that modifying only the traffic lights for the lanes that are on bus routes does not help either the buses or the cars. For this experiment, we set up the SUMO simulator to run the experiment for the settings, as defined in the first experiment. The varying parameters are our novel EPTrOn approach and the currently wildly spread solution (i.e., Naive). Moreover, we test 2 of the traffic orchestration approaches, specifically DTL-Red and DTL-Green.

Lost time: The results for the time lost for DTL-Red are depicted in Figure 49. The results are plotted on a logarithmic scale as there is an order of magnitude difference between the values for the two approaches. Our Dynamic Traffic Lights with Red signal (DTL-Red) solution consistently underperformed, compared to the Naive approach for both buses and cars, prolonging the lost time with an order of magnitude for cars and around 8 times for buses. This behavior is consistent for both the average and the median values. This is an expected effect, caused by changing the traffic lights signal to red for the edges that have at least one bus traveling towards the intersection. Furthermore, the results are consistent for all four cases—with static and dynamic scheduling of the traffic lights phases, as well as with or without dynamic user equilibrium. It is to be noted, however, that the method eliminated the differences between the median and average values. This shows the results follow a normal distribution. These results show that slowing down cars and buses in an effort to free the next edges that public transportation buses are heading to is not a promising strategy and it should be avoided as a measure to decrease the time lost by mass transit vehicles.

Lost time: The results for the lost time for DTL-Green are depicted in Figure 50. The results are plotted on a logarithmic scale as there is an order of magnitude difference between the values for the two approaches. Our DTL-Green solution consistently underperformed, compared to the Naive approach for both buses and cars, prolonging the lost time with an

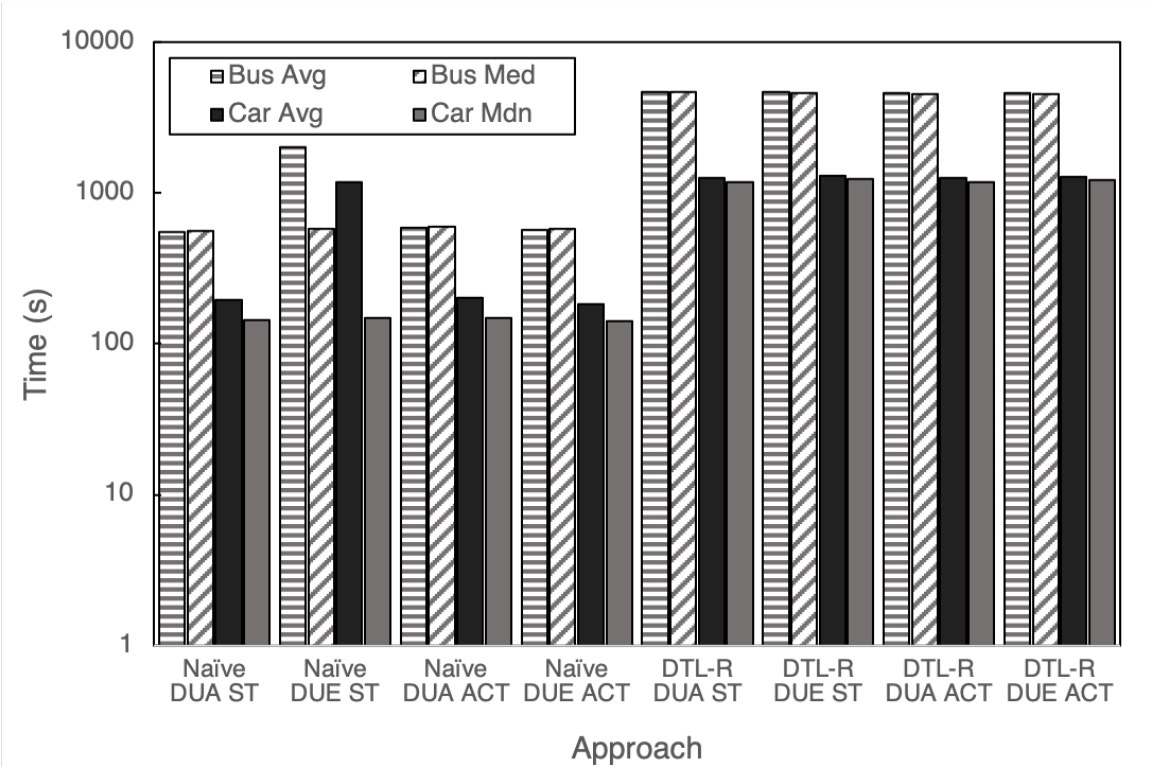

Figure 49: Time Lost for Naive and DTL-Red

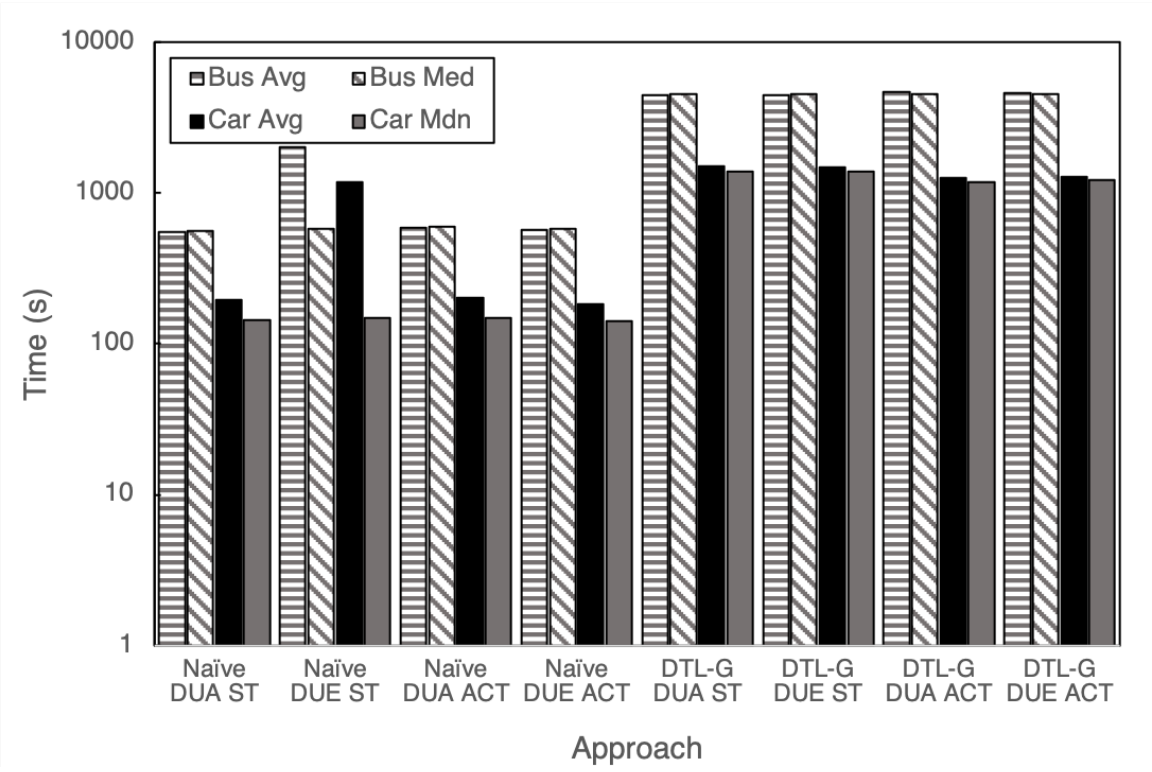

Figure 50: Time Lost for Naive and DTL-Green

order of magnitude for cars and around 8 times for buses. This behavior is consistent for both the average and the median values. This is an expected effect, caused by changing the traffic lights signal to green for the edges that have at least one bus traveling towards the intersection, cars on the other edges that are connected to the same intersection. Furthermore, the results are consistent for all cases - with static and dynamic scheduling of the traffic lights phases, as well as with or without dynamic user equilibrium. It is to be noted, however, that the method eliminated the differences between the median and average values. This shows the results follow a normal distribution.

Conclusions: These results show that slowing down cars and some buses in an effort to speed up the passing of the public transportation buses are heading to is not a promising strategy either and it should be avoided as a measure to decrease the time lost by mass transit vehicles. Noticeably, both approaches, specifically DTL-R and DTL-G induce the same delay, regardless of the difference in prioritizing vehicles.

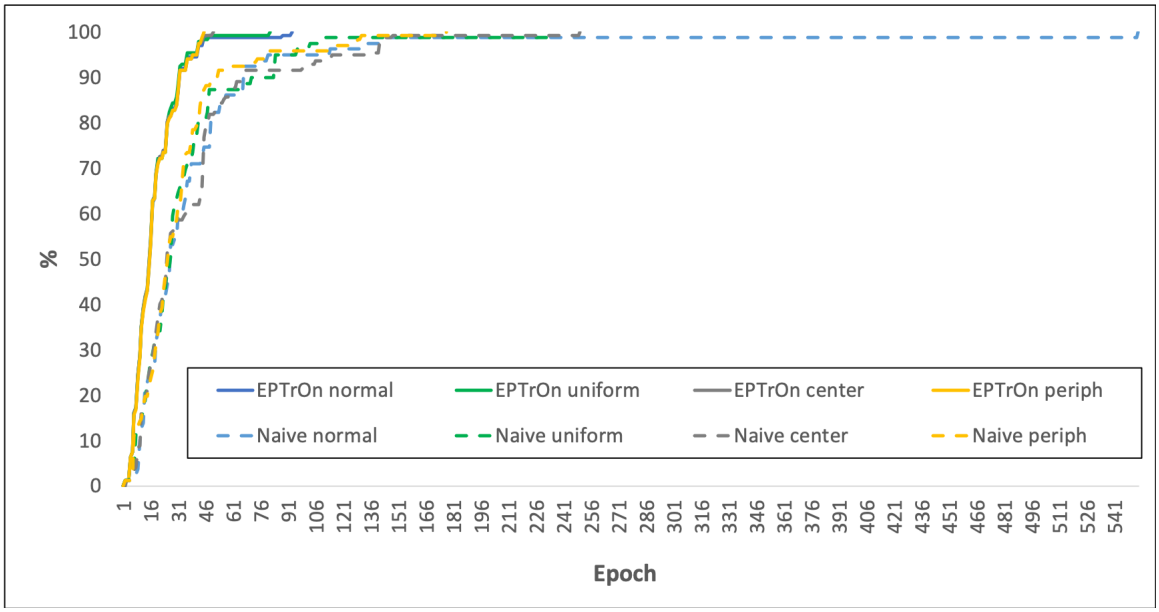

Figure 51: The percentage of bus trips completed for Gaussian, uniform, center and peripheral destination distribution, for EPTrOn and naive approaches.

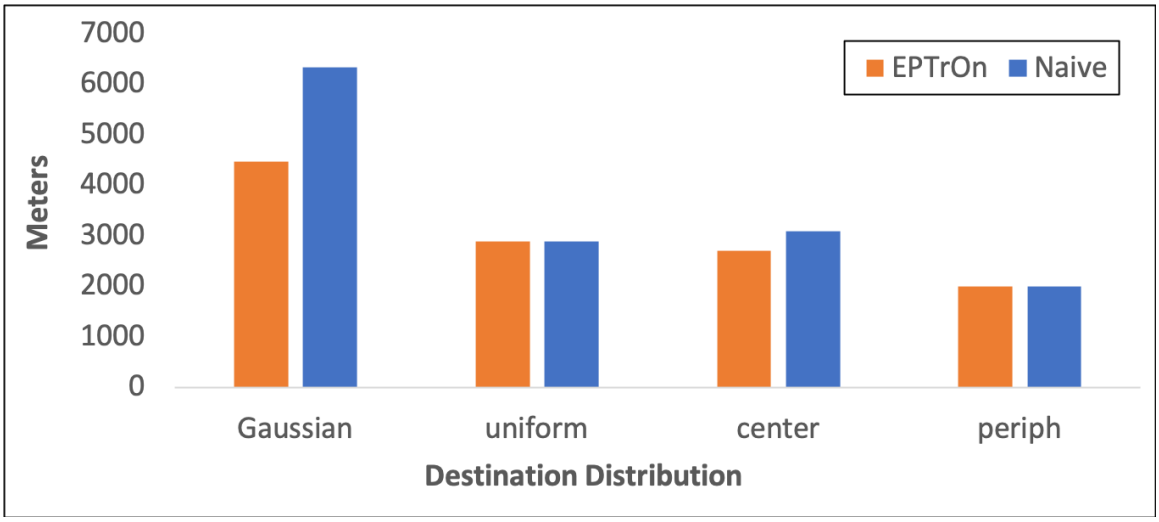

Figure 52: The average detour distance for ICEV cars, Gaussian, uniform, center and peripheral destination distribution, EPTrOn and naive approaches.

5.5.7 Experiment 7: Approach Robustness

The results of the experiment are depicted in Figure 51 and Figure 52. In this experiment, we study the robustness of *EPTrOn* against the number of traffic violators, who ignore the detour directives and go in a direction different than where they were directed to go by the light-boards. More details about modeling the violators are available in Section 5.4. The results for *EPTrOn* are presented with solid lines, while the Naive approach is depicted on dashed lines. Clearly, our solution consistently outperforms the naive, despite the number of violators. In conclusion, *EPTrOn* is robust enough to accommodate up to 80% of violators.

We also report the average length of the car detour. We measured the length of detours for all cars. The Naive approach detour mimics the decisions drivers take when they get stuck in traffic. Specifically, they detour and expect to get onto a less busy road. When *EPTrOn* is used, the detour is defined by the routing directives. The results are depicted in Figure 52. The difference between the detour penalty for Naive and *EPTrOn* for Gaussian, uniform, center, and peripheral distributions is 1859, 539, 394, and 283 meters, respectively. Our solution does not incur more than 768 meters detour penalty on average (between the 4 distributions).

5.6 Summary

In this chapter, we presented the experimental evaluation of our EPTrOn virtual bus lanes solution. We evaluated it with two different datasets, specifically the traffic scenario for the city of Luxembourg and the traffic in the city of Beijing. We evaluated experimentally our solution in two different simulators, namely SUMO and our BeSPi simulator. Our solution consistently outperforms the baseline naive approach and it is agnostic to the chosen car-following models. EPTrOn shortens the time lost in traffic by buses three times on average. Moreover, the penalty induced on cars in order to create the virtual bus lanes do not go beyond 10 seconds on average (less than 1% on average).

6.0 Conclusions

In this section we summarize our contributions, we also present the broader impact of this dissertation and briefly discuss some of the possible future extensions.

6.1 Summary of Contributions

In this dissertation, we aim to optimize the duty cycle of HVAC systems in smart homes and mitigate congestion in smart cities. The common goal in these two aims is the reduction of energy consumption and the reduction of atmospheric pollution. To this end, we propose to leverage scheduling principles and statistical techniques in the context of two applications, namely aiming to reduce the duty cycle of HVAC systems in smart homes and to mitigate road congestion in smart cities. Our **hypothesis** is:

Data processing and decision making need to be carried out at the network edge, specifically as close to the physical system as possible, where data are generated and used, in order to produce results in real-time and make sure the data is not exposed to privacy and security risks.

We developed two frameworks that we used for experimental evaluation of our hypothesis and demonstration of its impact:

(1) Practical HVAC Scheduling: We presented an IoT (Internet of Things) solution that leverages sensors in smart homes as input and schedules the duty cycles of HVAC systems in residential buildings intelligently. It reduces the energy consumption for space conditioning while meeting users' comfort requirements for the target temperature. It works on a per-room basis. Our solution, called *Integer Linear Programming for Smart Scheduling* (ILPSS), takes the desired temperature along with the maximum time the user expects for the temperature to be regulated (which we call *a deadline*). Its innovation is that ILPSS combines scheduling and regression techniques. The former optimizes HVAC

duty cycles and the latter estimates the time needed to reach the desired temperature for each request.

(3) On-demand Dynamic Bus Lanes Creation: We propose a proactive solution that ameliorates the traffic ahead of public buses in congested areas, called *Environment Protective Traffic Orchestration* (EPTrOn). EPTrOn mitigates congestion by establishing virtual bus lanes and shaping traffic by controlling traffic lights and directing traffic using lightboards at intersections. Moreover, our EPTrOn solution pushes ICEV (Internal combustion engine vehicles) away from congested streets, where their engines will be idling for a prolonged amount of time. We employ priority scheduling techniques to drive the implicit creation of "green waves" for traffic. Furthermore, we use spatial indexing and window-based aggregation techniques to assign vehicles to streets and determine traffic jams. Dijkstra's shortest path algorithm is used to calculate and recommend less congested routes.

6.2 Broad Impact

There is an increasing trend in the media outlets sharing concerns about the pollution, resulting from human activities. A significant proportion of the pollution is attributed to transportation and space conditioning. Parallel to this, the Internet of Things (IoT) is making its way into becoming a disruptive technology that leads to leaps in many aspects of our daily life. To this day our home appliances are smart, our cars get "over air updates" of their software, many industries used IoT as a springboard to optimize their production processes. Additionally, the ubiquitous cloud computing enabled possibilities for carrying out computations in environments that are not resource-constrained, theoretically. The latter gave rise to an increase in privacy and security concerns amongst users. This dissertation addresses the challenges related to reducing pollution, caused by space conditioning in residential buildings and public transportation buses equipped with internal combustion engines.

We showed that our proposed solutions can be implemented and run on resource constrained IoT-ready devices, alleviating the burden of significant capital investments for com-

putational power. Furthermore, the shortened duty cycles of the HVAC systems and the reduced travel time of buses lead to a reduction in the necessary fuel to run them - natural gas, gasoline, diesel, and electricity. The latter can be translated to reduced expenses, or savings, for the households and the port authorities that will use our solutions. The reduction of pollution will have an additional positive effect on decreasing the number of people suffering from respiratory diseases, related to it. Furthermore, improving the on-time performance of public transportation will make it more appealing to the audience. Having more people using mass transit will cause more efficient use of the available road infrastructure as the number of cars will decrease, and the congestions will be alleviated. Traffic improvements have a two-fold gain - they will lead to the reduction of stress amongst the drivers and they will also postpone the necessity of significant investments in building additional road infrastructure, while cities continue to grow as the percentage of people living in cities as opposed to villages keeps increasing worldwide.

6.3 Future Work

This dissertation can be extended further into several directions:

- *Fixed Time Interval* - in our current HVAC scheduling solution, we use the last eight sensor readings to estimate the time for supplying thermally conditioned air to a room. It is worth studying the trade-offs between a fixed number of readings and fixed time length - i.e., the last 20 minutes.
- *AggregateMultiple Requests Per Room* - it is worth proposing a model whereby the personal preferences of several occupants of a room are aggregated, as opposed to overwritten, and a compromise on the target temperature and their comfort zone is achieved.
- *Commercial Buildings* - there are a number of differences in the way a forced air space conditioning systems are implemented in commercial buildings versus residential buildings. Some installations have two parallel ducting systems, capable of supplying both warm and cool air at the same time. The lack of walls and the circulation of air within the open spaces is another challenge.

- *Cost − CautiousHVACScheduling* - In an effort to mitigate pollution and to drive prices down, a number of countries offer varying electricity prices [31]. The price changes throughout the day, reflecting the current demand and supply of electricity in the grid.
- *Beyond Forced Air Systems* - out HVAC scheduling solution assumes that the building is equipped with forced air system for space conditioning. The use of radiators and/or air condition units, that are installed on a per room basis affects the challenges in scheduling the duty cycle of the HVAC system.
- *Virtual Public Transportation Lanes* - mass transit vehicles that are running on tracks (trams, light rail, etc) share the road network with other traffic in many cities. Enabling the virtual bus lanes concept to them may lead to improving their on-time performance and reducing their energy consumption.
- *Intermodal Transportation* - the improved on-time performance of overground mass transit will affect waiting times for passengers, who are relying on inter-modal transportation. An additional study is necessary to optimize the connections and bound the waiting times within a reasonable limit. Moreover, the change in the distribution of people over time at the inter-modal stations may cause scheduling problems for the different types of vehicles and the different transportation lines.
- *Emissions and Pollution in Rural Areas* - given the disproportion of pollution emitted by cars and buses and population's density per square mile, a more thorough study should be carried out to identify the boundary within which the availability of public transportation leads to a net reduction in the emissions, compared to using cars.

In the future, we plan to work in close cooperation with companies and local authorities to implement the proposed solutions in buildings and cities as opposed to running simulations in our experimental frameworks.

Our EPTrOn solution can be integrated with some of the widely adopted navigation solutions available today—such as Google Maps, Waze, and others. This will increase the value of these applications for the end-users. Moreover, providing guidance on the cellular phones of the users will eliminate the need to install light boards, and, therefore, decrease the necessary investments in infrastructure the authorities have to make in order to enabled the solution. Providing guidance in commercially available applications, which collect in-

formation about the location of their users contradicts our premise to mitigate exposure of users' sensitive data to privacy and security concerns. It is to be noted, however, that this collection of information is accepted by many users today. As the location data is available in the cloud already, it can be processed there. The latter will completely eliminate the need to invest in additional infrastructure on the road network to monitor the traffic, will shorten the time-to-market of the functionality significantly, and will eliminate the financial burden on the authorities that own and operate the road network completely.

Last, but not least, some car manufacturers such as Audi made Google Maps their choice of navigation software. Integrating the functionality of EPTrOn with Google Maps will eliminate the use of cellular phones completely for those car brands. This will eliminate the distraction of the drivers to look at their phones and subsequently will decrease the number of car accidents.

Bibliography

[1] Accident Sketch. Accident Scene Diagrams. `draw.accidentsketch.com/`, 2019. [Accessed 2019-12-10].

[2] Bashar Al-Omari and Ahmad Alomari. Validation of the traffic simulation software (corsim) for roundabouts in jordan. In *10th Transport Engineering Conference (CIT 2012)*, 07 2012.

[3] Rakan Alseghayer, Daniel Petrov, and Panos K. Chrysanthis. Strategies for detection of correlated data streams. In *Proceedings of the 5th International Workshop on Exploratory Search in Databases and the Web*, ExploreDB, 2018.

[4] Rakan Alseghayer, Daniel Petrov, Panos K. Chrysanthis, Mohamed Sharaf, and Alexandros Labrinidis. Detection of highly correlated live data streams. In *Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics*, BIRTE, 2017.

[5] Rakan Alseghayer, Daniel Petrov, Panos K. Chrysanthis, Mohamed Sharaf, and Alexandros Labrinidis. Dcs: A policy framework for the detection of correlated data streams. In *Real-Time Business Intelligence and Analytics*, pages 191–210, 2019.

[6] O. Andersen, C. S. Jensen, K. Torp, and B. Yang. Ecotour: Reducing the environmental footprint of vehicles using eco-routes. In *2013 IEEE 14th International Conference on Mobile Data Management*, volume 1, pages 338–340, 2013.

[7] A. Aswani, N. Master, J. Taneja, D. Culler, and C. Tomlin. Reducing transient and steady state electricity consumption in HVAC using learning-based model-predictive control. *Proceedings of the IEEE*, 100, 2012.

[8] A. Aswani, N. Master, J. Taneja, V. Smith, A. Krioukov, D. Culler, and C. Tomlin. Identifying models of HVAC systems using semiparametric regression. In *ACC 2012*, 2012.

[9] Tyler Barker, David C. Poole, M. Larry Noble, and Thomas J. Barstow. Human critical poweroxygen uptake relationship at different pedalling frequencies. *Experimental Physiology*, 91(3):621–632, 2006.

[10] Logan Beal, Daniel Hill, R Martin, and John Hedengren. Gekko optimization suite. *Processes*, 6(8):106, 2018.

[11] Moshe Ben-Akiva, Haris Koutsopoulos, Tomer Toledo, Qi Yang, Charisma Choudhury, Constantinos Antoniou, and Ramachandran Balakrishna. Traffic simulation with mitsimlab. *Fundamentals of Traffic Simulation*, 145, 2010.

[12] Souhila Benmakrelouf, Neila Mezghani, and Nadjia Kara. Towards the identification of players' profiles using game's data analysis based on regression model and clustering. In *ASONAM*, 2015.

[13] Geoff Boeing. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *CoRR*, abs/1611.01890, 2016.

[14] Elmar Brockfeld, Reinhart D. Khne, Alexander Skabardonis, and Peter Wagner. Toward benchmarking of microscopic traffic flow models. *Transportation Research Record*, 1852(1):124–129, 2003.

[15] Giorgio C. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Springer Science & Business Media, 3rd edition, 2011.

[16] Jean-Paul Calvi. Lectures on multivariate polynomial interpolation. `https://www.math.univ-toulouse.fr/~calvi/res_fichiers/MPI.pdf`, 2020. [Accessed 2021-02-17].

[17] Paris Carbone, Asterios Katsifodimos, Kth, Sics Sweden, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. Apache flink: Stream and batch processing in a single engine. *IEEE Data Engineering Bulletin*, 38, 2015.

[18] M. Chiang, E. Lim, W. Lee, and A. T. Kwee. Btci: A new framework for identifying congestion cascades using bus trajectory data. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1133–1142, 2017.

[19] L. Codeca, R. Frank, and T. Engel. Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research. In *2015 IEEE Vehicular Networking Conference (VNC)*, pages 1–8, 2015.

[20] Lara Codecá, Raphaël Frank, Sébastien Faye, and Thomas Engel. Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation. *IEEE Intelligent Transportation Systems Magazine*, 9(2):52–63, 2017.

[21] Lara Codec, Raphael Frank, and Thomas Engel. Lust: a 24-hour scenario of luxembourg city for sumo traffic simulations. In *2015 IEEE Vehicular Networking Conference (VNC)*, 05 2015.

[22] Erik Dassi and Alessandro Quattrone. DynaMIT: the dynamic motif integration toolkit. *Nucleic Acids Research*, 44, 2015.

[23] D-Link Inc. D-Link Wi-Fi Smart Plug DSP-W215. `us.dlink.com/products/connected-home/wi-fi-smart-plug/`, 2018. [Accessed 2018-02-19].

[24] Bing Dong and Khee Poh Lam. A real-time model predictive control for building heating and cooling systems based on the occupancy behavior pattern detection and local weather forecasting. In *Building Simulation*, volume 7, pages 89–106, 2014.

[25] Gilles Duranton and Matthew A. Turner. The fundamental law of road congestion: Evidence from us cities. *American Economic Review*, 101(6):2616–52, October 2011.

[26] Emerson Electric Co. . Sensi geofencing does not seem to be working for me. What can I do, and what are best practices? `https://sensi.emerson.com/en-us/support/sensi-geofencing-doesnt-seem-to-be-working-for-me-what-can-i-do/`, 2018. [Accessed 2020-06-24].

[27] Adam Galant, Ryszard Kutner, and Andrzej Majerowski. Heat transfer, newton's law of cooling and the law of entropy increase simulated by the real-time computer experiment in java. In *Computational Science — ICCS 2003*, ICCS, 2003.

[28] Esther Galbrun, Konstantinos Pelechrinis, and Evimaria Terzi. Urban navigation beyond shortest route: The case of safe paths. *Information Systems*, 57(Supplement C):160 – 171, 2016.

[29] Robert Geist and Stephen Daniel. A continuum of disk scheduling algorithms. *ACM Trans. Comput. Syst.*, 5(1):7792, January 1987.

[30] P.G. Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15, 1981.

[31] X. Gong, T. De Pessemier, W. Joseph, and L. Martens. A power data driven energy-cost-aware production scheduling method for sustainable manufacturing at the unit process level. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2016.

[32] Nathan Graves and Shawn Newsam. Visibility cameras: Where and how to look. In *MAED*, 2012.

[33] Chenjuan Guo, Yu Ma, Bin Yang, Christian S. Jensen, and Manohar Kaul. Ecomark: Evaluating models of vehicular environmental impact. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 269–278, 2012.

[34] Chenjuan Guo, Bin Yang, Ove Andersen, Christian S. Jensen, and Kristian Torp. Ecomark 2.0: empowering eco-routing with vehicular environmental models and actual vehicle fuel consumption data. *GeoInformatica*, 19(3):567–599, 2015.

[35] Joe Hicklin, Cleve Moler, and Peter Webb. JAMA : A Java Matrix Package. `https://math.nist.gov/javanumerics/jama/`, 2020. [Accessed 2020-06-24].

[36] Y. Iino, M. Murai, D. Murayama, I. Motoyama, S. Kuzusaka, and K. Ueta. Hybrid modeling with physical and jit model for building thermal load prediction and optimal energy saving control. In *2009 ICCAS-SICE*, pages 2008–2011, 2009.

[37] Institute of Transportation Systems. SUMO Documentation. `https://sumo.dlr.de/docs/SUMO_User_Documentation.html`, 2001. [Accessed 2020-01-06].

[38] Neelam Jangid, Snehanshu Saha, Anand Narasimhamurthy, and Archana Mathur. Computing the prestige of a journal: A revised multiple linear regression approach. In *WCI*, 2015.

[39] X. Kan, Lin Xiao, Hao Liu, Meng Wang, W.J. Schakel, Xiao-Yun Lu, B. Arem, Steven Shladover, and Robert Ferlis. Cross-comparison and calibration of two microscopic traffic simulation models for complex freeway corridors with dedicated lanes. *Journal of Advanced Transportation*, 2019:1–14, 03 2019.

[40] Shiva Prasad Kasiviswanathan, Kobbi Nissim, and Hongxia Jin. Private incremental regression. In *ACM PODS*, 2017.

[41] A. Kelman, Y. Ma, and F. Borrelli. Analysis of local optima in predictive control for energy efficient buildings. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, 2011.

[42] Boris Kerner. Experimental features of self-organization in traffic flow. *Phys. Rev. Lett.*, 81:3797–3800, Oct 1998.

[43] Boris S. Kerner. Congested traffic flow: Observations and theory. *Transportation Research Record*, 1678(1):160–167, 1999.

[44] Boris S Kerner. The physics of traffic. *Physics World*, 12(8):25–30, aug 1999.

[45] S. Krauß. Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics. Technical report, DLR Deutsches Zentrum fuer Luft- und Raumfahrt e.V., Koeln, 1998.

[46] Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailesh Mittal, Jignesh M. Patel, Karthik Ramasamy, and Siddarth Taneja. Twitter heron: Stream processing at scale. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015.

[47] L. Doman, U.S. Energy Information Administration. EIA projects 48 increase in world energy consumption by 2040. `www.eia.gov/todayinenergy/detail.php?id=26212`, 2016. [2018-02-19].

[48] L. Doman, U.S. Energy Information Administration. Use of Energy in the United States Explained, Energy Use in Commercial Buildings. `www.eia.gov/energyexplained/index.cfm?page=us_energy_commercial`, 2017. [Accessed 2018-02-19].

[49] S. Li, S. Ren, and X. Wang. Hvac room temperature prediction control based on neural network model. In *2013 Fifth International Conference on Measuring Technology and Mechatronics Automation*, pages 606–609, 2013.

[50] J. Lin, W. Yu, X. Yang, Q. Yang, X. Fu, and W. Zhao. A novel dynamic en-route decision real-time route guidance scheme in intelligent transportation systems. In *2015 IEEE 35th International Conference on Distributed Computing Systems*, pages 61–72, 2015.

[51] J. Lin, W. Yu, X. Yang, Q. Yang, X. Fu, and W. Zhao. A real-time en-route route guidance decision scheme for transportation-based cyberphysical systems. *IEEE Transactions on Vehicular Technology*, 66(3):2551–2566, 2017.

[52] Hao Liu, X. Kan, Steven Shladover, Xiao-Yun Lu, and Robert Ferlis. Modeling impacts of cooperative adaptive cruise control on mixed traffic flow in multi-lane freeway facilities. *Transportation Research Part C Emerging Technologies*, 95:261–279, 10 2018.

[53] Hao Liu, Xiao-Yun Lu, and Steven Shladover. Traffic signal control by leveraging cooperative adaptive cruise control (cacc) vehicle platooning capabilities. *Transportation Research Part C Emerging Technologies*, 104:390–407, 2019.

[54] Hao Liu, Xiao-Yun Lu, and Steven Shladover. Traffic signal cooperation for enhancing cooperative adaptive cruise control (cacc) vehicle string operations. In *The Transportation Research Board (TRB) 98th Annual Meeting*, 01 2019.

[55] Hao Liu, Heng Wei, Ting Zuo, Zhixia Li, and Y. Jeffrey Yang. Fine-tuning adas algorithm parameters for optimizing traffic safety and mobility in connected vehicle environment. *Transportation Research Part C Emerging Technologies*, 76:132–149, 2017.

[56] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Fltterd, R. Hilbrich, L. Lcken, J. Rummel, P. Wagner, and E. WieBner. Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582, 2018.

[57] Nicholas E. Lownes and Randy B. Machemehl. Vissim: A multi-parameter sensitivity analysis. In *Proceedings of the 38th Conference on Winter Simulation*, 2006.

[58] Xiao-Yun Lu, Hao Liu, X. Kan, Steven Shladover, and X. Lu. The impact of cooperative adaptive cruise control (cacc) vehicle string operation strategies on mixed traffic flow. In *TRB Annual Meeting*, 01 2018.

[59] Madge Tech. RHTemp101A. `ttps://www.madgetech.com/products/rhtemp101a/?utm_term=&utm_campaign=Product+Listing+Ads&utm_source=adwords&utm_medium=ppc&hsa_acc=1721077720&hsa_net=adwords&hsa_cam=7193463357&hsa_ad=391319204603&hsa_kw=&hsa_grp=80068617773&hsa_mt=&hsa_ver=3&hsa_src=g&hsa_tgt=pla-835403226372&gclid=EAIaIQobChMIn820vIzT6wIVrx6tBh0p3gM8EAQYASABEgKNXvD_BwE`, 2020. [Accessed 2020-06-24].

[60] Chris Mayfield, Jennifer Neville, and Sunil Prabhakar. Eracer: A database approach for statistical inference and data cleaning. In *SIGMOD*, 2010.

[61] Ronald T. Milam, Marc Birnbaum, Chris Ganson, Susan Handy, and Jerry Walters. Closing the induced vehicle travel gap between research and practice. *Transportation Research Record*, 2653(1):10–16, 2017.

[62] John Moy. OSPF Version 2. RFC 2328, April 1998.

[63] Debnath Mukherjee and Suman Datta. Incremental time series algorithms for IoT analytics: An example from autoregression. In *17th ICDCN*, 2016.

[64] Nest Inc. Nest Thermostats. `nest.com/`, 2016. [Accessed 2018-02-19].

[65] Kaylin T et al. Nguyen. Smartphone-based geofencing to ascertain hospitalizations. *Circulation: Cardiovascular Quality and Outcomes*, 10(3), 2017.

[66] G. De Nunzio, L. Thibault, and A. Sciarretta. Bi-objective eco-routing in large urban road networks. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7, 2017.

[67] European Research Center of the European Commission. Handbook emission factors for road transport v3, 2017.

[68] Los Angelis Department of Transportation. The automated traffic surveillance and controls (atsc) system.

[69] One Hour Heating and Air Conditioning. Short Cycling is a Big Problem. `https://www.onehourheatandair.com/articles/expert-tips/air-conditioners/short-cycling-is-a-big-problem/`, 2019. [Accessed 2020-06-24].

[70] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org`, 2017.

[71] Jiannan Ouyang. Busgazer - realtime pittsburgh transit application. `http://www.busgazer.com`.

[72] D. Petrov, R. Alseghayer, D. Mosse, and P. K. Chrysanthis. Data-driven user-aware hvac scheduling. In *IGSC*, 2018.

[73] Daniel Petrov, Rakan Alseghayer, Mohamed Sharaf, Panos K. Chrysanthis, and Alexandros Labrinidis. Interactive exploration of correlated time series. In *Proceedings of the ExploreDB'17*, pages 2:1–2:6, 2017.

[74] Philips Inc. Philips Hue Personal Wireless Lighting. `www2.meethue.com/en-us/`, 2018. [Accessed 2018-02-19].

[75] Kallirroi N. Porfyri, Evangelos Mintsis, and Evangelos Mitsakis. Assessment of acc and cacc systems using sumo. In *SUMO 2018- Simulating Autonomous and Intermodal Transport Systems*, volume 2 of *EPiC Series in Engineering*, pages 82–93, 2018.

[76] Prakash RANJITKAR, Takashi NAKATSUJI, and Akira KAWAMUA. Car-following models: An experiment based benchmarking. *Journal of the Eastern Asia Society for Transportation Studies*, 6:1582–1596, 2005.

[77] William J Ripple, Christopher Wolf, Thomas M Newsome, Phoebe Barnard, and William R Moomaw. World Scientists Warning of a Climate Emergency. *BioScience*, 11 2019.

[78] Alexander Schrijver. *Theory of Linear and Integer Programming*. "John Wiley and Sons, Inc.", 1986.

[79] Anatoli U. Shein, Panos K. Chrysanthis, and Alexandros Labrinidis. Slickdeque: High throughput and low latency incremental sliding-window aggregation. In *Proceedings of the 21th International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018.*, pages 397–408, 2018.

[80] Connected Signals. San jose connected vehicle pilot.

[81] Stephen Smith. Smart infrastructure for future urban mobility. *AI Magazine*, 41:5–18, 2020.

[82] Elahe Soltanaghaei and Kamin Whitehouse. Practical occupancy detection for programmable and smart thermostats. *Applied Energy*, 2017.

[83] Tamim I. Sookoor, Brian Holben, and Kamin Whitehouse. Feasibility of retrofitting centralized HVAC systems for room-level zoning. In *IGSC*, 2012.

[84] Hamdy A. Taha. *Operations Research: An Introduction (8th Edition)*. Prentice-Hall, Inc., 2006.

[85] The Weather Underground, Inc. Historical Weather. `https://www.wunderground.com/history`, 2019. [Accessed 2018-11-10].

[86] Mikkel Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *J. ACM*, 46(3):362–394, May 1999.

[87] Abhishek Tiwari, Brian Ramprasad, Seyed Hossein Mortazavi, Moshe Gabel, and Eyal de Lara. Reconfigurable streaming for the mobile edge. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, 2019.

[88] Martin Treiber and Dirk Helbing. Memory effects in microscopic traffic models and wide scattering in flow-density data. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 68, 11 2003.

[89] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62:18051824, Aug 2000.

[90] Shivaram Venkataraman, Aurojit Panda, Kay Ousterhout, Michael Armbrust, Ali Ghodsi, Michael J. Franklin, Benjamin Recht, and Ion Stoica. Drizzle: Fast and adaptable stream processing at scale. In *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017.

[91] Pu Wang, Ruiquan Ge, Xuan Xiao, Manli Zhou, and Fengfeng Zhou. hmulab: A biomedical hybrid multi-label classifier based on multiple linear regression. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 14(5), 2017.

[92] Axel Wegener, Michal Piorkowski, Maxim Raya, Horst Hellbrck, Stefan Fischer, and Jean-Pierre Hubaux. Traci: An interface for coupling road traffic and network simulators. *11th Communications and Networking Simulation Symposium (CNS)*, 2008.

[93] Mingjun Wei and Yu Meng. Research on the optimal route choice based on improved dijkstra. In *2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)*, pages 303–306, Sept 2014.

[94] Hansong Xu, Jie Lin, and Wei Yu. *Smart Transportation Systems: Architecture, Enabling Technologies, and Open Issues*, pages 23–49. Springer Singapore, 2017.

[95] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM SIGKDD 2011, pages 316–324, 2011.

[96] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: Driving directions based on taxi trajectories. In *ACM SIGSPATIAL 2010*, pages 99–108, 2011.

[97] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, and et al. Apache spark: A unified engine for big data processing. *Commun. ACM*, 59, 2016.