Title
Learning and Optimization for Mixed Autonomy Systems - A Mobility Context

Author
Wu, Cathy

Publication Date
2018

Peer reviewed|Thesis/dissertation

Learning and Optimization for Mixed Autonomy Systems - A Mobility Context

by

Cathy Wu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Alexandre M. Bayen, Chair
Professor Pieter Abbeel
Professor Ruzena Bajcsy
Professor Claire Tomlin
Professor-in-Residence Alexander Skabardonis
Dr. Eric Horvitz

Fall 2018

Learning and Optimization for Mixed Autonomy Systems - A Mobility Context

Abstract

Learning and Optimization for Mixed Autonomy Systems - A Mobility Context

by

Cathy Wu

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Alexandre M. Bayen, Chair

Mixed autonomy characterizes problems surrounding the gradual and complex integration of automation and AI into existing systems. In the context of mobility, we consider: how will the gradual introduction of self-driving cars change urban mobility? In this dissertation, we develop machine learning and optimization techniques to address three key challenges: 1) quantifying the behavior of such complex systems, 2) addressing inherent sensing limitations, and 3) mitigating negative effects of introducing the automation.

We demonstrate that deep reinforcement learning (RL) can serve as a unifying framework for studying the behavior of disparate and complex scenarios common in mixed autonomy systems. In particular, using deep RL, we find that automating a small fraction of vehicles in various traffic scenarios can result in a significant system-level velocity increase and numerous emergent driving behaviors. We demonstrate through the development of variance reduction techniques for policy gradient methods, that deep RL has the potential to scale to high-dimensional control systems, such as traffic networks and other mixed autonomy systems. We additionally present Flow, an open source RL platform with the goal of easing the design and study of disparate traffic scenarios. To address sensing limitations inherent when only parts of a system are automated, sensor fusion is explored. In particular, we introduce a convex optimization method for cellular network measurements from AT&T at the scale of the Greater Los Angeles Area, to address a flow estimation problem previously believed to be intractable. Finally, when automation reduces the cost of the activity (of transport), anticipated negative effects include induced demand and increased energy consumption. We study how the design of the mobility system itself can mitigate these effects. In particular, joint work with Microsoft Research provides insight into how high-occupancy vehicle lanes can simultaneously satisfy comfort and time preferences of users, and provide system benefits. We introduce combinatorial optimization methods based on clustering and local search for the resulting ridesharing problem. Together, these learning and optimization methods demonstrate that a small number of vehicles and sensors can be harnessed for significant impact on urban mobility, and shed light into the future study of mixed autonomy systems.

ACKNOWLEDGMENTS

Because I knew you
I have been changed for good.

Stephen Schwartz,
For Good, Wicked (2003)

I am extremely fortunate to be advised by Alexandre Bayen throughout my graduate research. His constant support and advice is what made this thesis possible. I am astounded at his ability to place problems in many different contexts all at once–technical, domain, social, systemic–and it is through his guidance that I came to recognize the richness of *any* problem.

Thanks especially also to Eric Horvitz for mentoring me at Microsoft Research and ever since, and whose energy and thoughtful approach to people and AI both will always inspire me; and to Pieter Abbeel for mentoring me as I sought to introduce reinforcement learning to my own research, and whose calm and effectiveness I will always strive for.

I am grateful also for Daniela Rus, Seth Teller, and Jim Glass at MIT, who mentored me as an undergraduate researcher and Masters student and whose encouragement led me to pursue my dreams.

I am fortunate to have worked and learned with my amazing collaborators: Alexandre Bayen, Jerome Thai, Alexei Pozdnukhov, Steven Yadlowsky, K. Shankari, Christos Papadimitriou, Eric Horvitz, Ece Kamar, Eugene Vinitsky, Aboudy Kreidieh, Leah Dickstein, Kanaad Parvate, Nishant Kheterpal, Ankur Mehta, Pieter Abbeel, Yan Duan, Aravind Rajeswaren, Igor Mordatch.

Thanks to Pieter Abbeel, Claire Tomlin, Laurent El Ghaoui, Ruzena Bajcsy, Alexander Skabardonis, and Eric Horvitz for serving on my qualifying exam and dissertation committees – I have cherished each of our conversations, and your advice and feedback have greatly improved the research in this dissertation. I am grateful also for Microsoft Research, Google X, and OpenAI, for providing me perspective and guidance over the years. I thank also the National Science Foundation and the Berkeley Chancellor's fellowships for funding my PhD and for the freedom to explore important problems.

I want to thank the undergraduate, Masters, and graduate students and research assistants who journeyed with me and trusted in my mentorship: Steven Yadlowsky, Lei Du,

Chenyang Yuan, Leah Dickstein, Kanaad Parvate, Nathan Mandi, Aziz Khiyami, Eugene Vinitsky, Aboudy Kreidieh, Nishant Kheterpal, Ananth Kuchibhotla, and Luc Le Flem.

I had the great fortune of starting this journey with my cohort and great friends in *control, intelligent systems, and robotics*–Eric Kim, Jaime Fisac, Roel Dobbe, and Jason Poon–with whom I gained a deep appreciation for continuous thinking and control. What started out as a mere class project on compressed sensing and traffic assignment–with Philipp Moritz, Richard Shin, Fanny Yang–turned into countless discussions and unceasing explorations in optimization and statistics. I have tremendously enjoyed the many invigorating discussions and refreshing perspectives on transportation, with Alexander Skabardonis, Alexei Pozdnukhov, and Teddy Forscher. I am grateful for Yang Ruan, Mark Tobenkin and Michael Pihulic, who have pushed me to consider the societal impact, ethics, and the bigger picture, and who have always been there for me.

Both in and outside of research, there are just so many people not yet mentioned, who have enriched my life, supported me, and inspired me over the last few years. I am so grateful for: Marie McGraw, Pranjal Vachaspati, Alex Lee, Mo Chen, Timothée Hunter, Dorsa Sadigh, Katie Driggs-Campbell, Roy Dong, Jacob Steinhardt, Dan Work, Samitha Samaranayake, Walid Krichene, Jack Reilly, Kene Akametalu, Pavel Panchekha, Mark Velednitsky, Kevin Fischer, Geza Kovacs, Aviv Ovadya, Lei Du, Vrajesh Modi, Rishi Gupta, Michael Scarito, Ahmed El Alaoui, Lillian Ratliff, Sandy Huang, Eric Tzeng, Chelsea Finn, Aude Hofleitner, Constantin Berzan. I am grateful also for my sister Nancy, who is the most creative and understanding person I know, and my brother Joseph, who always has a spare hour or ten for some computer games.

My home for five years, a huge house in South Berkeley called *Little Mountain*, was the ultimate mix of calm and impetus–sparking intellectual discussions, dry ice parties, spontaneous sing-alongs, pull-up contests, and countless memories. I am grateful to Rishi Gupta for founding this wonderful home and for my amazing housemates for simply being there. I am so happy for all the fun times at Total Athletic Conditioning (TAC) led by coach Mark Jellison and the Cal Yongmudo Club, where I am especially grateful for instructors Randy Vogel and Norman Link – for not only helping to keep me sane, but also for bringing me closer with friends in the department.

Thanks to the wonderful staff within EECS and ITS, whose hard work keeps the research machine going–in particular, Shirley Salanio, Jessica Gamble, Helen Bassham, Jeanne Marie Acceturo, and Rosita Alvarez-Croft.

Finally, I dedicate this dissertation to my parents, Gody and Jeb Wu, who always wanted a doctor in the family, and whose love and support made everything possible.

CONTENTS

1 INTRODUCTION 1
- 1.1 Mixed automated and human decision making 1
- 1.2 Motivating examples of mixed autonomy systems 3
- 1.3 How will automated vehicles change mobility? 4
- 1.4 Mixed autonomy systems 6
- 1.5 Thesis overview and contributions 8

2 REVIEW OF OPTIMIZATION FRAMEWORKS 12
- 2.1 Convex optimization 12
- 2.2 Reinforcement learning 15
- 2.3 Combinatorial optimization 18

3 REVIEW OF AUTOMATED TRANSPORTATION 23
- 3.1 Safety first 24
- 3.2 Freeing the freeway 27
- 3.3 Confronting the last mile 29
- 3.4 Taking to the streets 30
- 3.5 Full speed ahead 32

I CONTROL 34

4 EMERGENT BEHAVIORS IN MIXED AUTONOMY TRAFFIC 35
- 4.1 Overview 36
- 4.2 Preliminaries 39
- 4.3 Mixed autonomy traffic as reinforcement learning 41
- 4.4 State equivalence classes 43
- 4.5 Network configurations 44
- 4.6 Mixtures of autonomy 45
- 4.7 Metrics 46
- 4.8 Emergent behaviors 47
- 4.9 Related work 51
- 4.10 Chapter summary 54
- 4.11 Experiment details 55

5 VARIANCE REDUCTION FOR POLICY GRADIENT WITH ACTION-DEPENDENT FACTORIZED BASELINES 56

5.1 Overview 57
5.2 Preliminaries 58
5.3 Action-dependent baselines 60
5.4 Experiments and Results 64
5.5 Related works 68
5.6 Chapter summary 70
5.7 Derivation of the optimal state-dependent baseline 70
5.8 Derivation of the optimal action-dependent baseline 71
5.9 Derivation of variance reduction improvement 73
5.10 Derivation of suboptimality of the optimal state-dependent baseline 75
5.11 Baselines for general actions 75
5.12 Compatibility with GAE 76
5.13 High-dimensional action spaces: training curves 78
5.14 Experiment details 78
6 FLOW: A LIBRARY FOR REINFORCEMENT LEARNING AND MICROSIMULATION 81
6.1 Overview 82
6.2 Preliminaries 86
6.3 Flow 87
6.4 Networks 90
6.5 Task space 92
6.6 Controller design case study: mixed autonomy ring 93
6.7 Related work 100
6.8 Chapter summary 103
6.9 Classical controllers 103
6.10 Additional experiments 107
6.11 Fail-safes 108
II STATE ESTIMATION 112
7 CELLPATH: FUSION OF CELLULAR AND TRAFFIC SENSOR DATA FOR ROUTE FLOW ESTIMATION VIA CONVEX OPTIMIZATION 113
7.1 Overview 114
7.2 Problem formulation 121
7.3 Dimensionality reduction and projection via isotonic regression 128
7.4 Experimental setting and validation process 133
7.5 Numerical results 141
7.6 Chapter summary 146

III SYSTEM DESIGN 148
8 HUMAN MOBILITY PREFERENCES 149
8.1 Overview 150
8.2 Induced demand 150
8.3 Methodology and data collection 151
8.4 Findings 152
8.5 Recommendations for ridesharing systems 157
8.6 Chapter summary 159
8.7 Mobility preference survey questions 159
9 OPTIMIZING THE DIAMOND LANE: COMPLEXITY AND ALGORITHMS FOR RIDESHARING 165
9.1 Overview 166
9.2 Survey of complexity results in ridesharing 170
9.3 The carpool problem 170
9.4 Problem formulation 176
9.5 Methods for solving the carpool problem 182
9.6 Warm-starting the local search methods 186
9.7 Numerical implementation 188
9.8 Numerical results 191
9.9 Chapter summary 193
9.10 Scalability: time breakdown for local search 194
9.11 Initialization for local search methods 195
10 CLUSTERING FOR SET PARTITIONING WITH A CASE STUDY IN RIDESHARING 201
10.1 Overview and combinatorial optimization problems 201
10.2 Set partitioning 202
10.3 A formal connection between clustering and set partitioning 204
10.4 Case study: ridesharing meetup problem 209
10.5 Chapter summary 210

IV FINAL REMARKS 212
11 THE ROAD AHEAD 213
11.1 Challenges in mixed autonomy 213
11.2 Opportunities in mixed autonomy 215

1

INTRODUCTION

> It is change, continuing change, inevitable change, that is the dominant factor in society today. No sensible decision can be made any longer without taking into account not only the world as it is, but the world as it will be...
>
> Isaac Asimov,
> Asimov on Science Fiction (1981)

1.1 MIXED AUTOMATED AND HUMAN DECISION MAKING

Current trends in computing and artificial intelligence give individuals, engineers, corporations, and governments an unprecedented capability to introduce automation into many aspects of our lives, from online services to our offline lives and our workplaces. Online, we engage with video and music recommendations; crossing into offline, we engage with restaurant recommendations and mapping services; in our workplaces, we see decision support systems for clinical decisions, traffic operations, etc. As seen by a multitude of articles pertaining to automation of work, unintended consequences of automation, and AI safety, the integration of automation into existing systems has complex and poorly understood effects, both near- and far-term. We term *mixed autonomy* as the problem area concerning the gradual integration of automation into existing systems, which often are large-scale, are complex in themselves, and involve many humans. The resulting system consists of a mixture of automated and human decision making, hence mixed autonomy. This designation indicates that the automated and human-decision

making aspects cannot be studied in isolation because each affects the other.

Learning, control, and mixed autonomy systems. To a large extent, automation using machine learning thus far has investigated the paradigm of "static learning". In this paradigm, data is collected from a process and is assumed to be independent and identically distributed (i.i.d.) from a fixed (static) but unknown distribution. An output measure is determined, a classifier or similar is developed to minimize the empirical risk, and if the world were to remain the same, akin to offline evaluation, then the classifier would perform as expected. However, in practice, the mere introduction of the classifier or any form of automation into the "wild" (open world) may drastically shift the underlying data distribution. This phenomenon is also called behavioral drift and may result from the altered behavior of interacting humans or other agents. Technically, the distributional shift results in a violation of the i.i.d. assumption of the data and thus the classifier may have unintended behaviors. The same story holds for control theory for sequential decision making. A controller is designed and calibrated to system models based on data collected from a natural process. The introduction of the controller into the open world, however, may again shift the underlying data distribution, thereby voiding the model calibration. The controller, operating in the resulting system, could yield unsafe or otherwise undesirable behaviors. While robust control alleviates these issues, standard practices are suitable only for handling small distributional shift. The result, whether due to automation by machine learning, control, or a combination thereof, is a mixed autonomy system. A science and engineering of mixed autonomy systems is required to ensure the continued performance of our systems amidst the introduction of automation.

A recent emergence of mixed autonomy systems. Automation has been a boon for civilization, from the autopilot in commercial aircraft to automated production lines in manufacturing. At the same time, it is possible that to date we have largely experienced distributional shift that is limited or sufficiently slow. This can be explained by the introduction of relatively isolated automation into society thus far. For instance, in an air traffic control system, for safety reasons there is ample spatial distance between the agents (including aircraft and weather sensing equipment), whether operated manually or by an autopilot. As such, it is possible that the effects of autopilot are largely contained to the aircraft itself, rather than imposing strong dynamics on the rest of the air traffic system. Similarly, in manufacturing, factories often install heavy automation equipment behind barriers or marked lines on the factory floor, as to limit its interaction with operators and other agents. A single engineer using a spam classifier on her own email inbox may also have limited systemic effects. However, the introduction of automation

into the open world may exhibit complex effects on both humans and other automated components. Moreover, in systems where agents (humans) are freely able to choose to adopt and use automation, the resulting distributional shift of the system dynamics may be especially difficult to limit. A characterization of the potential effects of introduced automation is an important direction of future research and is out of scope for this thesis. Following, are a few important motivating examples of mixed autonomy systems.

1.2 MOTIVATING EXAMPLES OF MIXED AUTONOMY SYSTEMS

Mobility System. A city with 5% adoption of self-driving cars is a mixed autonomy system. Studying the characteristics such as system throughput and efficiency of a system with no self-driving cars or 100% self-driving cars is distinct from that of a mixed autonomy system. The no-autonomy setting, although already challenging, is purely a modeling problem of current mobility systems. The 100% case is fictional (at the time of writing) but can be modeled by domain experts, and, with some restrictions, the control problem reduces to a coordination problem. The in-between setting, requires careful consideration of the interactions between the human and automated actors, as well as with the rest of the system. It is known that up to 30% of traffic jams are caused by deficiencies in human driving. Scalable computational tools for studying mixed autonomy systems could help determine whether automated agents will augment or correct these deficiencies.

Energy system. A power grid with 1% distributed devices which can choose to supply energy back to the grid, such as electric vehicles and refrigerators. The "duck curve", a so-called curve to describe the cyclic daily energy demand curve with a dip during mid day and a peak in the evening hours reflecting the aggregate behavior of the millions of human users, is a major challenge for the adoption of renewable energy, for which energy supply is much more challenging to control. Scalable computational tools for studying mixed autonomy systems could help determine whether a small fraction of controlled energy devices could help stabilize the overall network.

Social network. A social network with 1% of users which are "bots" that issue actions such as sharing articles and rating videos. Mixed autonomy systems techniques could help determine the effects of automation on the spread of public service announcements, "viral" content, or misinformation.

Criminal justice. A nation with 7% of local county courts using automated recidivism scores of criminals to determine sentences is a mixed autonomy system. Mixed auton-

omy could provide a perspective on long-term recidivism dynamics.

Financial systems. In the global financial system, 0.01% of monetary volume is exchanged via high-speed trading algorithms and systems, in an ecosystem with predominantly human traders. Mixed autonomy could help detect unintended effects of the complex interactions between the automated and non-automated components of our financial markets.

In each of the above settings, automation is introduced into an existing large-scale and already complex system. Although there is strong potential for benefit, the effects of the automation are unknown; for instance, the automation could magnify existing limitations in human behavior, such as bias or imprecise control, or achieve a desirable system-level outcome. Moreover the increase or decrease of automation over time may be dictated by a number of external factors, such as metrics of evaluation and market forces.

1.3 How will automated vehicles change mobility?

A running example throughout this thesis concerns the complex integration of automated vehicles into existing mobility systems, which we term *mixed autonomy mobility*. Mobility is a natural example because it is an existing system in which many human agents interact closely and regularly with many automated agents, such as traffic lights. An upcoming and long anticipated event is the introduction of automated vehicles into the mobility system. This example helps identify and highlight a number of new scaling challenges which push the limits of our optimization frameworks, as exemplified in this thesis.

Transportation systems today form a literal physical backbone to civilization. Daily, they touch the lives of 3.9 billion people who reside in urban areas or 54% of the world population (United Nations, 2014) and support 1.32 billion motor vehicles worldwide[1] (WardsAuto, 2017). At the same time, the transportation sector accounting for 29% of energy consumption in the US (see Figure 1) (US Energy Information Administration, 2018, Table 2.1) and more than 20% of energy-related global GHG emissions world-wide.

Self-driving vehicles are slated to bring about dramatic changes in terms of energy consumption, safety, access and time savings. They can affect energy consumption in a variety of ways, including though vehicle platooning, eco-driving (Gense, 2000), and many others, as summarized in Figure 2. In particular, the reduction in travel cost could

1 This measure, from 2016, excludes motorbikes.

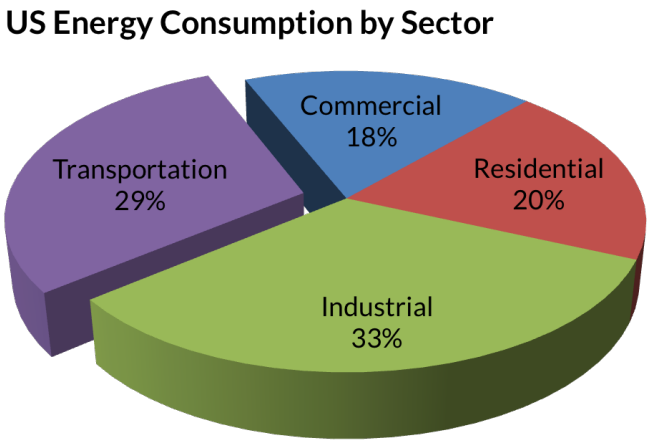


Figure 1: Energy consumption in the United States by sector (April 2018). Source: US Energy Information Administration (2018, Table 2.1).

result in a 5-60% increase in energy consumption. Depending on weighted likelihoods of each of these factors, studies determined that with full adoption of self-driving cars, the US mobility system could see anywhere from a minus 40% to a doubling of energy consumption (Wadud et al., 2016). In short, we have a great amount of uncertainty with regards to the impact of automation, even for a single metric (energy, in this case), let alone a holistic measure. How do we even start to reason about all these factors? And then shape the outcome?

This dissertation introduces several new machine learning and optimization techniques that are needed to help guide the evolution of urban mobility in light of the adoption of automated vehicles. We characterize the study of the integration of automated vehicles into existing mobility systems as *mixed autonomy mobility*.

A note on vehicle automation. In the spirit of near-term practicality, this thesis aims to be compatible with a variety of forms and levels of vehicle automation. Vehicles may be automated in a variety of ways, including a dedicated driver such as in a taxi, an instructed or incentivized driver, or an electro-mechanical system such as those designed for a self-driving vehicle. Levels of automation vary as well, from partial automation in the forms of cruise control, braking or parking assistance, routing guidance, or trip management to full automation of the driving or trip related tasks. Although there are many other interesting forms of automation in mobility systems, such as traffic lights, ramp meters, variable speed limits and other electronic roadway signage, mobility usage fees, congestion pricing, and road directionality, the study of their integration into the

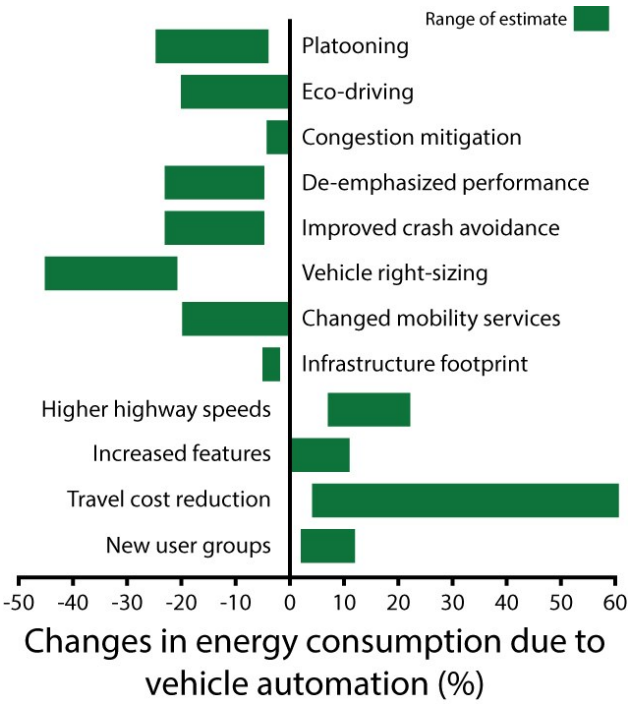


Figure 2: Automating transportation will not necessarily make transportation more efficient—some estimates even show that energy consumption could increase once vehicles are fully automated. In particular, the reduced cost of traveling may incentivize and enable more people to travel. Everything from future adoption rates of automated vehicles to government regulations will influence the net impact of automated transportation on transit-related energy consumption. Source: Wadud et al. (2016).

mobility system are out of scope for the present manuscript. However, the presented approaches may extend readily to many of these forms of automation.

1.4 MIXED AUTONOMY SYSTEMS

Mixed autonomy systems, that is dynamical systems which involve the interaction of automated and human actors (possibly dynamic in themselves), exhibit several attributes and challenges which push the limits of state-of-the-art methods.

- **Scale and heterogeneity**. Mixed autonomy systems may consist of hundreds or even millions of heterogeneous interacting agents, each of which have different utility functions and constraints. For instance, automated vehicles, non-automated vehicles, traffic lights, electronic signage, pedestrians, bicyclists, etc. each may have

different preferences, origins, destinations, and time bounds.

- **Complex dynamical system**. The overall mixed autonomy system is a highly stochastic, cascaded, nonlinear, discontinuous, hybrid, networked dynamical system, in which actions of an individual agent or a population may induce complex and delayed effects on the system, such as traffic or demand.
- **Limited actuators**. The gradual integration or adoption of automation (e.g. automated vehicles) into the existing system implies that the number or fraction of physical actuators may be small.
- **Limited sensors**. Often, actuators double as sensors, so in a mixed autonomy setting, the existing systems may have varied but limited sensing capabilities.
- **Limited human behavior models**. Human behavior and reasoning are challenging to model, as well as their interactions with automated decison making components of the system. The problem is further exacerbated in settings which humans have not yet experienced, such as with the introduction of new automation (e.g. automated vehicles).
- **External process governing automation integration**. There may be a process external to the mixed autonomy system which governs the dynamics of the automation integration. In the case of mobility, for instance, vehicle ownership models, service models, competition, and economic incentives may dictate the progression of adoption and use of automated vehicles.

Key questions. In light of the challenges in mixed autonomy, key questions of interest include:

- What are the capabilities of a mixed autonomy mobility system? How does a mobility system with mixed automated and non-automated vehicles behave differently from a non-automated mobility system?
- Given the system heterogeneity and complexity, what are the tools and techniques suitable for systematically studying the capabilities and limitations of mixed autonomy systems?
- What are the requirements in terms of sensing, computation, incentives, and infrastructure for enabling mixed autonomy mobility systems?
- What are the potential impacts of mixed autonomy systems, both positive and negative?

This thesis is a first step in answering these questions. Many additional questions and research areas of interest to mixed autonomy are discussed in Chapter 11. The study of mixed autonomy will require many more contributions from optimization, machine learning, computer systems, robotics, control theory, algorithms, economics, psychology,

and domain specific social sciences and engineering disciplines. A science and a theoretical foundation for the understanding of mixed autonomy and a corresponding engineering practice for their design are required to ensure long-term desirable performance of our existing systems, in particular as automation is introduced.

Automation science and engineering. The study of mixed autonomy aims to advance automation science and engineering, which is about developing methodology for efficiency, quality, productivity, and reliability in automation systems; it draws from computer science, control systems, electrical engineering, mathematics, mechanical engineering, operations research, among other fields. Mixed autonomy requires advances beyond the current methods, including important areas such as computational scalability and handling complex models. Recent advances in function approximation and representation learning indicate that reinforcement learning has the potential to overcome both challenges. At the same time, reinforcement learning is ill-suited for problems which benefit from exhaustive search, such as problems concerning safety, constraints, and many combinatorial optimization problems. Thus advances in combinatorial optimization would supplement reinforcement learning in the combinatorial aspects of studying mixed autonomy. This thesis places emphasis on challenges in optimization scalability for enabling the science and engineering of mixed autonomy and concludes with a discussion on the broad challenges in mixed autonomy.

1.5 THESIS OVERVIEW AND CONTRIBUTIONS

This thesis introduces, develops scalable computational tools (including algorithms and systems), and addresses concrete challenges in mixed autonomy systems. We begin with a review of the common and powerful optimization frameworks of reinforcement learning, convex optimization, and combinatorial optimization. We briefly review the history, research, challenges, and opportunities of automation in mobility systems. An early version of the historical overview was published as Wu (2016). Then, what follows is the structure and primary contributions of the thesis, as diagrammed in Figures 3 and 4.

Part i: Control. The first part of the thesis first demonstrates the existence of mixed autonomy systems in the open world, showing that a mixture of automated and human-driven vehicles may yield vastly different system characteristics, such as average velocity of all vehicles in the system, as compared to a system with only human-driven vehicles. Termed mixed autonomy traffic, this serves as a running example throughout the thesis for concretely demonstrating the problems and challenges of mixed autonomy systems.

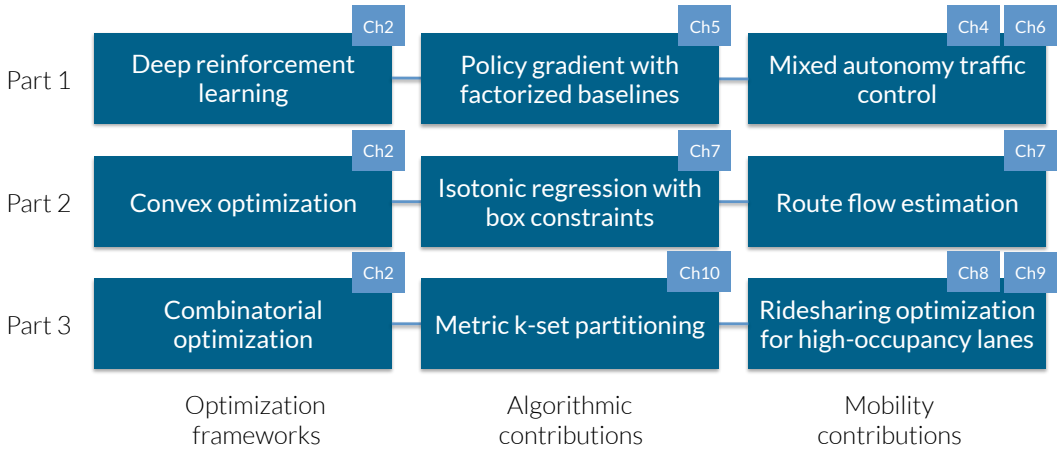


Figure 3: Summary of contributions of the thesis, in terms of optimization frameworks, specific optimization methods, and mobility challenges.

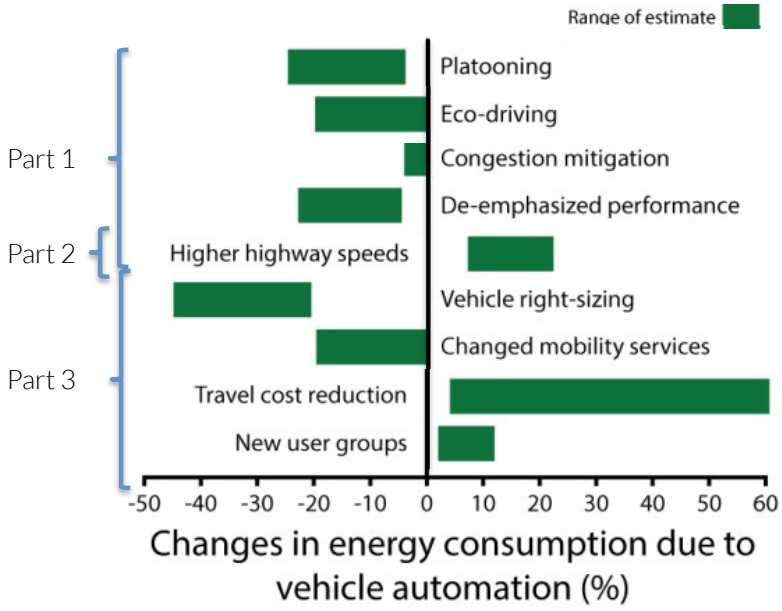


Figure 4: A categorization of the dissertation, based on coverage of the AV impacts on energy consumption, as shown in Figure 2.

Simultaneously, we demonstrate the potential of machine learning methods for studying mixed autonomy systems, in contrast to classical techniques based on partial differential equations and manual controller design. In particular, by casting the problem into the framework of model-agnostic reinforcement learning, it is established that a small

fraction of automated vehicles has the potential to dramatically improve overall road velocities for all vehicles, and a number of interesting driving behaviors emerge. Many advances are needed to enable mixed autonomy systems at the scale of thousands or even millions of agents, including high-speed distributed simulation and algorithmic developments. To this end, we present generic deep reinforcement learning techniques for scaling up to higher dimensional control problems, such as controlling many vehicles in a mobility system. Additionally, an open-source library is introduced which allows for integrated studies of reinforcement learning and traffic microsimulation, with scalable distributed simulation and cloud deployment. The work in this part was published in Wu et al. (2017d), Wu et al. (2018), and Wu et al. (2017c). The contributions of this part have implications for the environment and public policy concerning the regulation of automated vehicles, as well as scalable reinforcement learning.

Part ii: State estimation. The second part of the thesis explores the sensing challenges and requirements to enable mixed autonomy systems. In mixed autonomy systems, only parts of the system are automated and thus naturally observed, but information about the other parts of the system may be required in order to achieve optimal system performance. Thus being able to measure or estimate relevant quantities is critical to the performance of mixed autonomy systems. In mobility, owing to control theoretic analysis and advances in transportation engineering, vehicle throughput is such a critical quantity to estimate, but has classically been hindered by sparse sensing of the transportation infrastructure. By casting the problem into the framework of convex optimization, we determined that currently available transportation sensing infrastructure, augmented with also currently available aggregate data from cellular networks enables accurate throughput estimation. The structure of cellular network data and the large scale of urban systems motivates the design of a new algorithm for projected gradient descent with a block simplex constraint. The work in this part was published in Wu et al. (2015). The contributions of this part have implications for near-term transportation management, as opposed to long-term planning such as land-use planning, and infrastructure design, as well as scalable convex optimization.

Part iii: System design. The third part of the thesis explores the design of the system itself, to mitigate the potential effects of the external process on the system. Mixed autonomy systems may be viewed as being embedded within another dynamical system, one which dictates the progression of the integration, adoption, and use of automation. This process, external to the mixed autonomy system, may induce substantial effects upon the system, both positive and negative. Mobility is embedded in an overall socioeconomic system, and one major anticipated long-term impact of automated vehicles is induced de-

mand, in which more people travel in response to the newly available roadway capacity (enabled in Part i). This additional demand on the mobility system may compromise the benefits in road velocity and throughput by resulting in elevated energy consumption. We start by empirically studying the dynamics of the overall socioeconomic system and in particular, its couplings with the mobility system. To this end, we build a model of human mobility preferences based on a user study of 300 employees at a major technology corporation. We identify ridesharing as a promising approach and give it treatment as a design paradigm for the mobility system itself, with the goal of mitigating the effects of induced demand by dramatically improving the throughput of the mobility system. We conclude therefore that, with lightly modified existing infrastructure, ridesharing has the potential to dramatically improve (nearly triple) the throughput of the mobility system, and provide combinatorial algorithms to solve the allocation. The structure of the ridesharing problem motivates the adaptation of clustering algorithms from machine learning for set partitioning in the combinatorial optimization framework. The work in this part was published in Wu et al. (2016b) and Wu et al. (2016a). The contributions of this part have implications for mobility system design, urban planning, and public policy, as well as scalable combinatorial optimization.

Discussion and closing remarks. This thesis is only a first step in enabling a science and engineering of mixed autonomy systems, and takes the approach of introducing the broad definition and challenges of mixed autonomy systems, developing scalable computational tools, and addressing concrete challenges, in particular, in mobility. We therefore close with a discussion of the road ahead. We discuss remaining research questions and system requirements for deployment. We present also directions of future research for mixed autonomy systems.

2

REVIEW OF OPTIMIZATION FRAMEWORKS

> It is our choices, Harry, that show what we truly are, far more than our abilities.
>
> J.K. Rowling, Harry Potter and the Chamber of Secrets (1998)

Many situations arise in mixed autonomy systems where we could like to optimize the value of some function. That is, given a function $f : \mathbb{R}^n \to \mathbb{R}$, we want to find $x \in \mathbb{R}^n$ that minimizes (or maximizes) $f(x)$. Many problems in control, estimation, and system design can all be framed as optimization problems. This chapter presents a several common and powerful optimization frameworks used in this thesis–convex optimization, reinforcement learning, and combinatorial optimization. Each of these frameworks has seen extensive empirical and theoretical results. These frameworks solve vastly different and yet each gigantic classes of optimization problems. It is key to identify when a framework is appropriate for a problem at hand; this thesis utilizes these disparate optimization frameworks to address important problems within mixed autonomy systems in the context of mobility.

2.1 CONVEX OPTIMIZATION

In the general case, finding the global optimum of a function can be a very difficult task. However, for a special class of optimization problems known as convex optimization problems, we can efficiently find the global solution in many cases. Here, "efficiently" has both practical and theoretical connotations – it means that we can solve many real-world problems in a reasonable amount of time, and it means that theoretically we can solve problems in time that depend only polynomially on the problem size. Importantly,

and in fact, for convex optimization problems, any local optimum is also necessarily a global optimum. This section serves as preliminary material for Part ii of this thesis.

Convex optimization can be described as a fusion of three disciplines: optimization, convex analysis, and numerical computation. It has recently become a tool of central importance in engineering, enabling the solution of very large, practical engineering problems reliably and efficiently. Successful applications of convex optimization have been seen in a wide variety of areas, including control, signal processing, networks, circuit design, communication, information theory, computer science, operations research, economics, statistics, structural design. Much of the challenge, and art, in using convex optimization is in recognizing and formulating the problem. For an excellent text and overview on convex optimization, see Boyd and Vandenberghe (2004); for an excellent text and comprehensive account on numerical optimization, see Nocedal and Wright (2006). For a survey of convergence results and algorithms in convex optimization, we refer the reader to Bubeck et al. (2015).

A convex optimization problem consists of a convex function objective, inequality constraints, and equality constraint. It can be written as:

$$\begin{aligned} &\text{minimize} && f_0(x) && (1)\\ &\text{subject to} && f_i(x) \leqslant b_i, \quad i = 1, \dots, m && (2)\\ &&& h_i(x) = 0, \quad i = 1, \dots, p, && (3) \end{aligned}$$

where the functions $f_0, \dots, f_m : \mathbb{R}^n \to \mathbb{R}$ are convex, i.e. satisfy

$$f_i(\alpha x + \beta y) \leqslant \alpha f_i(x) + \beta f_i(y) \tag{4}$$

for all $x, y \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}$ with $\alpha + \beta = 1, \alpha \geqslant 0, \beta \geqslant 0$. Note also that the functions $h_0, \dots, h_p : \mathbb{R}^n \to \mathbb{R}$ are affine, i.e. can be written as a function of the form $x \mapsto a_i^T x + b_i$, where a_i is a linear transformation on x and b_i is a scalar.

For the design of a convex optimization method, some of the important issues that have to be dealt with are: how to descend the objective function, and how to remain feasible, i.e. satisfy the constraints. In the following, we review important classes of convex optimization problems, general descent methods, and general methods for handling constraints.

Unconstrained convex optimization. Iterative convex optimization methods take advantage of the local information $f(x_k)$ for the current iterate x_k. Common examples include using gradient information from the first- or second-order Taylor series approximation

to $f(x_k + p)$ for some search direction $p \in \mathbb{R}^n$; these are also called the steepest descent method and Newton's method, respectively. Of critical importance is determining how far to step in a descent direction, and these are commonly addressed by line search methods, which will look for the point in the direction of a search direction that minimizes the objective.

We often have constraints on optimization problems, which represent natural bounds on the variables, regularization, or restricted domains.

Equality constrained optimization. An equality constraint $h(x) = 0$ can be equivalently replaced by a pair of inequality constraints $h(x) \leqslant 0$ and $-h(x) \leqslant 0$. Therefore, for theoretical purposes, equality constraints are redundant; however, it can be beneficial to treat them specially in practice. There are several ways to handle the equality constraints $h_i(x) = 0$ in convex optimization: variable reparameterization, solving the dual, and analytically solving them. Equality constraints may be eliminated using a nullspace reparameterization, in which the optimization variable is replaced by one in the nullspace of the equality constraint. That is, $Ax = b$ is replaced by $A(Nz + x_0) = b$, where z is the new optimization variable, x_0 is any feasible solution to the original constraint, and N is a matrix whose range is in the nullspace of A. Equality constraints may similarly be introduced, as is useful for example in distributed optimization, e.g. ADMM (Boyd et al., 2011). There is also an important special case where linearly constrained quadratic programs are analytically solvable (as per its Karush-Kuhn-Tucker (KKT) conditions), and this forms the basis for Newton's method and interior-point methods, powerful methods for solving more complex forms of convex optimization problems.

Inequality constrained optimization. Perhaps the greatest difference between unconstrained (or equality constrained) optimization and inequality constrained optimization is that for the latter, it is not known beforehand which constraints will influence the result. Equality constraints in general influence the result. However, this is not the case for inequalities. Due to this fact, there exist only iterative algorithms to solve inequality constrained problems. The main ways to approach inequality constraints are based on identifying "active" constraints, penalizing constraints, and projecting onto constraints. Here, we briefly describe the approaches:

- *Active-set methods*: The essence of active-set methods follows from the observation that at the optimal solution to an optimization problem, inequality constraints are either satisfied with equality or not imposed (or "active") at all. The number of possible choices for the active set are very large, up to 2^m, where m is the number of inequality constraints, and therein lies the challenge of inequality constraints. Active-set methods iterate on guesses of the optimal active set, that is, the set

of constraints that are satisfied with equalities at the optimal solution. We then can solve the resulting equality constrained optimization problem with a variety of techniques as described above. Famous active-set methods include sequential quadratic programming (SQP).

- *Barrier methods*: Barrier methods avoid the combinatorial difficulty of inequality constraints in a different way. These methods generate iterates that stay away from the boundary of the feasible region defined by the inequality constraints, and hence are also known as interior-point methods. This is done through the relaxation of inequality constraints with "barrier" functions which act as soft (and differentiable) constraints. As the solution of the convex program is approached, the barrier effects are weakened to permit an increasingly accurate estimate of the solution. While interior-point methods are generally much faster on large problems, active-set methods often benefit greatly from warm-starting optimization.
- *Projection methods*: A conceptually simple approach to inequality constrained optimization is a projected descent method. Intuitively, these methods alternate between computing a descent direction and projecting the new iterate onto the feasible set of solutions. Projected gradient methods can be much faster than other methods, but projecting onto some types of constraints can be as difficult as solving the overall problem. In special cases of "simple" constraints, where efficient methods can be found for the projection step, these methods are very appealing because they do not need to consider the combinatorial set of possible active constraints, nor do they need to solve a relaxed problem. Examples of simple constraints include affine images, norm balls, non-negative orthants, and some simple polyhedra and cones.

Although presented separately above, the techniques that address different aspects of convex optimization can often by readily combined for designing an appropriate method for a new problem. Chapter 7 devises a projected gradient descent method, employing techniques for both equality and inequality constrained optimization with a convex objective.

2.2 REINFORCEMENT LEARNING

Reinforcement learning (RL) studies algorithms for optimizing performance in sequential decision making problems, where an agent continually interacts with its environment (see Figure 5).

Mathematically, we assume decisions are made at discrete time intervals. Each timestep, the agent receives an observation o_t (or state s_t) and takes an action a_t. The environment

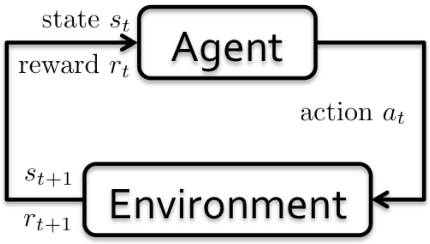


Figure 5: A simple illustration of reinforcement learning.

then receives and executes this action, and a reward signal r_t is computed according to a reward function $R(s_t, a_t)$. The environment advances the state to s_{t+1} according to a transition probability function, $P(s_{t+1}|s_t, a_t)$. Both s_{t+1}, r_t are sent back to the agent, and the loop continues. An RL algorithm seeks to maximize the agent's total reward, given a previously unknown environment, through a trial-and-error learning process. Some problems may optionally have a finite horizon H, after which the sequential process terminates. Additionally, a real-valued discount factor $\gamma \in (0, 1]$ may be given, which injects a preference of rewards received sooner rather than later. The objective is to optimize the expected sum of discounted rewards over time of the agent:

$$\eta = \mathbb{E}[\sum_{t=0}^{H} \gamma^t r_t], \tag{5}$$

Reinforcement learning is a general framework that can be used to study a wide range of problems and it has been applied in a variety of different fields, from business inventory management (Van Roy et al., 1997) to robot control (Kober et al., 2013), to structured prediction (Daumé et al., 2009). In mixed autonomy systems, the automated components can be framed as the learning agent; the human decision making components and the rest of the system can be framed as the environment. Chapter 4 will provide concrete examples in the context of mobility. For a more thorough overview of reinforcement learning and its history, we refer the reader to Sutton and Barto (1998).

This section serves as preliminary material for Part i of this thesis. Specific to this thesis, we review deep reinforcement learning, the use of function approximators in reinforcement learning, and policy gradient methods.

Deep learning. Deep learning employs powerful function approximators such as neural networks in order to learn representations from data. This stands in sharp contrast to traditional approaches in machine learning, which have typically required hand-crafted

features. Originally conceived in the 70s and 80s (Werbos, 1974; Parker, 1985; LeCun, 1985; Rumelhart et al., 1986), deep learning has grown increasingly popular in recent years, achieving state-of-the-art performance in speech recognition, image classification, machine translation, and many other applications. We refer the reader to LeCun et al. (2015) for a high-level overview of the underlying techniques and recent advances in deep learning, and to Goodfellow et al. (2016) for a more comprehensive account of the subject.

Deep reinforcement learning. Deep reinforcement learning (Deep RL) studies reinforcement learning algorithms that make use of expressive function approximators such as neural networks. This allows the algorithm to scale up to high-dimensional sensory inputs and complex control logic, without requiring manual feature engineering or limiting oneself to simple, insufficiently expressive models. Its success can be traced back to the 90s, with the work by Tesauro (1994) demonstrating a neural-network-learned strategy achieving superior performance on the backgammon game. Recently, advances in deep learning have led to significant progress in Deep RL, with impressive applications such as learning to play Atari games from raw pixels (Mnih et al., 2013; Mnih et al., 2015), mastering the game of Go (Silver et al., 2016), acquiring advanced manipulation skills (Levine et al., 2016), and learning high-dimensional locomotion controllers (Schulman et al., 2015; Schulman et al., 2016; Lillicrap et al., 2016). We refer the reader to Y. Li (2017) for a more recent survey of the algorithms and applications of deep reinforcement learning.

There are two main model-free strategies for solving (deep) reinforcement learning problems and both are amenable to the use of function approximation in different ways. We first introduce the two strategies, and then describe function approximation in their context. Model-free is meant to refer to approaches which are not based on a dynamics model. For a more detailed classification of deep reinforcement learning methods, we refer the reader to Schulman (2016).

Policy optimization. The first approach, policy optimization, is to search in the space of behaviors in order to find one that performs well in the environment. These methods are centered around the policy, the function that maps the agent's state to its next action. These methods view reinforcement learning as a numerical optimization problem where we optimize the expected reward with respect to the policy's parameters. There are two ways to optimize a policy.

- *Derivative free optimization (DFO)*: These methods work by perturbing the policy parameters in many different ways, measuring the performance, and then moving in the direction of good performance. These methods including random search, ge-

netic algorithms, genetic programming, and evolutionary algorithms. Some DFO algorithms used for policy optimization include cross-entropy method (Szita and Lörincz, 2006), covariance matrix adaptation (Wampler and Popović, 2009), and natural evolution strategies (Wierstra et al., 2008) (these three use Gaussian distributions); and HyperNEAT, which also evolves the network topology (Hausknecht et al., 2012).

- *Policy gradient methods*: These algorithms can estimate the policy improvement direction by using various quantities that were measured by the agent; unlike DFO algorithms, they do not need to perturb the parameters to measure the improvement direction. Policy gradient methods are a bit more complicated to implement, and they have some difficulty optimizing behaviors that unfold over a very long timescale, but they are capable of optimizing much larger policies than DFO algorithms. Policy gradient methods include the work of Williams (1992), Sutton et al. (2000), Jaakkola et al. (1994), and Kakade (2002).

Approximate dynamic programming. The second approach, approximate dynamic programming, is to use statistical techniques and dynamic programming methods to estimate the utility of taking actions in states of the environment. These methods focus on learning value functions, which predict how much reward the agent will receive. The true value functions obey certain consistency equations, and ADP algorithms work by trying to satisfy these equations. Policy iteration and value iteration are two well-known algorithms for exactly solving RL problems that have a finite number of states and actions.

Finally, there are actor-critic methods that combine elements from both policy optimization and approximate dynamic programming. The method described in Chapter 5, along with Lillicrap et al. (2016), Schulman et al. (2016), and Heess et al. (2015), are examples of actor-critic methods.

Function approximation in reinforcement learning. In policy optimization methods, function approximators such as deep neural networks are used to represent the *policy*. In approximate dynamic programming methods, function approximators such as deep neural networks are used to represent the *value function* (e.g. Mnih et al., 2015).

2.3 COMBINATORIAL OPTIMIZATION

In combinatorial optimization, we also wish to optimize a function $f(x)$, however this time $x \in \mathcal{S}$ resides in a set $\mathcal{S}$ of feasible solutions to the problem. For instance and in con-

trast to convex optimization, x may reside in a discrete space, e.g. $\mathcal{S} = \mathbb{N}^m$. Combinatorial optimization problems arise in countless applications, from billion-dollar operations to everyday computing tasks. They are used by airline companies to schedule and price flights; by large companies to decide what and where to stock in their warehouses; by delivery companies to decide the routes of their delivery trucks; by Netflix to decide which movies to recommend you, by a GPS navigator to come up with driving directions; and by word-processors to decide where to introduce blank spaces to justify a paragraph.

Importantly, in combinatorial optimization, one solution vector x may have no relation to a different vector x', no matter how similar they may look. This is in stark contrast to convex optimization, where conditions such as Lipschitz constants provide bounds on the deviation $\|f(x) - f(x')\|$. As such, in general, solution methods must either find a global optimum through exhaustively searching all possible solutions (or cleverly reducing the search space using deductive reasoning techniques), find a local optimum within some defined neighborhood, or find a solution that is within some bound of the global optimum. In the following, we review the computational complexity of solving combinatorial optimization problems and these main classes of combinatorial optimization methods, based on the type of solution they produce. We refer the reader to Papadimitriou and Steiglitz (1998) and Trevisan (2011) for a more thorough overview and specific examples of combinatorial optimization and combinatorial algorithms. This section serves as preliminary material for Part iii of this thesis.

Computational complexity. The complexity of an algorithm is the amount of resources required for running it, for instance time or space. The complexity of a problem is the minimum of the complexities of all possible algorithms for this problem (including unknown algorithms). Problems are characterized by complexity classes, which divide problems based on their resource requirements.

Famous named complexity classes include P, NP, NP-Complete, and NP-Hard.

- *P*: The complexity class P contains all decision problems that can be solved by a deterministic Turing machine using a polynomial amount of computation time (polynomial time).
- *NP*: The complexity class NP is the set of all decision problems for which the instances where the answer is "yes" have efficiently verifiable proofs.
- *NP-Complete*: The hardest problems in NP are called NP-complete problems, whose solutions are sufficient to solve any other NP problem in polynomial time.
- *NP-Hard*: The complexity class NP-Hard is the set of problems that are at least as hard as the hardest problems in NP. Contained in the NP-Hard class are search and optimization problems, not just decision problems.

That is P $\subseteq$ NP $\subseteq$ NP-Hard and NP-complete $\subseteq$ NP. The most important open question in complexity theory, the P versus NP ("P = NP?") problem, asks whether polynomial time algorithms actually exist for solving NP-complete, and by corollary, all NP problems. It is widely believed that this is not the case. If P $\neq$ NP, then NP-hard problems cannot be solved in polynomial time. As such, many combinatorial optimization problems are NP-hard, and so it is unlikely that we can design exact efficient algorithms for them. Chapter 9 includes a survey of complexity results in ridesharing, typically posed as search or optimization problems.

Reductions. A standard method for obtaining lower bounds of complexity consists of reducing a problem to another problem. That is, a reduction from one problem to another may be used to show that the second problem is at least as difficult as the first. Problem A reduces to problem B, written $A \leqslant_m B$[1], if an algorithm for solving problem B efficiently (if it existed) could also be used as a subroutine to solve problem A efficiently. When this is true, solving A cannot be harder than solving B. In Chapter 9, we employ a reduction to show that a ridesharing system optimization problem is NP-Hard.

Combinatorial optimization methods may be characterized based on the type of solution they produce: globally optimal, locally optimal, and boundedly suboptimal.

Exact methods. Exact methods terminate with the global optimal solution to a combinatorial optimization method. In some special cases, there exist efficient exact methods, such as for problems within the complexity class P, for example the shortest path, minimum spanning tree, and max flow problems. In general, however, these methods will undergo exhaustive enumeration or implicit evaluation of all admissible solutions, in order to obtain the optimal solution. Two important classes of methods which perform implicit evaluation are dynamic programming and branch-and-bound techniques. The main difference between them resides in the way they divide a problem into subproblems. A branch and bound algorithm partitions the problem into independent subproblems, solves the subproblems and outputs as optimal solution to the original problem the best feasible solution found along the search. Branch and bound is the main workhorse of commercial Mixed Integer Linear Programming (MILP) solvers (Koch et al., 2011). Many techniques can heuristically speed up the search progress of branch and bound techniques; for instance, cutting planes can reduce the search space. In contrast, a dynamic programming framework is applicable to a smaller set of optimization problems

1 The m in $A \leqslant_m B$ indicates a *mapping* reduction and is a common type of reduction. A mapping reduction is an algorithm that can transform (map) any instance of a decision problem A into an instance of decision problem B, *and* the solution to any problem B can be efficiently (poly-time) transformed into the solution of problem A.

and more specifically to problems that can be divided into subproblems not independent, but into subproblems that share subsubproblems. In this context, a dynamic programming approach does not repeatedly solve common subsubproblems: each of them is solved just once and its optimal solution saved in a suitable table. Chapter 9 employs exact methods, in the context of mobility systems.

Local search. One of the most intuitive solution approaches to combinatorial optimization is to start with a known feasible solution and slightly perturb it while decreasing the value of the objective function. In order to operationalize the concept of "slight perturbation," let us associate with every $x \in \mathcal{S}$ a subset $N(s) \subseteq \mathcal{S}$, called the neighborhood of x. The solutions in $N(s)$, or the neighbors of x, are viewed as perturbations of x. Local search methods then move from neighbor to neighbor as long as possible while improving the objective value. This simplest variant of local search is called hill climbing (or simple hill climbing). Local search can be seen as the basic principle underlying many classical optimization methods, such as gradient descent for continuous optimization or the simplex method for linear programming. Some of the important issues that have to be dealt with when designing a local search procedure are: how to pick the initial solution, how to define neighborhoods, and how to select a neighbor of a given solution.

A key benefit of these methods is their wide applicability and low empirical complexity; local search methods can be used for highly intricate combinatorial optimization problems, for which analytical models would involve astronomical numbers of variables and constraints, or about which little theoretical knowledge is available. A key drawback of local search methods is that the procedure stops when it encounters a local optimum. Another is that, in general, there is no guarantee that the value of the objective function at an arbitrary local optimum comes close to the globally optimal value. Metaheuristic algorithms extend local search methods to overcome some of these challenges by introducing restarts (i.e. repeated local search), memory (i.e. tabu search), or stochasticity (e.g. simulated annealing). Chapter 9 employs local search methods, in the context of mobility systems.

Approximation algorithms. Approximation algorithms are those that run in polynomial time and find solutions that are guaranteed to be close to optimal. By no longer seeking the globally optimal solution, it may become feasible to aim for a polynomial running time. At the same time, it is of interest that the worst case performance of the algorithm is close to optimal. Common techniques for designing approximation algorithms include greedy algorithms, local search, linear programming relaxations, primal-dual methods, dynamic programming, and random sampling. Famous examples of problems with approximation algorithms include vertex cover, set cover, Steiner tree, knapsack, and travel

salesman problems. Chapter 10 presents an approximation algorithm for metric k-set partitioning.

3

REVIEW OF AUTOMATED TRANSPORTATION

> Study the past if you would define the future.
>
> Confucius (479 BC)

A self-driving taxi picks you up at home and seamlessly delivers you to work. Automated vehicles deliver your lunch and your packages; they pick up your kids from school; they magically solve the problem of finding a parking spot; and they eliminate drunk driving. Transportation is as easy as clicking a hyperlink in a browser. With minimal effort on your part, available vehicles are routed to you, intelligently balancing network congestion to quickly get you where you are going.

Although these descriptions are still the future (as of the time of writing), automated vehicle technology is now gradually nearing maturity and commercial introduction. Alphabet's efforts–which involve a fleet of cars that collectively have logged hundreds of thousands of autonomous miles–have received widespread media attention and demonstrate that this technology has advanced considerably. Every major commercial automaker is engaged in research in this area and full-scale commercial introduction of truly autonomous (including driverless) vehicles are being predicted to occur within three to 20 years. Several states have passed laws to regulate the use of automated vehicles, and many more laws have been proposed.

Some parts of this chapter are written based on personal correspondence with Steven Shladover, a research engineer with UC Berkeley's Partners for Advanced Transportation Technology (PATH) program. Steven began pioneering automated transportation in the 1980s. He believes that a seamless, autonomous transportation network is still decades away. However, recent advances in engineering and computer science have demonstrated that automated driving can dramatically improve the safety, efficiency, and availability of transportation, bringing this futuristic scenario a little closer to reality. This chapter

provides an overview of just how far automated transportation has come in recent years, and where it may go in our lifetimes.

3.1 SAFETY FIRST

The stakes for automated vehicles are high. In the United States alone, more than 30,000 people are killed each year in crashes, approximately 2.5 million are injured, and the vast majority of these crashes are the result of human error (Choi et al., 2008). All told, 5.4 million automotive crashes happen each year in the United States, and these car crashes result in an estimated $871 billion in economic, social, and emotional damage (see Figure 6). By greatly reducing the opportunity for human error, AV technologies have the potential to greatly reduce the number of crashes. Automobiles have become increasingly heavy over the past 20 years partly to meet more rigorous crash test standards. If crashes become exceedingly rare events, it may be possible to dramatically lighten automobiles.

Automated vehicles may also reduce congestion and its associated costs. Estimates suggest that effective road capacity (vehicles per lane per hour) can be doubled or tripled. The costs of congestion can also be greatly reduced if vehicle operators can productively conduct other work. Automated vehicle technology may also reduce energy use.

In the long run, automated vehicles may also improve land use. Quite apart from the environmental toll of fuel generation and consumption, the existing automobile shapes much of our built environment. Its centrality to our lives accounts for the acres of parking in even our most densely occupied cities. With the ability to drive and park themselves at some distance from their users, automated vehicles may obviate the need for nearby parking for commercial, residential, or work establishments, which may enable a reshaping of the urban environment and permit new in-fill development as adjacent parking lots are made unnecessary.

Simply replacing human drivers with machines does not guarantee safety–in fact, nothing will ever guarantee absolute safety in a moving vehicle. From children playing in the street, to birds, snowstorms, and even inattentive or aggressive human drivers, the hurdles to making driving safer can seem insurmountable. However, autonomous vehicles can limit the risks of driving. To work toward this goal, researchers and engineers are trying to improve all the parts involved: on-board sensors, vehicle-to-vehicle and vehicle-to-road communications, and underlying software (Korkmaz et al., 2004; F. Li and Y. Wang, 2007; Gozalvez et al., 2012).

Nearly 20 years ago, researchers from UC Berkeley's PATH showed that autonomous vehicles could operate safely on a freeway. In partnership with General Motors (GM),

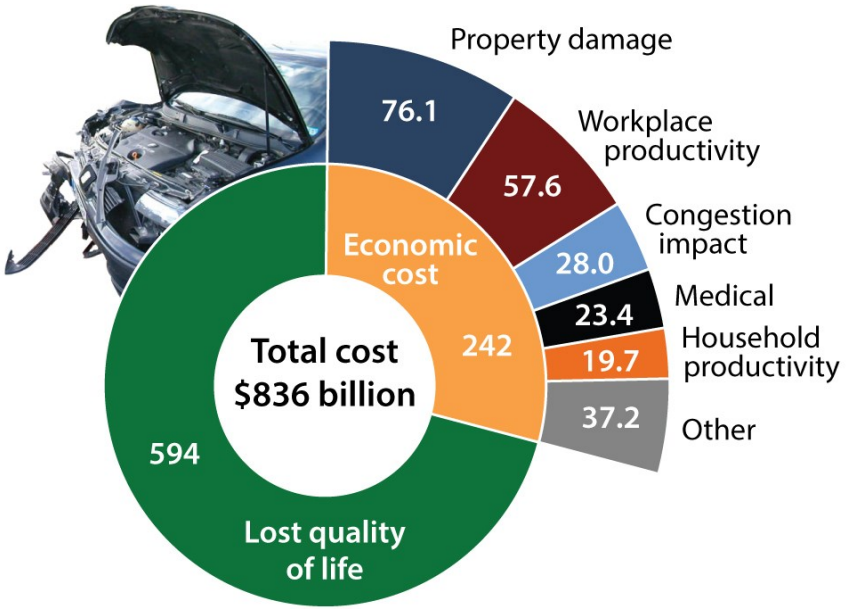


Figure 6: The annual cost of motor vehicle crashes in the United States in billions of dollars, according to a study from 2010 by the US Department of Transportation. *Credit: Florian Brown-Altvater*.

they tested an autonomous platoon of vehicles on a freeway in San Diego in 1997 (Shladover, 2005; Shladover, 2007). Magnets were embedded a few feet underneath the road surface and guided specialized vehicles along the center of each lane with high precision. Due to the nature of magnetic fields, this system was largely immune to weather conditions and sensor malfunction. Eight driverless Buick LeSabres drove single-file with just seven feet between each vehicle. They changed lanes and shifted positions, all without any human input. The event enjoyed national news coverage, and the 1,000 visitors who rode in the vehicles were ecstatic.

Despite the success of PATH's early trials with its autonomous vehicles, the US Department of Transportation decided to cut the funding for the project, citing budget pressures and the greater importance of near-term safety systems. Much of the work was left unfinished and unpublished. For a comprehensive view of automated highway systems, we refer the reader to P. Ioannou (2013).

Nevertheless, this scheme for automated driving–using road-embedded magnets to guide vehicles–remains the preferred strategy for helping autonomous vehicles "see" the road. On a snowy day, most human drivers struggle to control their vehicles on the road, due to low tire traction and reduced visibility. Soon after PATH's first successful demonstration in 1997, PATH researchers turned the problem of snow driving on its

head by developing a magnet-based automated guidance system for safe snow removal (K. S. Yen et al., 1999; Ravani and K. S. Yen, 2000). The system allowed human snow plow operators to safely navigate snow-obscured roads. Dubbed the Advanced Snowplow Project, the automated snowplows were successfully tested at several locations, including the notorious Donner Pass in the Sierra Nevada.

This precise magnetic guidance system (Lasky et al., 2004) has also made it easier for wheelchair users to board public transportation vehicles, by guiding large vehicles to within two centimeters of the curb. In 2013, PATH ran an autonomous public bus line in Eugene, Oregon as part of a six-month study on improving bus accessibility for the disabled (W.-B. Zhang et al., 2007). This embedded sensor approach also benefits from robustness to changes in temperature, weather, and other conditions and has little maintenance requirements, and thus has potential for commercialization for wider adoption.

In addition to magnetically guided systems, which depend on the modification of roads, researchers have developed alternative schemes for automating driving in novel environments. In 2005, the US Department of Defense's Defense Advanced Research Projects Agency (DARPA) issued a "Grand Challenge" on autonomous driving in desert areas, with the motivation of aiding in combat scenarios. This challenge spurred university researchers across the United States to begin new projects with the goal of designing autonomous vehicles that could navigate the desert environment (Thrun et al., 2006). In 2007, DARPA issued a follow-up challenge to develop autonomous vehicles for urban driving (Buehler et al., 2009). Five vehicles completed the desert track and six completed the urban track, expanding the applicability of automated transportation beyond highways. Several members from the winning teams of both contests were soon recruited to work on Google's self-driving car team.

Unlike UC Berkeley's PATH vehicles, the vehicles borne from the DARPA challenges relied on expensive sensors, computer vision algorithms, and planning algorithms from the field of robotics to "see" the road. Relying on computer vision alone complicates road safety in an autonomous vehicle, but Google's efforts to develop a self-driving car have proven the feasibility of automating driving without specially-designed roads. At the time of writing, Google's self-driving cars have completed over one million miles of testing, resulting in only one accident for which Google was at fault.

Following research continues to develop new methods for improving safety in automated transportation. One such approach employs reachability analysis, which uses detailed physical models of vehicles and their environments to evaluate unsafe conditions that could be encountered when moving along a particular route (Mitchell et al., 2005; Bayen et al., 2007; Gillula et al., 2011). These models explicitly account for uncer-

tainty and worst-case scenarios, providing bounds for what an autonomous vehicle can safely do.

NASA is now applying these reachability analyses to study the safety of drones flying in proximity to one another in vast airspaces. One goal of this research is to help automate platoons of many drones, which are in part inspired by the ground vehicle platoons of PATH's 1997 highway experiments. Their research shows how treating groups of drones as single units can dramatically improve the safety and efficiency of automated flight.

Airspace and public roads have some key differences, however. "Automating driving a vehicle on public roads is orders of magnitude more complicated than an autopilot system for commercial aircraft flying at 30,000 feet," says Shladover. "If something goes wrong at 30,000 feet, you've got on the order of 10 seconds to identify what's going on and try to correct it; if you're in a road vehicle, you have about one-tenth of a second." Still, efforts like these are bringing us closer to making safer roads a reality.

3.2 FREEING THE FREEWAY

While road safety deserves careful consideration in the design of any autonomous car, bumper-to-bumper traffic affects drivers frequently–and might be alleviated by driverless cars of the future. "Phantom" traffic jams, which appear out of nowhere, are perplexing to both drivers and researchers. There are no accidents, traffic bottlenecks, work zones, or bad weather to explain why a road is backed up with traffic. This type of traffic jam accounts for about a quarter of all traffic jams, and results in inefficient stop-and-go driving. Solving phantom jams and reducing stop-and-go driving with the help of autonomous vehicles would do more than save drivers time–the braking and acceleration inherent to driving in traffic also wastes gas. So why do these jams occur in the first place?

Car-following models were first developed by researchers at GM in the 1950s and explain patterns of stop-and-go traffic by relating one car's speed to the speed of the car in front of it (Brackstone and McDonald, 1999). As one car slows down, the following car is forced to brake after a brief delay due to the driver's reaction time. Then the next following car brakes, and the reaction times of each driver begin to add up. This compounded delay, caused by human reaction times and other response patterns, causes backwards propagating waves to travel down the freeway, which we experience as stop-and-go traffic. As demonstrated beautifully in the field experiment of Sugiyama et al. (2008), from a constant reference frame, for instance watching the stop-and-go traffic from a high-

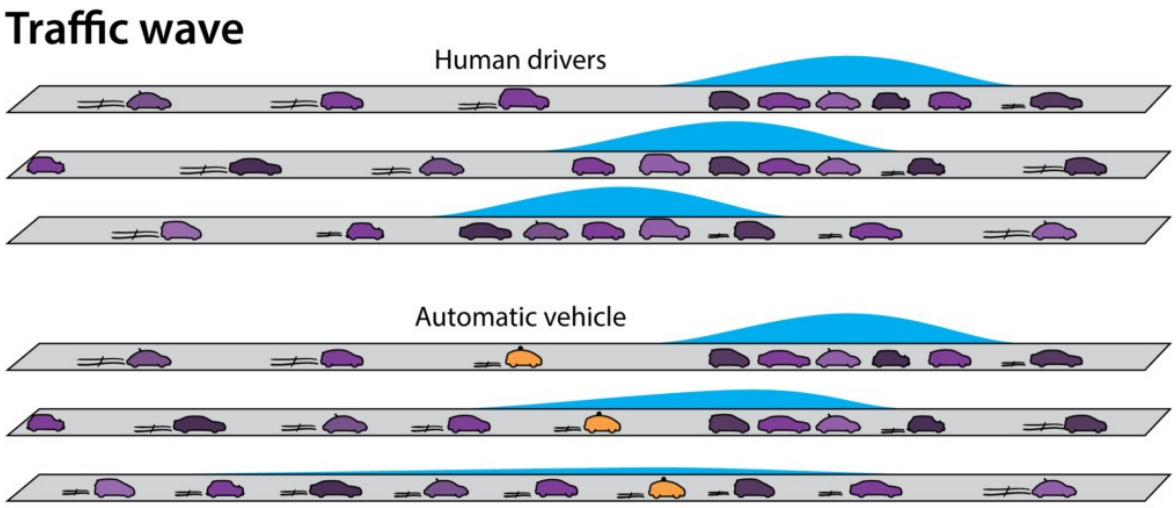


Figure 7: **Top**: In freeway traffic, human drivers typically accelerate to catch up with the car preceding their own. This diagram shows how human reaction times can cause a chain reaction of drivers braking, one after another, even after the first car begins to accelerate again. The group of slowed or stopped vehicles seems to travel backwards, even though all the vehicles are moving forward. This phenomenon is known as the backwards wave.
Bottom: An automated car can accelerate just enough to keep a safe distance between vehicles both in front of and behind itself, dampening the backwards wave. Any following vehicles, driven by humans or not, will react accordingly. Instead of stop-and-go, all cars can drive on smoothly. *Credit: Florian Brown-Altvater*.

way overpass, one would see a wave of vehicles moving backwards even though all the cars are moving forwards. In physics, this concept is known as the backwards wave (see Figure 7).

One of the primary motivations of the automated highway project of 1997 was to prove that autonomous vehicles can operate with minimal reaction times, and can be designed to drive without the stop-and-go phenomenon. When vehicles follow one another so closely, air drag is reduced and fuel economy improves by 20%. This finding has particular implications for America's vast trucking routes and net energy consumption. In the future, freeway traffic may be optimized through the automated coordination of platoons of passenger and commercial vehicles, cutting transit times, saving gas, and reducing a major source of frustration for drivers.

3.3 CONFRONTING THE LAST MILE

If an automated taxi dropped passengers off at the freeway exit nearest to their destination–rather than at their doorstep–most people would still have miles left to travel. This is particularly true in the United States, where many people still live in sprawling suburbs. This is called the last mile problem, referring to the technical, economic, and political challenges of transporting people efficiently across short distances.

While the volume of vehicles passing through a given freeway exit is likely to be high, the volume of vehicles driving on a particular neighborhood street tends to be low. Such sparse traffic in residential areas has its benefits in less noise pollution and better air quality. However, from an economic standpoint, this means that it is much cheaper for transit systems to operate exclusively along freeways and freeway exits, avoiding individual homes.

In dense urban centers like downtown San Francisco, the economics of the last mile problem are less daunting. However, coordinating transportation needs in such places presents its own challenges, requiring excellent estimation of timing and demand. When and to where should available vehicles be sent? Can we measure or even predict where people are coming and going?

Answering these questions relies on origin-destination estimation, which estimates the demand for travel between pairs of locations in a network. Origin-destination estimation has been actively studied by the transportation research community since the 1940s, but a paucity of real-world data has long stymied its successful application to the last mile problem. For decades, the best available data was based on census surveys, which directly query the population for its transportation needs. Since the 1970s, advances in sensing infrastructure, including traffic counters embedded underneath roads, have bolstered origin-destination estimates. These estimates have played an integral role in long-term urban planning and road development in highly-trafficked areas, but relatively few roads in a given city are actually equipped to monitor their traffic, complicating efforts to accurately predict exactly where and when bottlenecks might occur.

In 2015, we studied an alternative approach for estimating short-term transportation needs, further elaborated in Chapter 7. In particular, we collaborated with AT&T to develop a new optimization method, called Cellpath, which uses high-resolution data from cell phones to predict transportation demands in the greater Los Angeles area. This work showed that the demand for travel on particular routes could be estimated with 90% accuracy using location traces from cell phones. We found that the aggregated phone traces, demand for travel between an origin and destination, and demand for

a particular road could be combined to efficiently predict transportation demands with unrivaled precision. Armed with comprehensive demand predictions, autonomous fleets have enormous potential to efficiently coordinate a minimum number of deployed vehicles on the street. A 2014 study, conducted by MIT and Stanford in Singapore, estimated that an autonomous fleet of taxis could provide the same level of transportation service as today's taxi systems using just one-third the current number of vehicles (Spieser et al., 2014).

3.4 TAKING TO THE STREETS

"Today we are well underway to a solution of the traffic problem." This claim, made in 1948 by Robert Moses, the master builder of the New York City area, is as true today as it was then. Which is to say, not at all. In the middle of the last century, the preferred solution to "the traffic problem" was more cement: new highways, bridges, and lanes. Today, the sensible solution includes more sensors and better computers: highly automated vehicles that use existing roadways and roadway networks much more efficiently. This automation, we are told, will make vehicular congestion a "thing of the past." As in the past, however, this prediction presumes that more capacity necessarily means less congestion. Today's transportation planners recognize that the relationship between these two concepts is much more complex.

Though automated transportation has come a long way in the last few decades of research and development, the benefits of automated driving are still out of reach for most people. Gains in efficiency promised by today's studies of autonomous vehicles and platoons are often computed in ideal settings. In these studies, autonomous vehicles are either rare enough not to alter preexisting traffic patterns, or numerous enough to coordinate nearly all vehicles on a given road.

However, adoption of autonomous vehicles will likely be gradual (as discussed in Chapter 4), making it difficult to model future interactions between human and computer-driven vehicles. Perhaps it should be the responsibility of policymakers to ensure a smooth transition–one that is projected to take at least 20 years. The rapid rate of technological change makes it increasingly difficult to anticipate even short-term future transportation needs (Levinson and Krizek, 2015). For instance, it is not yet known how companies will financially manage their future autonomous fleets. Will these vehicles be privately owned or rented? Will passengers hail the vehicles like taxis, or book particular routes on shared vehicles in advance? These distinct possibilities for organizing our autonomous vehicle systems may create a whole new set of challenges, both technical

and political, for managing our roads.

By reducing the time cost of driving, automated vehicles may encourage greater travel and increase total vehicle miles traveled (VMT), which could lead to more congestion. A 2016 study estimated that fuel consumption could decrease as much as 40% or increase as much as 100% once autonomous fleets of vehicles are rolled out onto the streets (see Figure 2). Automated vehicles may increase sprawl if commuters move ever farther away from workplaces. Similarly, they may eventually shift users' preferences toward larger vehicles to permit other activities, such as errands and leisure. In theory, this could even include beds, showers, kitchens, shops, or offices. If automated vehicle software becomes standardized, a single flaw might lead to many accidents. Internet-connected systems might be hacked by the malicious. And perhaps the biggest risks are simply unknowable. From seat belts, to air bags, to anti-lock brakes, automakers have often been reluctant to incorporate expensive new technology, even if it can save many lives (Mashaw and Harfst, 1990). Navigating the AV landscape makes implementation of these earlier safety improvements appear simple by comparison. For now, the prospect of autonomous vehicles introduces further uncertainty into our predictions about transportation in the future, and negotiating the risks to reach the opportunities will require careful engineering and policymaking.

Different population densities also complicate the adoption and operation of autonomous fleets. Even in dense urban areas, autonomous vehicles will not become common for at least the next decade. In contrast to the densely populated city of Singapore, where extensive automated fleet modeling has been carried out, half of the American population resides in suburban and rural areas. In such sparsely-populated regions, car ownership is high, and the economics of autonomous fleets are unsustainable. This has already been observed with human-driven fleets: ride-hailing services such as Uber and Lyft are expanding in urban centers worldwide, but efforts to expand into suburbs have been limited.

Lastly, travel time and fuel costs are but two of the simplest factors that influence the adoption of new transportation technologies. Each person has his or her own unique transportation needs and preferences. Families with young children and people with disabilities need reliable and flexible transportation options that may not resemble mainstream commutes. More broadly, transportation must meet the diverse inclinations of a large population, from safety to personal hygiene and smoking preferences. Some people might prefer reliable, shared forms of transit versus fast, private forms of transit, and might even have contradictory desires for transit that is both safe and fast, or both cheap and available. This motivates the design of algorithms for coordinating vehicles amidst

these complex human needs, which will be discussed in Part iii of the thesis.

The ideal future–where transportation is safe, seamless, affordable, and available to all–is a grand and beautiful picture. However, there are still many hurdles to cross when it comes to making autonomous transportation generally available.

3.5 FULL SPEED AHEAD

In the early 1900s, there were no driver's licenses, stop signs, traffic lights, lane lines, street lighting, brake lights, or posted speed limits. In 1917, 75% of traffic fatalities were pedestrians; now that number is less than 15%. We have certainly made a lot of progress since then. Autonomous vehicles have the potential to solve some long-standing problems in traffic, but also threaten to introduce new ones. Perhaps in a couple decades, other technological advances will eliminate a majority of transportation needs altogether, thus providing an alternative to optimizing traffic.

As of the time of writing, there have been four self-driving car fatalities (three caused by Tesla "Autopilot" vehicles in 2016-2018, and one caused by an Uber self-driving prototype in 2018). In recent years, the world has been bracing itself for these first major accidents caused by autonomous vehicles, with countless discussions of the trolley problem as well as active work in public policy. In the California State Legislature of 2012, state senators passed a bill directing the California Department of Motor Vehicles (DMV) to set testing and operating guidelines for autonomous vehicles, an unprecedented responsibility. The California DMV was the first transit agency in the world to tackle the implications of self-driving vehicles, and called upon the Institute of Transportation Studies at UC Berkeley to help the agency navigate this deeply technical problem. In the following months, Shladover and his colleagues advised the DMV on the technologies involved with autonomous vehicles and made policy recommendations to ensure the safety of autonomous vehicles, while also providing companies with the room to innovate. Unfortunately, as evidenced by the recent fatalities, there is much more that is needed to ensure safe testing and deployment practices.

Automating transportation is coming, slowly but surely. Automated vehicles can dramatically cut down on traffic accidents and fuel consumption on highways, complete the last mile of each journey, and bring low-cost transportation to many people. Someday, the average commute may even resemble boarding an escalator (see Figure 8). But a world in which we can leave transportation to the machines lies ahead of us, and the challenges span from the technical and political to the human and the personal. It may not be time for us to buckle our seat belts inside driverless cars, but our shared dream

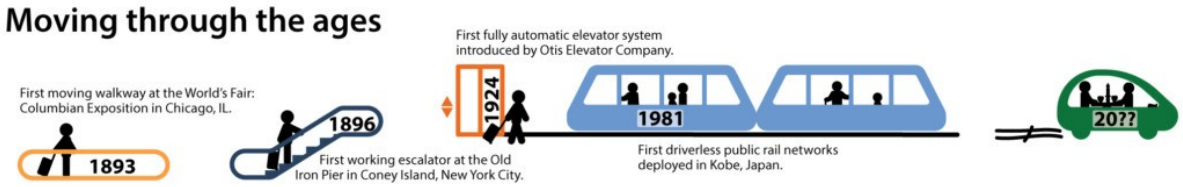


Figure 8: Credit: Florian Brown-Altvater.

of an automated transportation system is slowly but surely becoming our future.

Part I

CONTROL

4

EMERGENT BEHAVIORS IN MIXED AUTONOMY TRAFFIC

A sip of wine, a cigarette,
And then it's time to go.
I tidied up the kitchenette;
I tuned the old banjo.
I'm wanted at the traffic-jam.
They're saving me a seat.

Leonard Cohen, Boogie Street (2001)

Part i of the thesis is dedicated to studying the control of mixed autonomy systems. In particular, we demonstrate that machine learning methods, in particular deep reinforcement learning, are suitable for this study, surpassing classical methods based in partial differential equations and manual controller design in handling the system complexity. We begin by first showing that a mixture of automated and human-driven vehicles may yield vastly different system characteristics, such as average velocity, as compared to a system with only human-driven vehicles (Chapter 4), thus demonstrating the existence of mixed autonomy systems in the open world. In particular, by casting the problem into the framework of model-agnostic reinforcement learning, it is established that a small fraction of automated vehicles has the potential to eliminate congestion and dramatically improve overall road velocities for all vehicles.

Many advances are needed to enable mixed autonomy systems at the scale of thousands or even millions of agents, including high speed and distributed simulation systems and algorithmic development. To this end, we present generic deep reinforcement learning techniques for scaling up to higher dimensional control problems, such as controlling many vehicles in a mobility system (Chapter 5). Additionally, an open-source library is introduced which allows for integrated studies of reinforcement learning and traffic microsimulation, with scalable distributed simulation and cloud deploy-

ment (Chapter 6). The contributions of this part have implications for the environment and public policy concerning the regulation of automated vehicles, as well as scalable reinforcement learning.

The present chapter formulates and approaches the mixed autonomy traffic control problem (where both automated and human-driven vehicles are present) using the powerful framework of deep *reinforcement learning* (RL). Traffic dynamics are often modeled by complex dynamical systems for which classical analysis tools can struggle to provide tractable policies used by transportation agencies and planners. In light of the introduction of automated vehicles into transportation systems, there is a new need for understanding the impacts of automation on transportation networks.

The resulting policies and emergent behaviors in mixed autonomy traffic settings provide insight for the potential for automation of traffic through mixed fleets of automated and manned vehicles. Model-agnostic learning methods are shown to naturally select policies and behaviors previously designed by model-driven approaches to improve roadway efficiency, such as stabilization, compressive platooning, traffic breaks, and cooperative merging. We additionally demonstrate that state-of-the-art hand-designed controllers for stabilizing traffic excel when in-distribution, but fail to generalize; on the other hand, we show that even simple neural network policies can solve the stabilization task across density settings, generalize to out-of-distribution settings, and yield or even exceed a theoretical velocity upper bound on the non-controlled system. Remarkably, in an intersection network configuration, the learned policy succeeds at maximizing velocity by leveraging the human driving behavior to form an emergent mixed autonomy platoon, which efficiently spaces the vehicles in the network. We describe our results in the context of existing control theoretic results for stability analysis and mixed autonomy analysis. This chapter additionally introduces state equivalence classes to improve the sample complexity for the learning methods. Mixed autonomy traffic, then, is a mixed autonomy system in the open world, and serves as a running example throughout the thesis for concretely demonstrating the problems and challenges of mixed autonomy systems.

4.1 OVERVIEW

Emergent behaviors have long motivated general learning methods such as genetic algorithms, simulated annealing, and RL algorithms, producing interesting, useful and captivating behaviors in complex dynamical systems such as swarms (Reynolds, 1987), ant colonies (Maniezzo, 1992), and life (Gardner, 1970). The present chapter studies emer-

gent behaviors in road transportation networks in the presence of mixed autonomy, the mixture of automated and non-automated vehicles, through the use of reinforcement learning techniques. Such studies can eventually help to design controllers for deployment in real-world settings, and can aid in complex tasks such as traffic management, reducing energy consumption, environmental monitoring, etc.

It has been found that greenhouse gas emissions could be reduced by up to 20% through traffic congestion mitigation strategies (Barth and Boriboonsomsin, 2008) and numerous field operational tests have demonstrated that a small number of vehicles injected into a congested traffic system can result in a 13-40% reduction in fuel consumption (CIECA, 2007; Barth and Boriboonsomsin, 2009; Stern et al., 2017). These studies motivate the design of controllers for a small number of vehicles in complex traffic scenarios to achieve significant outcomes. However, modeling and analysis of traffic dynamics is notoriously complex and yet is a prerequisite for model-based traffic control (Treiber and Kesting, 2013; Papageorgiou et al., 2003). Researchers classically trade away the complexity of the model (and thus the realism of the model) for tractability of analysis (for example through aggregate models), often with the goal of designing optimal controllers with desirable provable properties, such as safety or optimality (Technical Committee ISO/TC 204, Intelligent transport systems, 2010). Consequently, results in traffic control can largely be clustered into several groups which include simulation-based numerical analysis or theoretical analysis on idealized settings. In the present chapter, we largely focus our discussion on control of microscopic longitudinal dynamics[1] and lateral dynamics (Zheng, 2014) on a variety of network configurations.

Deep RL, as presented in Section 2.2, is a powerful tool for control and has already demonstrated success in complex but data-rich problem settings such as Atari games (Mnih et al., 2013), 3D locomotion and manipulation (Schulman et al., 2015; Heess et al., 2015), and chess (Lai, 2015), among others. In this chapter, we revisit the problem of traffic control and view automated vehicles as a mechanism for congestion control, using the framework of deep RL.

Using model-free RL methods, the present chapter studies emergent behaviors in mixed autonomy traffic. This study sets the stage for further study of increasingly complex and realistic scenarios and the discovery of policies that can be deployed in real life. Real-world phenomena have highly stochastic driving dynamics with different human drivers exhibiting differing levels of aggression or timidity, drivers merging and exiting, accidents blocking a road, drivers distracted by nearby accidents, sudden slowdowns in

1 Microscopic longitudinal dynamics are sometimes referred to as car following models (CFMs) (Brackstone and McDonald, 1999).

the presence of cops, different driving styles for different weather conditions, etc. All of these affects the types of policies that might be deployed to mitigate congestion, and the complexity and diversity of these policies make automatic discovery critical. This chapter studies how model-free RL methods can autonomously discover interesting policies that exploit the dynamics of the uncontrolled drivers. It sets the seed for further research in more complex settings, one that will be increasingly important as companies start deploying autonomous vehicles commercially, alongside several other core research problems in the context of autonomous vehicles, such as localization, path planning, collision avoidance, and perception. The discovery of policies in the presence of high-level goals and complex dynamics is another crucial piece of the overall research which will enable safe and efficient next generation mobility systems.

The contributions of this chapter include:

- The formulation of the mixed autonomy traffic problem, in which automated and human-driven vehicles co-exist in the same system, in the framework of deep RL.
- The presentation of the first demonstration of model-free RL for the longitudinal and lateral control of a fleet of automated vehicles in a variety of complex mixed autonomy environments, including single- and multi-lane ring roads, a figure-eight network, and environments with frequent perturbations.
- The introduction of the concept of state equivalence classes, which improves sample efficiency of the learning method.
- Emergent behaviors, demonstrated numerically in a variety of network configurations, including stabilizing traffic, tailgating, platooning, and efficient vehicle spacing. The chapter demonstrates the selection of policies previously discovered by model-driven approaches, such as stabilization, platooning, and efficient vehicle spacing at intersections, using a model-free learning paradigm.
- For the single-lane ring road, the presentation of a theoretical upper bound on the average velocity of a mixed autonomy setting, and experiments which demonstrate that the learned policy exceeds the optimal human performance and is close to optimal mixed autonomy performance.
- The demonstration of an effective leveraging of the structure of the human driving behavior, which allows the learned policies to surpass the performance of state-of-the-art controllers designed for automated vehicles, for ring road and figure-eight settings.

Videos and additional results of the chapter are available at `https://bit.ly/2tm8lEV`.

Notation. This chapter assumes a discrete-time Markov decision process (MDP), defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma, H)$, in which $\mathcal{S} \subseteq \mathbb{R}^n$ is an n-dimensional state space, $\mathcal{A} \subseteq \mathbb{R}^m$ an m-dimensional action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ a transition probability function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ a bounded reward function, $\rho_0 : \mathcal{S} \to \mathbb{R}_+$ an initial state distribution, $\gamma \in (0, 1]$ a discount factor, and H a time horizon. The presented models are based on the optimization of a stochastic policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$ parameterized by θ. Let $\eta(\pi_\theta)$ denote its expected return: $\eta(\pi_\theta) = \mathbb{E}_\tau[\sum_{t=0} \gamma^t r(s_t, a_t)]$, where $\tau = (s_0, a_0, \ldots)$ denotes the entire trajectory, $s_0 \sim \rho_0(s_0)$, $a_t \sim \pi_\theta(a_t|s_t)$, and $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$ for all t. Our goal is to find the optimal policy $\theta^* := \text{argmax}_\theta \, \eta(\pi_\theta)$ (Sutton et al., 2000).

Car following models. Car following models (CFMs) describe the longitudinal dynamics of human-driven vehicles. We now describe standard CFMs and the concept of equilibrium velocity. In the next section, we detail a commonly used CFM called the intelligent driver model, which we employ for the longitudinal dynamics of the human drivers, in the numerical experiments of this chapter.

Standard CFMs are of the form:

$$a_i = \dot{v}_i = f(h_i, \dot{h}_i, v_i), \tag{6}$$

where the acceleration a_i of car i is some typically nonlinear function of $h_i, \dot{h}_i, v_i$, which are the headway, relative velocity, and velocity for vehicle i, respectively. Though a general model may include time delays from the input signals $h_i, \dot{h}_i, v_i$ to the resulting output acceleration a_i, we will consider a non-delayed system, where all signals are measured at the same time instant t. Example CFMs include the Intelligent Driver Model (IDM) (Treiber et al., 2000) and the Optimal Velocity Model (OVM) (Bando et al., 1994; Bando et al., 1995).

In a uniform flow equilibrium in homogeneous settings, each car moves at a constant velocity v^* with constant headway h^*. At this uniform flow equilibrium, we have

$$a_i = 0 = f(h^*, 0, v^*), \tag{7}$$

defining the relationship between the two equilibrium quantities h^*, v^*.

It is intuitive to think of the equilibrium density (which is related to the equilibrium headway h^*) as a traffic condition. Each traffic condition has associated with it an optimal (equilibrium) velocity v^*. In practice, the equilibrium density can be determined or estimated by the local traffic density. In settings with heterogeneous vehicle types,

the equilibrium can be numerically solved by constraining the total headways to be the total road length and the velocities to be uniform. It is important to note the difference between the equilibrium velocity v^* and the target velocity v_0 (free flow speed) of the vehicle models; v_0 can be thought of as a speed limit for highway traffic (Treiber et al., 2000). On the other hand, v^* is a control theoretic quantity jointly determined by the traffic condition h^*, the target velocity v_0, the system dynamics, and various other parameters.

Intelligent Driver Model. The Intelligent Driver Model (IDM) is a microscopic car following model capable of accurately modeling realistic driver behavior (Treiber et al., 2000). Using this model, the acceleration for a vehicle following IDM dynamics is defined by its bumper-to-bumper headway s (distance to preceding vehicle), velocity v, and relative velocity Δv, via the following equation:

$$a_{\text{IDM}} = \frac{dv}{dt} = a\left[1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s}\right)^2\right] \tag{8}$$

where s^* is the desired headway of the vehicle, denoted by:

$$s^*(v, \Delta v) = s_0 + \max\left(0, vT + \frac{v\Delta v}{2\sqrt{ab}}\right) \tag{9}$$

where $s_0, v_0, T, \delta, a, b$ are given parameters.

Table 1 describes the parameters of the model and provides typically used values (cite here):

Table 1: Model parameters of the Intelligent Driver Model

Symbol	**Definition**	**Typical values** (Treiber et al., 2000)
v_0	Desired speed	54 km/h
T	Time gap	1.0 s
s_0	Minimum gap	2 m
δ	Acceleration exponent	4
a	Maximum acceleration	1.0 m/s^2
b	Comfortable deceleration	1.5 m/s^2

Consider a string of vehicles on a single-lane road. This relatively simple system can be modeled as a dynamical system consisting of a cascade of n (possibly different) nonlinear systems, one for each vehicle. Although conceptually clean, the complexity of such an overall system already largely constrains formal analysis of such systems to homogeneous settings, where each of the n systems are identical, or where the system components are assumed to be linear, non-delayed, etc. With the introduction of lane changes (in any multi-lane setting), the dynamics additionally exhibit discrete events, when lane changes occur. Modeling and studying lane changes and their effects on traffic is a difficult and active area of research (Oh and Yeo, 2015; W.-L. Jin, 2010). Additional components such as traffic lights and intersections, turns, route choice, on- and off-ramps, demand, specialized lanes, and heterogeneous vehicle types, introduce further complexities in the overall dynamics of traffic control problems.

Due to discontinuous, non-smooth, and highly complex dynamics inherent to traffic control problems, we choose to study the control problem using model-agnostic RL methods, as opposed to model-based RL methods or classical control techniques. This chapter assumes a discrete-time Markov decision process (MDP), See Section 2.2 for a review of reinforcement learning and a summary of the notation.

Problem statement. In this chapter, we study the problem of *mixed autonomy traffic*: How can a set of automated vehicles optimize a traffic system in the presence of both automated and human-driven vehicles? The present chapter studies this problem for the objective of maximizing system-level velocity, for a variety of traffic settings. We note that of particular interest in this problem is the study of system-level objectives rather than local (vehicle-level or platoon-level) objectives. Other system-level objectives may be studied as well, such as energy consumption, comfort, throughput, delays, air quality, or a combination thereof.

Problem setting. We formulate the mixed autonomy traffic problem as a fully observed cooperative RL problem. We study a centralized training regime and centralized execution; other learning settings such as shared policies, multi-headed policies, and decentralized training and execution are beyond the scope of this work.

System dynamics and assumptions. The transitions of the MDP are dictated by the system dynamics, which include longitudinal dynamics for individual vehicles, lateral dynamics for individual vehicles, right-of-way dynamics at intersections, and fail-safe dynamics for individual vehicles.

The basic longitudinal dynamics of human-driven vehicles follow IDM as described in Equation 8, calibrated to accurately model realistic driver behavior on freeways (Treiber et al., 2000). In order to incorporate stochasticity into the dynamics of human-driven vehicles, the accelerations are additionally perturbed by Gaussian acceleration noise of $\mathcal{N}(0, 0.2)$, calibrated to match measures of stochasticity presented in (Treiber and Kesting, 2017). All other system dynamics are represented within SUMO, a state-of-the-art and open-source traffic microsimulator (Behrisch et al., 2011), including calibrated lateral (lane-changing) dynamics for human-driven vehicles, right-of-way, and a fail-safe (optionally) imposed on automated vehicles to ensure safety.

State representation. The state representation provides full information of the system of vehicles in the network, and takes advantage of state equivalence classes, explained in Section 4.4. For a network of vehicles in a single lane setting, the full state consists of a vector of velocities v and absolute positions x for each vehicle in the network, ordered by the absolute position of each vehicle. Note that the absolute position is defined relative to a pre-specified starting point for the network. For a network of vehicles in a multi-lane setting, a vector of lane numbers l for each vehicle is also added to this representation. Let $\text{sorted}(x)$ denote the sequence of indices by ascending order of the values in vector x. Then, the overall state representation is $s := (v_i, x_i, l_i)_{i \in \text{sorted}(x)} \in \mathbb{R}^{3k}$. Partially observable settings are out of the scope of this chapter, as here we study possible learned behaviors rather than focusing on the design of practical controllers. See Chapter 6 for an exposition of a partially observed mixed autonomy study towards practical controller design.

Action representation. The action representation permits automated vehicles to perform lateral and longitudinal actions in the traffic network. In a single lane setting, the action space simply consists of a vector of requested accelerations $c \in [c_{\text{min}}, c_{\text{max}}]^k$, where k is the number of automated vehicles, bounded between certain minimum and maximum acceptable accelerations. For the purpose of this study, automated vehicles are allowed to perform accelerations within the range $[-6, 3]$ m/s^2. If the scenario contains more than one lane, a vector of lane changing directions $d \in [-1, 1]^k$ is also provided. The lane of the vehicle is then updated as follows: $l_{t+1} = l_t + \text{round}(d)$, thus encoding actions $\{\text{left}, \text{stay}, \text{right}\}$. The lane change updates are restricted to 1) existence of the lane, and 2) a lane change cooldown duration, which prevents lane changes in quick succession. The actions $a = (c, d) \in \mathbb{R}^{2k}$ are applied to agents (automated vehicles) in order of absolute position.

Reward function. We choose a reward function to encourage high system-level velocity.

This function measures the deviation of all vehicle velocities from a user-specified desired velocity. Moreover, in order to ensure that the reward function naturally punishes the early termination of rollouts due to collisions or other failures, the function must have a non-negative range $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{\geqslant 0}$. This is done by subtracting the deviation of the system from the desired velocity from the peak allowable deviation from the desired velocity. Additionally, since the velocity of vehicles are unbounded above, the reward is bounded below by zero, to ensure nonnegativity. Define v_{des} as the desired velocity, $\mathbf{1}^n$ a vector of ones of length n, n as the number of vehicles in the system, and v as a vector of velocities. The reward function (with abuse of notation for clarity) is given as

$$r(v) = \max\{0, \|v_{\text{des}} \cdot \mathbf{1}^n\|_2 - \|v - v_{\text{des}} \cdot \mathbf{1}^n\|_2\} \tag{10}$$

4.4 STATE EQUIVALENCE CLASSES

We define a *state equivalence class* to be subset of states $\mathcal{T} \subseteq \mathcal{S}$ such that for any $s_1, s_2 \in \mathcal{T}, \pi^*(s_1, a) = \pi^*(s_2, a)$ for all actions $a \in \mathcal{A}$. Define $\mathcal{C}$ to be a set of *canonical states* of the state space $\mathcal{S}$; for each equivalence class $\mathcal{T}$, there exists exactly one state in $\mathcal{C}$, that is, $|\mathcal{T} \cap \mathcal{C}| = 1$. We call $T : \mathcal{S} \to \mathcal{S}$ a canonical projection mapping if $T(s) \in \mathcal{S} \cap \mathcal{C}, \forall s \in \mathcal{T}$. Then, we call $s \in \mathcal{S} \cap \mathcal{C}$ the *canonical state* for state equivalence class $\mathcal{T}$. This concept is analogous to specific solutions in constraint elimination in constrained convex optimization; $\mathcal{C}$ is analogous to particular solutions, and T is analogous to a mapping from an arbitrary solution to the particular solution (by projecting from the nullspace).

State equivalence classes arise commonly in multi-agent settings due to the redundancy in state and action information exhibited by arbitrary ordering of agents, which may occur due to random initialization, lane changes, or turns in the mixed autonomy traffic setting. Such redundancy can lead to a combinatorial explosion in equivalent states. This selection of a canonical state effectively reduces the combinatorial number of states in each equivalence class to a single state. By learning for canonical states, the policy learns for all states in the respective equivalence classes, thereby reducing the sampling complexity of learning algorithms. However, the problem of finding and reducing such symmetries in MDPs is in general NP-Hard (Narayanamurthy and Ravindran, 2008).

We now demonstrate the use of this concept in the mixed autonomy problem. In the mixed autonomy traffic problem, a naive state representation, for instance a vector of vehicle positions and velocities, implicitly encodes an index for each human-driven vehicle. However, swapping the indices of any two human-driven vehicles yields a change in

the state representation but should not change the behavior of the learning agents (and could result from lane changes). Thus, with this state representation, the state equivalence classes are closed under pairwise swaps of non-automated or automated agents. In the ring road and figure-eight settings, we choose a canonical state which orders non-automated agents based on absolute distance, followed by automated agents ordered on absolute distance (thereby yielding a projection mapping $T(\cdot)$). Swapping the indices of two automated or two non-automated agents then leaves the resulting state unchanged. Unfortunately, even though ordering by absolute distance resolves the symmetry reduction problem in these special cases, this solution may not generalize to more complex network structures and is an open problem. For instance, absolute distance can be defined for the closed circuit networks studied in this chapter, but is less clearly defined for a large complex network or even for a road with many lanes.

An alternative approach not explored in this chapter is to discretize the space and treat the observation space as an image-like representation. Both representations preserve much of the spatial information in the state. While this approach resolves the issue of combinatorially similar states in multi-agent environments, an image-like representation exhibits different challenges, such as learning fine-grained and high-dimensional control. Levine et al. (2016) interprets an image-like representation as a partial observation, from which the true state information can be inferred or used in an end-to-end manner. Related is the pixel-to-torques problem; Wahlström et al. (2015) uses a deep auto-encoders to learn a closed-loop control policy from pixel information only, and could be composed with our problem setup to work with pixel inputs.

Numerically, the learning curve in the multi-lane setting (Figure 9) demonstrates that learning for state equivalence classes by ordering the observations speeds up the convergence of the RL method. The state equivalence classes also allows the algorithm to move past local maxima.

4.5 NETWORK CONFIGURATIONS

Three traffic networks are studied: a ring road, a multi-lane ring road, and a figure-eight. These are selected to isolate and study common traffic phenomena: 1) traffic shockwaves, 2) lane changes, which are large naturally occurring perturbations in the traffic flow, and 3) queuing at intersections.

Ring road. The ring road network consists of a circular lane with a specified length, similar to that of Sugiyama et al. (2008). A visual representation of the scenario can be seen in Figure 10a.

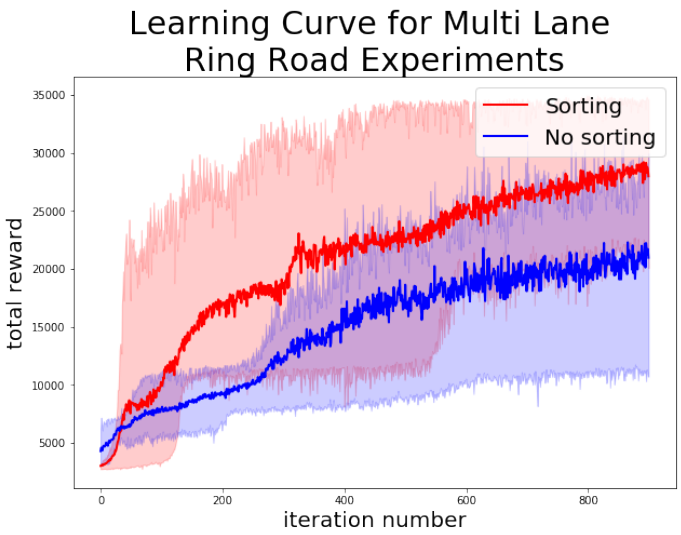


Figure 9: Learning curve for mixed autonomy in multi-lane traffic. Ordering the observations by position accelerates the empirical convergence of the RL algorithm and allows the algorithm to escape the local maximum of the unordered observations.

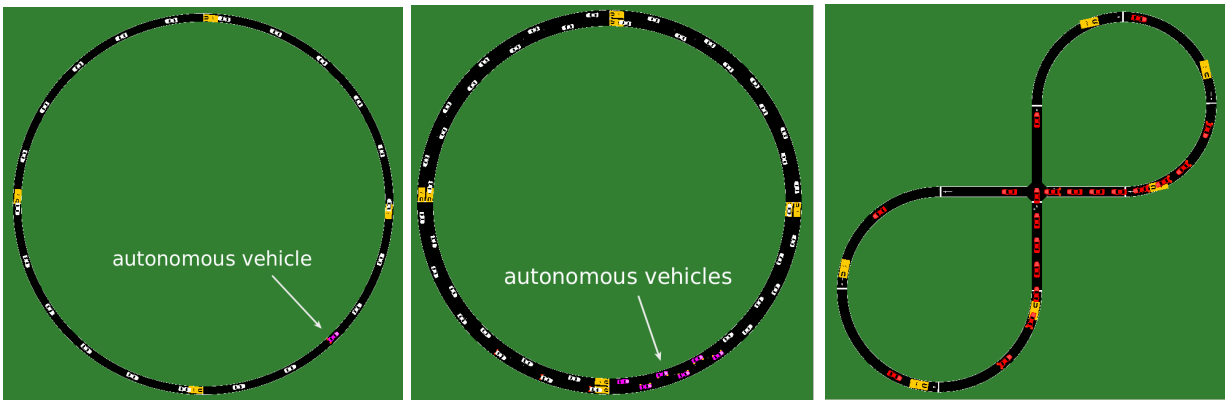
Multi-lane ring road. Multi-lane ring roads contain two or more lanes and permit lane changing (See Figure 10b).

Figure-eight. The figure-eight network is an extension of the ring road network–two rings, placed at opposite ends of the network, are connected by an intersection with road segments of length equal to the diameter of the rings. A visual representation of the scenario can be seen in Figure 10c. If two vehicles attempt to cross the intersection from opposing directions, the dynamics of these vehicles are constrained by right-of-way rules provided by SUMO.

4.6 MIXTURES OF AUTONOMY

This section details the specific experimental scenarios studied in this chapter. Detailed training details are provided in Section 4.11.

No autonomy. Experimental work by Sugiyama et al. (2008) involving 22 human-driven vehicles driving on a single-lane ring of length 230m has shown that similar dynamical systems produce instabilities (also called stop-and-go waves; see Figure 11a). Similarly, a figure-eight with a ring radius of 30m and total length of 402m is studied. The network contains a total of 14 vehicles. The intersection in this environment is not controlled



(a) Ring road. (b) Multi-lane ring road. (c) Figure-eight.

Figure 10: Network configurations for studies of emergent behaviors in mixed autonomy traffic.

by a traffic light; instead vehicles cross the intersection following a right-of-way model provided by SUMO to prevent crashes. In the absence of autonomous vehicles, human drivers begin queuing at the intersection, leading to a significant reduction in the average speed of vehicles in the network (see Figure 15a).

One automated vehicle among human-driven vehicles. A single human-driven vehicle is replaced by an automated vehicle, which is operated by a learned controller. In Section 4.7, we give an upper bound on the equilibrium velocity for vehicles in a single lane ring, which serves as a performance measure for our learned policy.

Strings of automated vehicles. This scenario considers multiple consecutive automated vehicles, in single- and multi-lane ring road settings. Recall that this chapter considers cooperative (centralized) control. In the single-lane case of Figure 10a, a total of 22 vehicles on a 230m length road are studied, and automated vehicle strings of size 3-11 are considered. We consider also a multi-lane scenario with twice as many lanes and vehicles.

All automated vehicles. All human vehicles are replaced with automated vehicles.

4.7 METRICS

Consider a single-lane ring road with n_a automated vehicles and n_h human-driven vehicles, with varying mixtures of autonomy, as described in Section 4.6. Let all the human-driven vehicles follow homogeneous deterministic longitudinal dynamics, such as any

car following model.

We consider two metrics. First, we consider the equilibrium velocity of the uniform flow equilibrium to the all-human traffic setting (no automated vehicles), which corresponds to an unstable equilibrium.

Second, we derive the following performance upper bound for the *Intelligent Driver Model* (IDM) (Treiber et al., 2000) and the bound can similarly be derived for other car following models. Denote s_e the equilibrium headway and v_e the equilibrium velocity. For IDM, the equilibrium headway is given by:

$$s_e(v) = s^*(v, 0)\left[1 - \left(\frac{v}{v_0}\right)^{\delta}\right]^{-1/2} \tag{11}$$

when in traffic equilibrium ($\dot{v}_\alpha = 0, \Delta v_\alpha = 0$) (Khalil, 1996). Denote $\hat{v}_e : \mathbb{R} \to \mathbb{R}$ the inverse mapping, from equilibrium headway to velocity.

Then, an upper bound on the average velocity of this mixed autonomy system, denoted $V^{max}_{n_a,n_h}$ is conjectured as follows:

$$V^{max}_{n_a,n_h} = \hat{v}_e\left(\frac{L - n_a L_{veh}}{n_h}\right), \tag{12}$$

where L is the total length of the ring and L_{veh} is the length of each vehicle.

Conceptually, this is equivalent to the equilibrium velocity attained if the automated vehicles were removed from the network and their cumulative length were added to one of the human vehicles. In the fully human-driven setting, the upper bound $V_{0,N}$ is exactly the equilibrium velocity of the system and, in such a setting, the bound is tight.

4.8 EMERGENT BEHAVIORS

In this section, we present the findings of studying the analytically challenging scenarios presented in Section 4.6, which push the limits of control theoretic analysis, including multi-lane settings and mixed autonomy intersection control. Each of the following behaviors result simply from maximizing the average velocity of a corresponding mixed autonomy traffic system.

Emergent traffic stabilization. In the presence of 21 string unstable human-driven vehicles on a single-lane road, a single automated vehicle eliminates the stop-and-go waves and stabilizes the ring when provided the reward function of Equation (10), as shown in

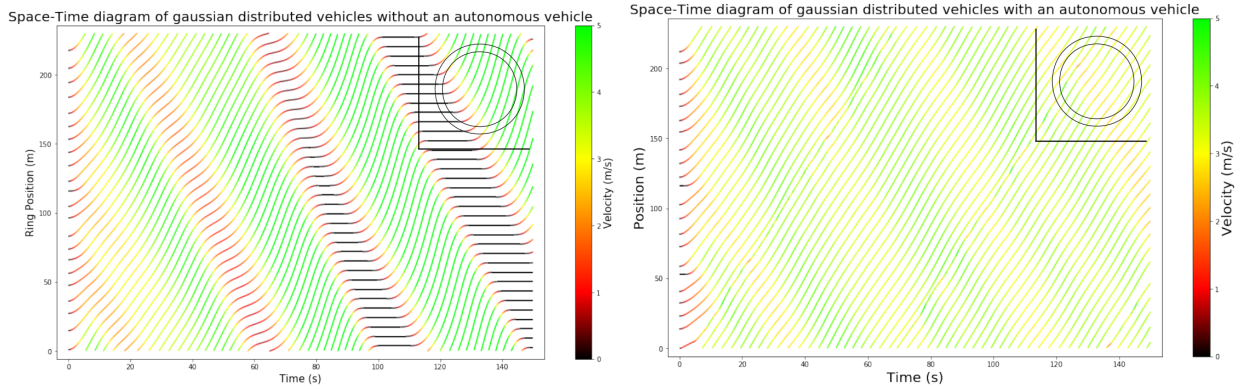
Figure 11. Surprisingly, we observe that a single damping component (via the policy for the automated vehicles) is sufficient for stabilizing the overall dynamical system, which consists of a cascade of 21 nonlinear systems. This implies that, in a setting with multiple automated vehicles, not much benefit is expected from spreading the vehicles out among the human vehicles. Experimentally, we thus study settings in which the automated vehicles are initialized in a string, rather than spread out among human vehicles. Our results demonstrate the potential for machine learning techniques to exceed explicit controllers obtained by classical control theory (S. Cui et al., 2017; Stern et al., 2017), which successfully stabilize the ring below the equilibrium velocity. In Chapter 6, we will further explore the use of machine learning techniques for designing controllers in mixed autonomy settings.

Emergent tail-gating and collision avoidance. Notably, in the single-lane traffic system of 22 total vehicles, the single automated vehicle learns to tailgate its preceding vehicle and uses a safe distance less than that of the human drivers (see the magenta vehicle in Figure 10a), thereby allowing the average velocity of vehicles in the ring to exceed the theoretic equilibrium velocity of the system (an unstable equilibrium in the absence of external control), as shown in Figure 12a. As predicted, the automated vehicle remains below the average velocity upper bound derived in Section 4.7 (black dotted line).

Additionally, the reward function in Equation (10) successfully encourages automated vehicles to avoid collisions. These emergent behaviors are observed even in the presence of additive Gaussian acceleration noise in the human driver models.

Emergent compressive platoon and load balancing. A total of 22 vehicles are placed in a ring road with a circumference of 230m. Strings of autonomous vehicles are placed consecutively, with between three and eleven autonomous vehicles. A string of consecutive autonomous vehicles learn to drive with a smaller headway than the human models, resulting in greater roadway utilization, thereby permitting a higher velocity for the overall system, as can be seen in Figure 12b.

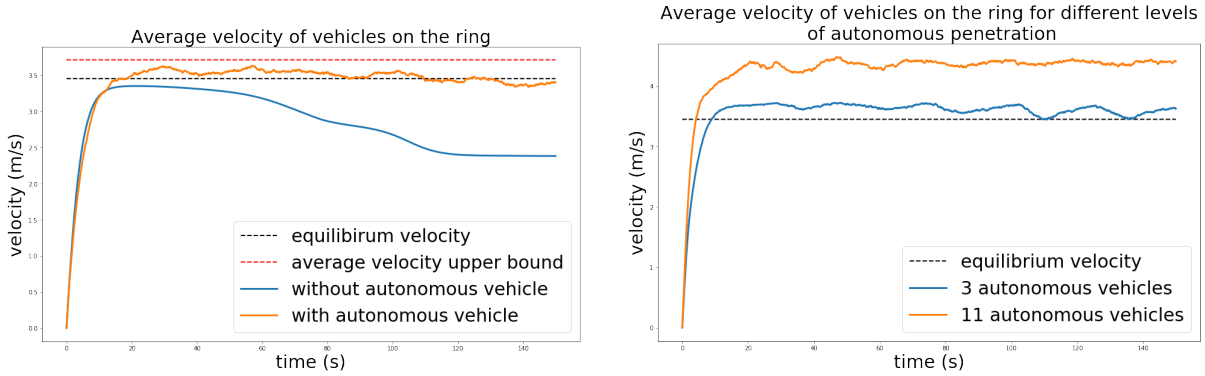
The single-lane multiple vehicle experiment is extended to the multiple lane setting, with 44 vehicles placed in a two-lane ring road of circumference 230m. In this setting, a string of six autonomous vehicles are initialized side-by-side (all in one lane). The human-driven vehicles follow IDM longitudinal control and SUMO's lane changing model for lateral control. In a multi-lane setting, in addition to platooning together, the automated vehicles learn to evenly distribute themselves among the lanes which ensures that each lane in the network benefits from the same level of platooning, as shown in Figure 10b (magenta vehicles). The resulting average velocity is 3.66 m/s, again exceeding the 3.45 m/s uniform flow equilibrium velocity with six automated vehicles and 38



(a) 22 human-driven vehicles.

(b) 21 human-driven, 1 automated vehicle.

Figure 11: A total of 22 vehicles are place in a ring road of length 230 m. Each line in the space-time diagrams represents the position of a specific vehicle as a function of time. Once a vehicle crosses the entire length of the ring, its position is reset to zero. **Left**: In the absence of automated vehicles, the inherently unstable human-driver vehicles experience stop-and-go traffic. **Right**: A single automated vehicle stabilizes a string of string-unstable vehicles, when provided with the reward function in Equation (10).



(a) Average velocity (1 AV).

(b) Average velocity (multiple AVs).

Figure 12: **Left**: In a single lane ring with one automated vehicle (AV) and 21 human driven vehicles, the automated vehicle tailgates its preceding vehicle with a safe distance less than that of the human drivers, thereby allowing the average velocity of vehicles in the ring to exceed the control theoretic equilibrium velocity of the all-human system. **Right**: Having multiple consecutive automated vehicles exhibits further improvements in average velocity, even for relatively few automated vehicles. This velocity increases as the level of autonomous penetration increases. At three autonomous vehicles, the average velocity settles at 3.70 m/s; at 11 autonomous vehicles, the average velocity settles at 4.44 m/s. The automated vehicles are found to platoon together.

human-driven vehicles. This experiment demonstrates the ability of the reinforcement learning approach to handle settings with discontinuous model dynamics, such as lane changes.

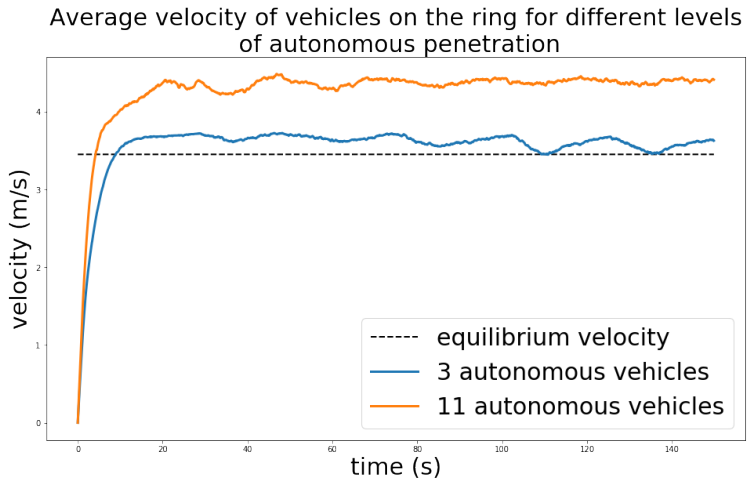
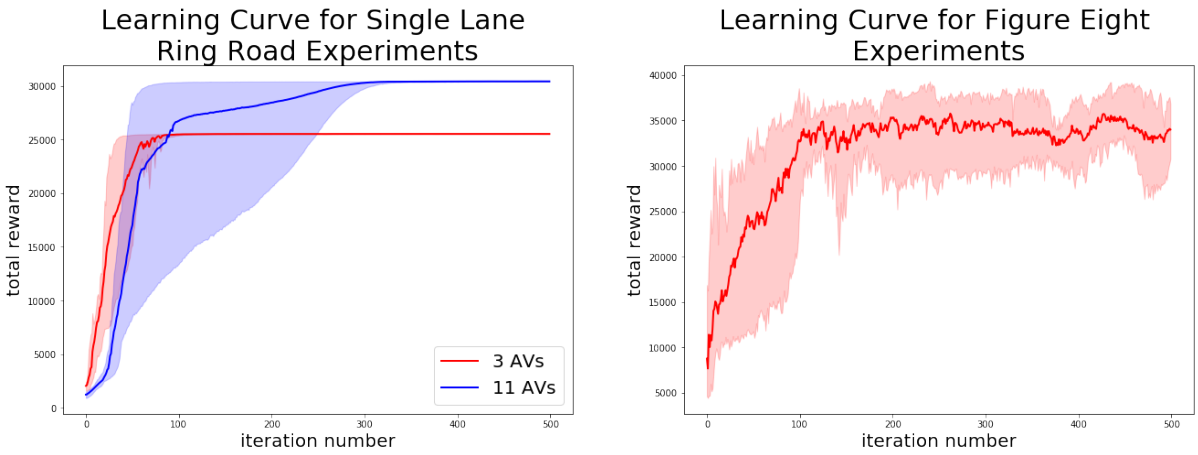


Figure 13: Velocity profile for single-lane ring road with multiple autonomous vehicles. The addition of autonomous vehicles permits in exceeding the uniform flow equilibrium velocity. This velocity increases as the level of autonomous penetration increases. At three autonomous vehicles, the average velocity settles at 3.70 m/s; at 11 autonomous vehicles, the average velocity settles at 4.44 m/s.

Emergent mixed autonomy platoon and intersection weaving. In a figure-eight scenario (described in Section 4.6), we examine the behavior of vehicles crossing the intersection. In the mixed autonomy setting, consisting of one automated vehicle and 13 human-driven vehicles, a single automated vehicle slows or stops entirely to allow all other vehicles to uniformly space themselves behind it. The automated vehicle exploits the dynamics of the human-driven vehicles to travel at a velocity *just* slow enough to allow all vehicles to pass through the intersection without stopping for the other direction of traffic, and *just* fast enough that all the available roadway (without causing weaving traffic) is used by the vehicles. That is, the automated vehicle forms a train or snake the length of half the length of the network and is an emergent mixed autonomy platoon, in which most of the platoon vehicles are human-driven. This achieves a system-level (average) velocity of 8.75 m/s with one automated vehicle and 13 human-driven vehicles, as compared to 5.48 m/s with no automated vehicles (60% improvement).

In the same experimental setup, when the number of automated vehicles is increased to 100% (14 of 14 vehicles), the vehicles weave through the intersection seamlessly at 14



(a) Single lane ring.

(b) Figure-eight.

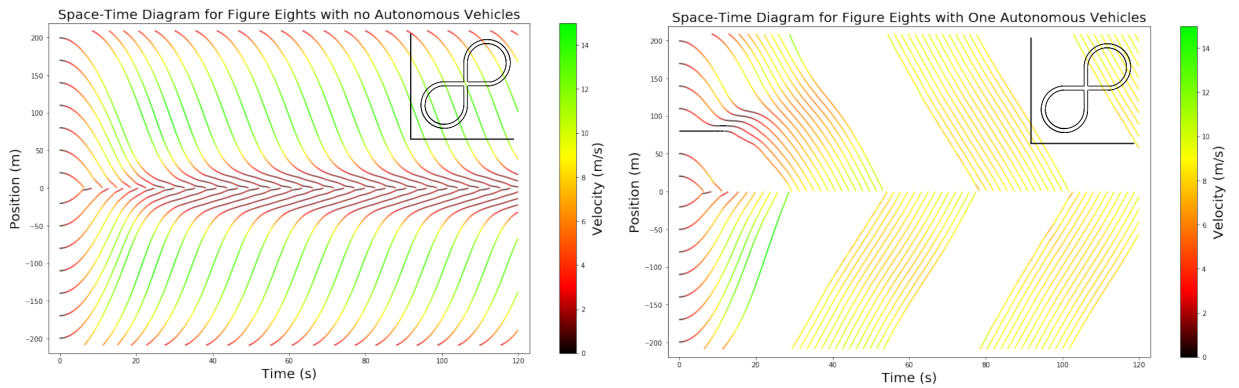
Figure 14: Learning curves for mixed autonomy traffic scenarios.

m/s (255% improvement).

For each of these mixed autonomy benchmarks, more investigation is required to understand the resulting learned behaviors and policies and thereby take steps towards a real-world deployment.

4.9 RELATED WORK

Mixed autonomy traffic. Numerous field operational tests have demonstrated that a small number of vehicles injected into a congested traffic system can result in a 13-40% reduction in fuel consumption. In particular, deployed eco-driving programs, which provide simple eco-friendly driving rules to individual drivers, in the Netherlands have observed 15-25% reduction in fuel consumption and improved safety (CIECA, 2007). Field studies in which a single vehicle is injected into real traffic on the California SR-91 freeway during PM peak traffic observed a 13% reduction in fuel consumption by following dynamic eco-driving rules (Barth and Boriboonsomsin, 2009). Recent field operational tests by Stern et al. (2017) demonstrated a reduction in fuel consumption of 40% by the insertion of an autonomous vehicle in ring traffic to dampen the famous ring instabilities displayed by Sugiyama et al. (2008) in their seminal experiment. These studies are characterized by hand-designed control laws for specific settings and do not consider the use of reinforcement learning, as is concerned by this chapter. However, the



(a) 14 human-driven vehicles.

(b) 13 human-driven, 1 automated vehicle.

Figure 15: A total of 14 vehicles are placed in a figure-eight road network of ring radius 30 m (total road length of 402 m). Position 0 in the above time-space diagrams denotes the location of the intersection, and vehicles are traveling towards it from two different directions. **Left**: In the absence of automated vehicles, the right-of-way model of the traffic simulator induces a stop-sign-like intersection behavior, generating queues at either sides of the intersection. **Right**: With a single automated vehicle, an emergent behavior observed is a *mixed autonomy platoon*, where nearly all vehicles in the platoon are human drivers, and yet there is a 60% increase in the average velocity of the overall system, as compared to non-automated setting.

impact on fuel consumption demonstrated by these studies motivates the (automatic) design of controllers for a small number of vehicles in complex traffic scenarios to achieve significant outcomes on the system.

Emergent behaviors and multi-agent systems. Multi-agent systems is a rich modeling framework, which can capture the complexities of organism dynamics and social organization (Weiss, 1999). These complex systems, which include many engineering and infrastructure systems, exhibit cascaded mixed discrete-continuous nonlinear dynamics, which are challenging to understand and control. When fused with learning frameworks such as deep RL, multi-agent learning systems (Busoniu et al., 2008) have the potential to exhibit many interesting emergent behaviors. Mordatch and Abbeel (2017) demonstrate the interesting emergence of compositional language from multi-agent systems. P. Peng et al. (2017) demonstrate emergent combat tactics in the multi-agent environment of combat scenarios in the real-time strategy game StarCraft, including hit-and-run and coordinated cover attacks. In this chapter, we study the behaviors which emerge when a few automated vehicles are mixed in with the dynamics of human-driven vehicles in traffic, in order to understand the potentially complex effects of partial penetration of automation in transportation systems.

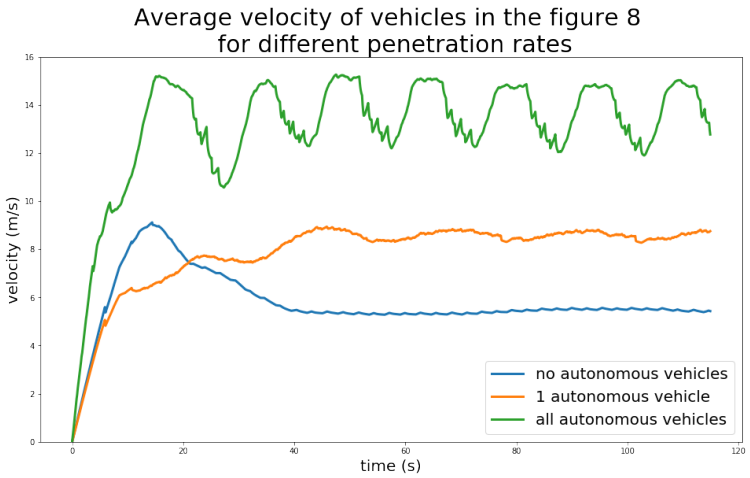


Figure 16: Velocity profile in the figure-eight for different levels of autonomous penetration. Similar to the platooning setting, the performance of the network improves as the number of autonomous vehicles improves. In particular, the inclusion of full autonomy almost triples the average velocity in the network from around 5 m/s in the absence of autonomy for around 14 m/s.

Reinforcement learning and traffic. In the context of traffic problems, RL has been explored by Belletti et al., 2018, who demonstrates the use of multi-agent RL for ramp metering and matches the performance of state-of-the-art techniques using feedback control. Additionally, Stevens and Yeh (2016) explore RL on traffic lights to increase traffic flow through intersections. Deep learning methods (Goodfellow et al., 2016) are used in several other aspects of transportation, including vision for self-driving cars (Bojarski et al., 2016), traffic flow prediction (Lv et al., 2015; Polson and Sokolov, 2017), and origin-destination prediction (Brébisson et al., 2015). For an overview on neural and non-neural statistical methods in transportation, we refer the reader to Karlaftis and Vlahogianni (2011). At a vehicular level, deep RL has been used to learn collision avoidance strategies (Kahn et al., 2017; Y. F. Chen et al., 2017). Recognizing the potential for reinforcement learning in complex domains such as traffic, this chapter focuses on studying the impact through an exploration of emergent behaviors when optimizing with a simple system objective.

Vehicle controller design. Classical techniques for vehicle controller design, such as adaptive cruise control (ACC) (Technical Committee ISO/TC 204, Intelligent transport systems, 2010; P. A. Ioannou and Chien, 1993; Vahidi and Eskandarian, 2003) and cooperative ACC (CACC) (Shladover, 2005; Rajamani and Zhu, 2002; Lu et al., 2004; Sheik-

holeslam and Desoer, 1992; Van Arem et al., 2006), typically optimize local metrics such as driver comfort or local fuel consumption. Controllers and conditions for linear stability have been proposed for suppressing stop-and-go waves in ACC models (Swaroop et al., 1994; C.-Y. Liang and H. Peng, 1999; C.-Y. Liang and H. Peng, 2000). CACC additionally permits the model-based design of compressive platoons, similar to those studied in this chapter. Approaches to vehicle controller design include model predictive control for steering control (Falcone et al., 2007; Falcone et al., 2008) and traffic control with automated vehicles (Baskar, 2009; M. Wang et al., 2014; Kamal et al., 2014), and frequency domain analysis (Naus et al., 2010; I. G. Jin and Orosz, 2014). Relatedly, model-based approaches have permitted the reservation system design and derivation of vehicle weaving behavior for fully automated intersections (Dresner and Stone, 2008; Miculescu and Karaman, 2014). However, there is no such result for a mixed autonomy intersection. Recently, a few studies have started to use formal techniques for controller design for system-level evaluation of mixed autonomy traffic, including state-space (S. Cui et al., 2017) and frequency domain analysis (Wu et al., 2017a). There are also several modeling- and simulation-based evaluations of mixed autonomy systems (Kesting et al., 2007; Y.-M. Yuan et al., 2009; Au et al., 2014). Despite these advances in controller design, many of these approaches are generally limited to simplified models, such as homogeneous, non-delayed, deterministic driver models, or restricted network configurations. The present chapter presents the first model-agnostic study of system-level optimization of mixed autonomy traffic through modern machine learning techniques.

4.10 CHAPTER SUMMARY

This chapter studies the use of state-of-the-art machine learning techniques on control of road traffic. In particular, we study emergent behaviors of automated vehicles in a mixed autonomy environment through the use of model-free RL methods. Understanding emergent behaviors in such complex systems is an important and often overlooked part of understanding the impact of automation on transportation systems. By investigating a variety of scenarios, behaviors such as stabilization, tailgating, platooning, and efficient vehicle spacing at intersections have emerged as useful behaviors for improving the system-level velocity of traffic. We additionally demonstrate that RL has the potential of sometimes exceeding performance measures based in control theory for fully-human traffic. Numerous extensions are still needed, however, to understand the potential impact of automation on transportation systems. Topics of future research include learning policies which generalize across a variety of traffic scenarios, studying other system-level

and local objectives, studying mixed control of automated vehicles and infrastructure, and studying effects at the scale of city networks. Important is the study of other system-level and local objectives, for instance trading off performance with energy consumption and comfort. Additionally, the design of objectives may require careful consideration of its implications on fairness (Lint et al., 2008, e.g.). Parameterized policies can also be used to study settings with a variable number of vehicles by introducing a pooling component to the policy, similar to (Mordatch and Abbeel, 2017), or by using recurrent policies (Graves et al., 2012).

4.11 EXPERIMENT DETAILS

The experiments conducted in this chapter utilize an open source library called Flow, presented in Chapter 6, which enables the use of reinforcement learning methods for traffic control. Three natural settings for benchmarking are the fully autonomous setting, the mixed autonomy setting, and the fully-human setting. Designed controllers for these settings may be implemented in Flow for additional benchmarking as well.

For all experiments, we use linear feature baselines as described in Duan et al. (2016) and Trust Region Policy Optimization (Schulman et al., 2015) for learning the policy, with discount factor $\gamma = 0.999$ and step size 0.01. A diagonal Gaussian MLP policy is used with hidden layers $(100, 50, 25)$ and tanh non-linearity. In all experiments in this chapter, a fully observable setting is assumed. The experiments are run on *Amazon Web Services* (AWS) *Elastic Compute Cloud* (EC2) instances of model `c4.2xlarge`, which have eight CPUs and 15 GB of memory.

5

VARIANCE REDUCTION FOR POLICY GRADIENT WITH ACTION-DEPENDENT FACTORIZED BASELINES

> Certainty is the mother of quiet and repose, and uncertainty the cause of variance and contentions.
>
> Sir Edward Coke, Institutes of the Lawes of England, 1628

Policy gradient methods have demonstrated potential for control of mixed autonomy systems, for example in mixed autonomy traffic as introduced in Chapter 4, and have enjoyed great success in deep reinforcement learning. However, these methods suffer from high variance of gradient estimates, which translates to poor sample complexity and high computational cost. Moreover, the high variance problem is particularly exasperated in problems with long horizons or high-dimensional action spaces, both characteristics of mixed autonomy systems. To mitigate this issue, we derive a bias-free action-dependent baseline for variance reduction which fully exploits the structural form of the stochastic policy itself and does not make any additional assumptions about the MDP. We demonstrate and quantify the benefit of the action-dependent baseline through both theoretical analysis as well as numerical results, including an analysis of the suboptimality of the optimal state-dependent baseline. The result is a computationally efficient policy gradient algorithm, which scales to high-dimensional control problems, as demonstrated by a synthetic 2000-dimensional target matching task. Our experimental results indicate that action-dependent baselines allow for faster learning on standard reinforcement learning benchmarks and high-dimensional hand manipulation and synthetic tasks. Finally, we show that the general idea of including additional information in baselines for improved variance reduction can be extended to partially observed and multi-agent tasks.

5.1 OVERVIEW

Deep reinforcement learning has achieved impressive results in recent years in domains such as video games from raw visual inputs (Mnih et al., 2015), board games (Silver et al., 2016), simulated control tasks (Schulman et al., 2016; Lillicrap et al., 2016; Rajeswaran et al., 2017a), and robotics (Levine et al., 2016). An important class of methods behind many of these success stories are policy gradient methods (Williams, 1992; Sutton et al., 2000; Kakade, 2002; Schulman et al., 2015; Mnih et al., 2016), which directly optimize parameters of a stochastic policy through local gradient information obtained by interacting with the environment using the current policy. Policy gradient methods operate by increasing the log probability of actions proportional to the future rewards influenced by these actions. On average, actions which perform better will acquire higher probability, and the policy's expected performance improves.

A critical challenge of policy gradient methods is the high variance of the gradient estimator. This high variance is caused in part due to difficulty in credit assignment to the actions which affected the future rewards. Such issues are further exacerbated in long horizon problems, where assigning credit properly becomes even more challenging. To reduce variance, a "baseline" is often employed, which allows us to increase or decrease the log probability of actions based on whether they perform better or worse than the average performance when starting from the same state. This is particularly useful in long horizon problems, since the baseline helps with temporal credit assignment by removing the influence of future actions from the total reward. A better baseline, which predicts the average performance more accurately, will lead to lower variance of the gradient estimator.

The key insight of this chapter is that when the individual actions produced by the policy can be decomposed into multiple factors, we can incorporate this additional information into the baseline to further reduce variance. In particular, when these factors are conditionally independent given the current state, we can compute a separate baseline for each factor, whose value can depend on all quantities of interest except that factor. This serves to further help credit assignment by removing the influence of other factors on the rewards, thereby reducing variance. In other words, information about the other factors can provide a better evaluation of how well a specific factor performs. Such factorized policies are very common, with some examples listed below.

- In continuous control and robotics tasks, multivariate Gaussian policies with a diagonal covariance matrix are often used. In such cases, each action coordinate can be considered a factor. Similarly, factorized categorical policies are used in

game domains like board games and Atari (Silver et al., 2016; Mnih et al., 2015).

- In multi-agent and distributed systems, each agent deploys its own policy, and thus the actions of each agent can be considered a factor of the union of all actions (by all agents). This is particularly useful in the recent emerging paradigm of centralized learning and decentralized execution (Lowe et al., 2017; Foerster et al., 2017). In contrast to the previous example, where factorized policies are a common design choice, in these problems they are dictated by the problem setting.

We demonstrate that action-dependent baselines consistently improve the performance compared to baselines that use only state information. The relative performance gain is task-specific, but in certain tasks, we observe significant speed-up in the learning process. We evaluate our proposed method on standard benchmark continuous control tasks, as well as on a high-dimensional door opening task with a five-fingered hand, a synthetic high-dimensional target matching task, on a blind peg insertion POMDP task, and a multi-agent communication task. We believe that our method will facilitate further applications of reinforcement learning methods in domains with extremely high-dimensional actions, including multi-agent systems. Videos and additional results of the chapter are available at https://sites.google.com/view/ad-baselines.

5.2 PRELIMINARIES

In this section, we establish the notations used throughout this chapter, as well as basic results for policy gradient methods, and variance reduction via baselines.

5.2.1 *Notation*

This chapter assumes a discrete-time Markov decision process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma)$, in which $\mathcal{S} \subseteq \mathbb{R}^n$ is an n-dimensional state space, $\mathcal{A} \subseteq \mathbb{R}^m$ an m-dimensional action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ a transition probability function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ a bounded reward function, $\rho_0 : \mathcal{S} \to \mathbb{R}_+$ an initial state distribution, and $\gamma \in (0, 1]$ a discount factor. The presented models are based on the optimization of a stochastic policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$ parameterized by θ. Let $\eta(\pi_\theta)$ denote its expected return: $\eta(\pi_\theta) = \mathbb{E}_\tau[\sum_{t=0} \gamma^t r(s_t, a_t)]$, where $\tau = (s_0, a_0, \ldots)$ denotes the whole trajectory, $s_0 \sim \rho_0(s_0)$, $a_t \sim \pi_\theta(a_t|s_t)$, and $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$ for all t. Our goal is to find the optimal policy $\operatorname{argmax}_\theta \eta(\pi_\theta)$. We will use $\hat{Q}(s_t, a_t)$ to describe samples of cumulative discounted return, and $Q(a_t, s_t)$ to describe a function approximation of $\hat{Q}(s_t, a_t)$. We will use "Q-function" when describing an abstract action-value function.

For a partially observable Markov decision process (POMDP), two more components are required, namely Ω, a set of observations, and $\mathcal{O} : \mathcal{S} \times \Omega \to \mathbb{R}_{\geqslant 0}$, the observation probability distribution. In the fully observable case, $\Omega \equiv \mathcal{S}$. Though the analysis in this chapter is written for policies over states, the same analysis can be done for policies over observations.

5.2.2 *The Score Function (SF) Estimator*

An important technique used in the derivation of the policy gradient is known as the score function (SF) estimator (Williams, 1992), which also comes up in the justification of baselines. Suppose that we want to estimate $\nabla_\theta \mathbb{E}_x[f(x)]$ where $x \sim p_\theta(x)$, and the family of distributions $\{p_\theta(x) : \theta \in \Theta\}$ has common support. Further suppose that $\log p_\theta(x)$ is continuous in θ. In this case we have

$$\begin{aligned}\nabla_\theta \mathbb{E}_x[f(x)] &= \nabla_\theta \int p_\theta(x) f(x) dx = \int p_\theta(x) \frac{\nabla_\theta p_\theta(x)}{p_\theta(x)} f(x) dx \\ &= \int p_\theta(x) \nabla_\theta \log p_\theta(x) f(x) dx = \mathbb{E}_x \left[\nabla_\theta \log p_\theta(x) f(x)\right]. \end{aligned} \tag{13}$$

5.2.3 *Policy Gradient*

The Policy Gradient Theorem (Sutton et al., 2000) states that

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_\tau \left[\sum_{t=0} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=t} \gamma^{t'-t} r_{t'} \right]. \tag{14}$$

where $\tau := (s_0, a_0, r_0, \ldots, s_t, a_t, r_t)$ denotes a trajectory induced by the policy π_θ. For convenience, define $\rho_\pi(s) = \sum_{t=0} \gamma^t p(s_t = s)$ as the state visitation frequency, and $\hat{Q}(s_t, a_t) = \sum_{t'=t} \gamma^{t'-t} r_{t'}$. We can rewrite the above equation (with abuse of notation) as

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\rho_\pi, \pi} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \hat{Q}(s_t, a_t) \right]. \tag{15}$$

It is further shown that we can reduce the variance of this gradient estimator without introducing bias by subtracting off a quantity dependent on s_t from $\hat{Q}(s_t, a_t)$ (Williams, 1992; Greensmith et al., 2004). See Section 5.7 for a derivation of the optimal state-

dependent baseline.

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\rho_\pi,\pi}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\left(\hat{Q}(s_t,a_t) - b(s_t)\right)\right] \tag{16}$$

This is valid because, applying the SF estimator in the opposite direction, we have

$$\mathbb{E}_{a_t}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t)\right] = \nabla_\theta \mathbb{E}_{a_t}\left[b(s_t)\right] = 0 \tag{17}$$

5.3 ACTION-DEPENDENT BASELINES

In practice there can be rich internal structure in the policy parameterization. For example, for continuous control tasks, a very common parameterization is to make $\pi_\theta(a_t|s_t)$ a multivariate Gaussian with diagonal variance, in which case each dimension a_t^i of the action a_t is conditionally independent of other dimensions, given the current state s_t. Another example is when the policy outputs a tuple of discrete actions with factorized categorical distributions. In the following subsections, we show that such structure can be exploited to further reduce the variance of the gradient estimator without introducing bias by changing the form of the baseline. Then, we derive the optimal action-dependent baseline for a class of problems and analyze the suboptimality of non-optimal baselines in terms of variance reduction. We then propose several practical baselines for implementation purposes. We conclude the section with the overall policy gradient algorithm with action-dependent baselines for factorized policies. We provide an exposition for situations when the conditional independence assumption does not hold, such as for stochastic policies with general covariance structures, in Section 5.11, and for compatibility with other variance reduction techniques in Section 5.12.

5.3.1 *Baselines for Policies with Conditionally Independent Factors*

In the following, we analyze action-dependent baselines for policies with conditionally independent factors. For example, multivariate Gaussian policies with a diagonal covariance structure are commonly used in continuous control tasks. Assuming an m-dimensional action space, we have $\pi_\theta(a_t|s_t) = \prod_{i=1}^m \pi_\theta(a_t^i|s_t)$. Hence

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\rho_\pi,\pi}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{Q}(s_t,a_t)\right] = \mathbb{E}_{\rho_\pi,\pi}\left[\sum_{i=1}^m \nabla_\theta \log \pi_\theta(a_t^i|s_t)\hat{Q}(s_t,a_t)\right] \tag{18}$$

In this case, we can set b_i, the baseline for the ith factor, to depend on all other actions in addition to the state. Let a_t^{-i} denote all dimensions other than i in a_t and denote the ith baseline by $b_i(s_t, a_t^{-i})$. Due to conditional independence and the score function estimator, we have

$$\mathbb{E}_{a_t}\left[\nabla_\theta \log \pi_\theta(a_t^i|s_t) b_i(s_t, a_t^{-i})\right] = \mathbb{E}_{a_t^{-i}}\left[\nabla_\theta \mathbb{E}_{a_t^i}\left[b_i(s_t, a_t^{-i})\right]\right] = 0 \tag{19}$$

Hence we can use the following gradient estimator

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\rho_\pi, \pi}\left[\sum_{i=1}^{m} \nabla_\theta \log \pi_\theta(a_t^i|s_t)\left(\hat{Q}(s_t, a_t) - b_i(s_t, a_t^{-i})\right)\right] \tag{20}$$

This is compatible with advantage function form of the policy gradient (Schulman et al., 2016):

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\rho_\pi, \pi}\left[\sum_{i=1}^{m} \nabla_\theta \log \pi_\theta(a_t^i|s_t)\hat{A}_i(s_t, a_t)\right] \tag{21}$$

where $\hat{A}_i(s_t, a_t) = Q(s_t, a_t) - b_i(s_t, a_t^{-i})$. Note that the policy gradient now consists of m component policy gradient terms, each with a different advantage term.

In Section 5.11, we show that the methodology also applies to general policy structures (for example, a Gaussian policy with a general covariance structure), where the conditional independence assumption does not hold. The result is bias-free albeit different baselines.

5.3.2 *Optimal action-dependent baseline*

In this section, we derive the optimal action-dependent baseline and show that it is better than the state-only baseline. We seek the optimal baseline to minimize the variance of the policy gradient estimate. First, we write out the variance of the policy gradient under any action-dependent baseline. Let us define $z_i := \nabla_\theta \log \pi_\theta(a_t^i|s_t)$ and the component policy gradient:

$$\nabla \eta_i(\pi_\theta) := \mathbb{E}_{\rho_\pi, \pi}\left[\nabla_\theta \log \pi_\theta(a_t^i|s_t)\left(\hat{Q}(s_t, a_t) - b_i(s_t, a_t^{-i})\right)\right]. \tag{22}$$

The optimal action-dependent baseline is then derived to be:

$$b_i^*(s_t, a_t^{-i}) = \frac{\mathbb{E}_{a_t^i}\left[\nabla_\theta \log \pi_\theta(a_t^i|s_t)^T \nabla_\theta \log \pi_\theta(a_t|s_t)\hat{Q}(s_t, a_t)\right]}{\mathbb{E}_{a_t^i}\left[\nabla_\theta \log \pi_\theta(a_t^i|s_t)^T \nabla_\theta \log \pi_\theta(a_t^i|s_t)\right]}. \tag{23}$$

See Section 5.8 for the full derivation. The optimal baseline is similar in form to that of the optimal state-only baseline, included in Section 5.7 for completeness. Since the optimal action-dependent baseline is in general different for different action coordinates, it is outside the family of state-dependent baselines, barring pathological cases.

5.3.3 *Suboptimality of the optimal state-dependent baseline*

How much do we reduce variance over a traditional baseline that only depends on state? We use the following notation:

$$Z_i := Z_i(s_t, a_t^{-i}) = \mathbb{E}_{a_t^i}\left[\nabla_\theta \log \pi_\theta(a_t^i|s_t)^T \nabla_\theta \log \pi_\theta(a_t^i|s_t)\right] \tag{24}$$

$$Y_i := Y_i(s_t, a_t^{-i}) = \mathbb{E}_{a_t^i}\left[\nabla_\theta \log \pi_\theta(a_t^i|s_t)^T \nabla_\theta \log \pi_\theta(a_t^i|s_t)\hat{Q}(s_t, a_t)\right] \tag{25}$$

Then, using Equation (64) (Section 5.9), we show the following improvement with the optimal action-dependent baseline:

$$I_{b=b^*(s)} = \sum_i \mathbb{E}_{\rho_\pi, a_t^{-i}}\left[\frac{1}{Z_i}\left(\frac{Z_i}{\sum_j Z_j}\sum_j Y_j - Y_i\right)^2\right] \tag{26}$$

See Sections 5.9 and 5.10 for the full derivation. We conclude that the optimal action-dependent baseline does not degenerate into the optimal state-dependent baseline. Equation (26) states that the variance difference is a weighted sum of the deviation of the per-component score-weighted marginalized Q (denoted Y_i) from the component weight (based on score only, not Q) of the overall aggregated marginalized Q values (denoted $\sum_j Y_j$). This suggests that the difference is particularly large when the Q function is highly sensitive to the actions, especially along those directions that influence the gradient the most. Our empirical results in Section 5.4 additionally demonstrate the benefit of action-dependent over state-only baselines.

5.3.4 *Marginalization of the global action-value function*

Using the previous theory, we now consider various baselines that could be used in practice and their associated computational cost.

MARGINALIZED Q BASELINE Even though the optimal state-only baseline is known, it is rarely used in practice (Duan et al., 2016). Rather, for both computational and conceptual benefit, the choice of $b(s_t) = \mathbb{E}_{a_t}[\hat{Q}(s_t, a_t)] = V(s_t)$ is often used. Similarly, we propose to use $b_i(s_t, a_t^{-i}) = \mathbb{E}_{a_t^i}\left[\hat{Q}(s_t, a_t)\right]$ which is the action-dependent analogue. In particular, when log probability of each policy factor is loosely correlated with the action-value function, then the proposed baseline is close to the optimal baseline.

$$I_{b=\mathbb{E}_{a_t^i}[\hat{Q}(a_t,s_t)]} = \sum_i \mathbb{E}_{\rho_\pi, a_t^{-i}}\left[Z_i\left(\mathbb{E}_{a^i}\left[\hat{Q}(a_t, s_t)\right] - \frac{\mathbb{E}_{a_t^i}\left[z_i^T z_i \hat{Q}(s_t, a_t)\right]}{\mathbb{E}_{a_t^i}\left[z_i^T z_i\right]}\right)^2\right] \approx 0 \quad (27)$$

when $\mathbb{E}_{a_t^i}\left[z_i^T z_i \hat{Q}(s_t, a_t)\right] \approx \mathbb{E}_{a_t^i}\left[z_i^T z_i\right] \mathbb{E}_{a_t^i}\left[\hat{Q}(s_t, a_t)\right]$.

This has the added benefit of requiring learning only one function approximator, for estimating $Q(s_t, a_t)$, and implicitly using it to obtain the baselines for each action coordinate. That is, $Q(s_t, a_t)$ is a function approximating samples $\hat{Q}(s_t, a_t)$.

MONTE CARLO MARGINALIZED Q BASELINE After fitting $Q_{\pi_\theta}(s_t, a_t)$ we can obtain the baselines through Monte Carlo estimates:

$$b_i(s_t, a_t^{-i}) = \frac{1}{M}\sum_{j=0}^{M} Q_{\pi_\theta}(s_t, (a_t^{-i}, \alpha_j)) \quad (28)$$

where $\alpha_j \sim \pi_\theta(a_t^i | s_t)$ are samples of the action coordinate i. In general, any function may be used to aggregate the samples, so long as it does not depend on the sample value a_t^i. For instance, for discrete action dimensions, the sample max can be computed instead of the mean.

MEAN MARGINALIZED Q BASELINE Though we reduced the computational burden from learning m functions to one function, the use of Monte Carlo samples can still be computationally expensive. In particular, when using deep neural networks to approximate the Q-function, forward propagation through the network can be even more computationally expensive than stepping through a fast simulator (e.g. MuJoCo). In such

settings, we further propose the following more computationally practical baseline:

$$b_i(s_t, a_t^{-i}) = Q_{\pi_\theta}(s_t, (a_t^{-i}, \bar{a}_t^i)) \tag{29}$$

where $\bar{a}_t^i = \mathbb{E}_{\pi_\theta}\left[a_t^i\right]$ is the average action for coordinate i.

5.3.5 *Final algorithm*

The final practical algorithm for fully factorized policies is as follows.

Algorithm 1 Policy gradient for factorized policies using action-dependent baselines

Require: number of iterations N, batch size B, initial policy parameters θ
  Initialize action-value function estimate $Q_{\pi_\theta}(s_t, a_t) \equiv 0$ and policy π_θ
  for j in $\{1, \ldots, N\}$ **do**
    Collect samples: $(s_t, a_t)_{t \in \{1,\ldots,B\}}$
    Compute baseline: $b_i(s_t, a_t^{-i}) = \mathbb{E}_{a_t^i}\left[\hat{Q}(s_t, a_t)\right]$ for $i \in \{1, \ldots, m\}$ [e.g. Equations (28-29)]
    Compute advantages: $\hat{A}_i(s_t, a_t) := \hat{Q}(s_t, a_t) - b_i(s_t, a_t^{-i}), \forall t$
    Perform a policy update step on θ using $\hat{A}_i(s_t, a_t)$ [Equation (21)]
    Update action-value function approximation with current batch: $Q_{\pi_\theta}(s_t, a_t)$
  end for

Computing the baseline can be done with either proposed technique in Section 5.3.4. A similar algorithm can be written for general policies (Section 5.11), which makes no assumptions on the conditional independence across action dimensions.

5.4 EXPERIMENTS AND RESULTS

CONTINUOUS CONTROL BENCHMARKS Firstly, we present the results of the proposed action-dependent baselines on popular benchmark tasks. These tasks have been widely studied in the deep reinforcement learning community (Duan et al., 2016; Gu et al., 2017; Lillicrap et al., 2016; Rajeswaran et al., 2017b). The studied tasks include the hopper, half-cheetah, and ant locomotion tasks simulated in MuJoCo (Todorov et al., 2012).[1] In addition to these tasks, we also consider a door opening task with a high-dimensional multi-fingered hand, introduced in Rajeswaran et al. (2017a), to study the

1 We used physics parameters as recommended in Rajeswaran et al. (2017b) and use the MuJoCo 1.5 simulator. Thus the reward numbers may not be consistent with numbers previously reported in literature.

effectiveness of the proposed approach in high-dimensional tasks. Figure 17 presents the learning curves on these tasks. We compare the action-dependent baseline with a baseline that uses only information about the states, which is the most common approach in the literature. We observe that the action-dependent baselines perform consistently better.

A popular baseline parameterization choice is a linear function on a small number of non-linear features of the state (Duan et al., 2016), especially for policy gradient methods. In this work, to enable a fair comparison, we use a Random Fourier Feature representation for the baseline (Rahimi and Recht, 2007; Rajeswaran et al., 2017b). The features are constructed as: $y(x) = \sin(\frac{1}{\nu}Px + \varphi)$ where P is a matrix with each element independently drawn from the standard normal distribution, φ is a random phase shift in $[-\pi, \pi)$, and ν is a bandwidth parameter. These features approximate the RKHS features under a RBF kernel. Using these features, the baseline is parameterized as $b = w^T y(x)$ where x are the appropriate inputs to the baseline, and w are trainable parameters. P and φ are not trained in this parameterization. Such a representation was chosen for two reasons: (a) we wish to have the same number of trainable parameters for all the baseline architectures, and not have more parameters in the action-dependent case (which has a larger number of inputs to the baseline); (b) since the final representation is linear, it is possible to accurately estimate the optimal parameters with a Newton step, thereby alleviating the results from confounding optimization issues. For policy optimization, we use a variant of the natural policy gradient method as described in Rajeswaran et al. (2017b). See Section 5.14 for further experimental details.

CHOICE OF ACTION-DEPENDENT BASELINE FORM Next, we study the influence of computing the baseline by using empirical averages sampled from the Q-function versus using the mean-action of the action-coordinate for computing the baseline (both described in 5.3.4). In our experiments, as shown in Figure 18 we find that the two variants perform comparably, with the latter performing slightly better towards the end of the learning process. This suggests that though sampling from the Q-function might provide a better estimate of the conditional expectation in theory, function approximation from finite samples injects errors that may degrade the quality of estimates. In particular, sub-sampling from the Q-function is likely to produce better results if the learned Q-function is accurate for a large fraction of the action space, but getting such high quality approximations might be hard in practice.

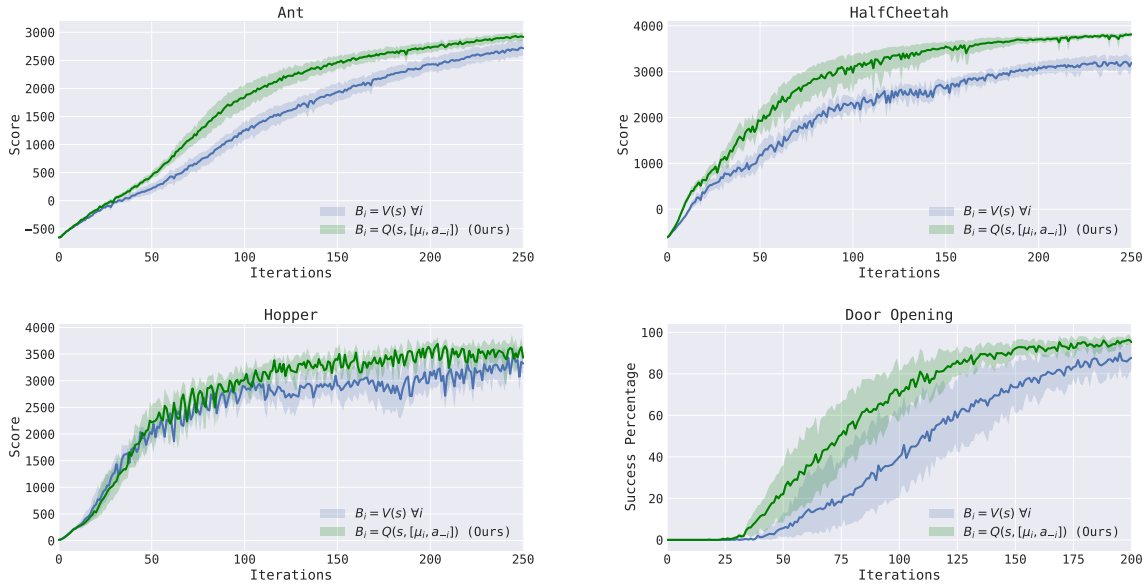


Figure 17: Comparison between value function baseline and action-conditioned baseline on various continuous control tasks. Action-dependent baseline performs consistently better across all the tasks.

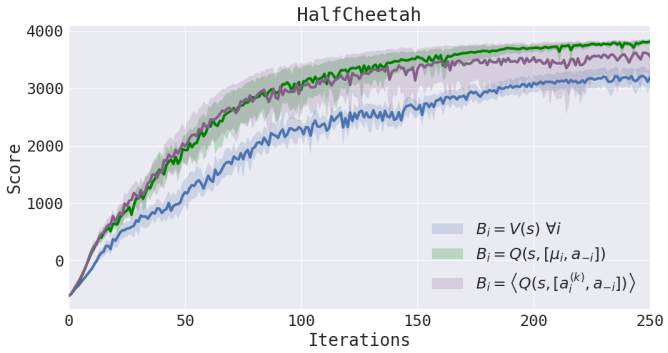


Figure 18: Variants of the action-dependent baseline that use: (i) sampling from the Q-function to estimate the conditional expectation; (ii) Using the mean action to form a linear approximation to the conditional expectation. We find that both variants perform comparably, with the latter being more computationally efficient.

HIGH-DIMENSIONAL ACTION SPACES Intuitively, the benefit of the action-dependent baseline can be greater for higher dimensional problems. We show this effect on a sim-

ple synthetic example called m-DimTargetMatching. The example is a one-step MDP consisting of a single state, $\mathcal{S} = \{0\}$, an m-dimensional action space, $\mathcal{A} = \mathbb{R}^m$, and a fixed vector $c \in \mathbb{R}^m$. The reward is given as the negative squared ℓ_2 loss of the action vector, $r(s, a) = -\|a - c\|_2^2$. The optimal action is thus to match the given vector by selecting $a = c$. The results for the demonstrative example are shown in Table 2, which shows that the action-dependent baseline successfully improves convergence more for higher dimensional problems than lower dimensional problems. Due to the lack of state information, the linear baseline reduces to whitening the returns. The action-dependent baseline, on the other hand, allows the learning algorithm to assess the advantage of each individual action dimension by utilizing information from all other action dimensions. Additionally, this experiment demonstrates that our algorithm scales well computationally to high-dimensional problems.

Action	Solve time (iterations)			% speed	Solution
dimensions	Action-dependent	State-dependent	Delta	improvement	threshold
12	45.6	45.6	0	0.0%	-0.01
100	136	150	14	**9.3%**	-0.25
400	268.2	304	35.8	**11.8%**	-0.99
2000	595.5	671.5	76	**11.3%**	-4.96

Table 2: Shown are the results for the synthetic high-dimensional target matching task (5 seeds), for 12 to 2000 dimensional action spaces. At high dimensions, the linear feature action-dependent baseline provides notable and consistent variance reduction, as compared to a linear feature baseline, resulting in around 10% faster convergence. For the corresponding learning curves, see Section 5.13.

PARTIALLY OBSERVABLE AND MULTI-AGENT TASKS Finally, we also consider the extension of the core idea of using global information, by studying a POMDP task and a multi-agent task. We use the blind peg-insertion task which is widely studied in the robot learning literature (Montgomery and Levine, 2016). The task requires the robot to insert the peg into the hole (slot), but the robot is blind to the location of the hole. Thus, we expect a searching behavior to emerge from the robot, where it learns that the hole is present on the table and performs appropriate sweeping motions until it is able to find the hole. In this case, we consider a baseline that is given access to the location of the hole. We observe that a baseline with this additional information enables faster learning. For the multi-agent setting, we analyze a two-agent particle environment task

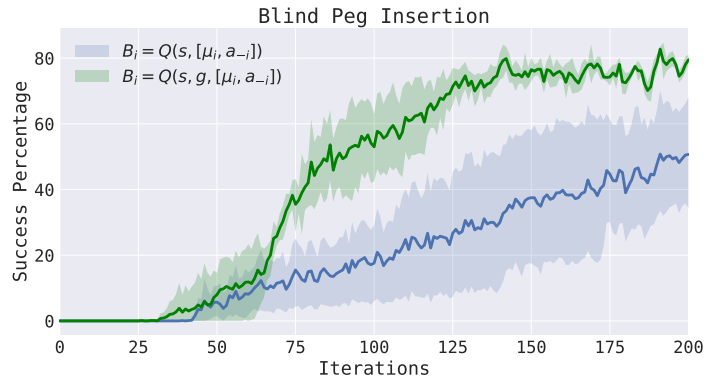
in which the goal is for each agent to reach their goal, where their goal is known by the other agent and they have a continuous communication channel. Similar training procedures have been employed in recent related works (Lowe et al., 2017; Levine et al., 2016). Figure 19 shows that including the inclusion of information from other agents into the action-dependent baseline improves the training performance, indicating that variance reduction may be key for multi-agent reinforcement learning.

5.5 RELATED WORKS

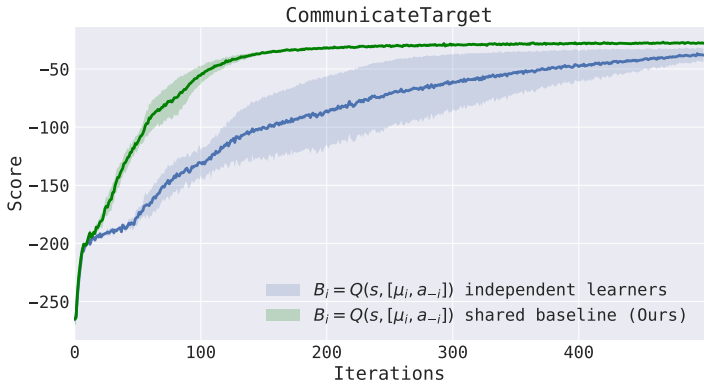
Three main classes of methods for reinforcement learning include value-based methods (Watkins and Dayan, 1992), policy-based methods (Williams, 1992; Kakade, 2002; Schulman et al., 2015), and actor-critic methods (Konda and Tsitsiklis, 2000; Peters and Schaal, 2008; Mnih et al., 2016). Value-based and actor-critic methods usually compute a gradient of the objective through the use of critics, which are often biased, unless strict compatibility conditions are met (Sutton et al., 2000; Konda and Tsitsiklis, 2000). Such conditions are rarely satisfied in practice due to the use of stochastic gradient methods and powerful function approximators. In comparison, policy gradient methods are able to compute an unbiased gradient, but suffer from high variance. Policy gradient methods are therefore usually less sample efficient, but can be more stable than critic-based methods (Duan et al., 2016).

A large body of work has investigated variance reduction techniques for policy gradient methods. One effective method to reduce variance without introducing bias is through using a baseline, which has been widely studied (Sutton and Barto, 1998; Weaver and Tao, 2001; Greensmith et al., 2004; Schulman et al., 2016). However, fully exploiting the factorizability of the policy probability distribution to further reduce variance has not been studied. Recently, methods like Q-Prop (Gu et al., 2017) make use of an action-dependent control variate, a technique commonly used in Monte Carlo methods and recently adopted for RL. Since Q-Prop utilizes off-policy data, it has the potential to be more sample efficient than pure on-policy methods. However, Q-prop is significantly more computationally expensive, since it needs to perform a large number of gradient updates on the critic using off-policy data, thus not suitable with fast simulators. In contrast, our formulation of action-dependent baselines has little computational overhead, and improves the sample efficiency compared to on-policy methods with state-only baseline.

The idea of using additional information in the baseline or critic has also been studied in other contexts. Methods such as Guided Policy Search (Levine et al., 2016; Mordatch



(a) Success percentage on the blind peg insertion task. The policy still acts on the observations and does not know the hole location. However, the baseline has access to this goal information, in addition to the observations and action, and helps to speed up the learning. By comparison, in blue, the baseline has access only to the observations and actions.



(b) Training curve for multi-agent communication task with two agents. Two policies are simultaneously trained, one for each agent. Each policy acts on the observations of its respective agent only. However, the shared baseline has access to the other agent's state and action, in addition to its own state and action, and results in considerably faster training. By comparison, in blue, the independent learners baseline has access to only a single agent's state and action.

Figure 19: Experiments with additional information in the baseline.

et al., 2015) and variants train policies that act on high-dimensional observations like images, but use a low dimensional encoding of the problem like joint positions during the training process. Using the structure in the policy parameterization itself to enhance the learning speed via a marginalised baseline that conditions on other independent action dimensions, as we do in this work, was independently proposed in COMA by Foerster et al. (2017), who evaluated it in the context of multi-agent learning. By contrast, our method is applicable to continuous action spaces, and we derive the optimal baseline and analyze the variance. We also propose a lower bias variant using GAE. For an overview of other developments in action-dependent baselines, we refer the reader to Tucker et al. (2018).

5.6 CHAPTER SUMMARY

An action-dependent baseline enables using additional signals beyond the state to achieve bias-free variance reduction. In this work, we consider both conditionally independent policies and general policies, and derive an optimal action-dependent baseline. We provide analysis of the variance reduction improvement over non-optimal baselines, including the traditional optimal baseline that only depends on state. We additionally propose several practical action-dependent baselines which perform well on a variety of continuous control tasks and synthetic high-dimensional action problems. The use of additional signals beyond the local state generalizes to other problem settings, for instance in POMDP and multi-agent tasks. In future work, we propose to investigate related methods in such settings on large-scale problems.

5.7 DERIVATION OF THE OPTIMAL STATE-DEPENDENT BASELINE

We provide a derivation of the optimal state-dependent baseline, which minimizes the variance of the policy gradient estimate, and is based in Greensmith et al., 2004, Theorem 8. More precisely, we minimize the trace of the covariance of the policy gradient; that is, the sum of the variance of the components of the vectors. Recall the policy gradient expression with a state-dependent baseline:

$$\nabla_\theta \eta(\pi_\theta) := \mathbb{E}_{\rho_\pi, \pi}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\left(\hat{Q}(s_t, a_t) - b(s_t)\right)\right] \tag{30}$$

Denote g to be the associated random variable, that is, $\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\rho_\pi,\pi}[g]$:

$$g := \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\hat{Q}(s_t, a_t) - b(s_t)\right), \quad a_t \sim \pi_\theta(a_t|s_t), s_t \sim \rho_\pi(s_t) \tag{31}$$

The variance of the policy gradient is:

$$\begin{aligned}
\mathrm{Var}(g) &= \mathbb{E}_{\rho_\pi,\pi}\left[(g - \mathbb{E}_{\rho_\pi,\pi}[g])^T (g - \mathbb{E}_{\rho_\pi,\pi}[g])\right] && (32)\\
&= \mathbb{E}_{\rho_\pi,\pi}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)^T \nabla_\theta \log \pi_\theta(a_t|s_t)\right] b(s_t)^2 && (33)\\
&\quad - 2\mathbb{E}_{\rho_\pi,\pi}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)^T \nabla_\theta \log \pi_\theta(a_t|s_t) \hat{Q}(s_t, a_t)\right] b(s_t) && (34)
\end{aligned}$$

Note that $\mathbb{E}[\eta(\pi_\theta)])$ contains a bias-free term, by the score function argument, which then does not affect the minimizer. Terms which do not depend on $b(s_t)$ also do not affect the minimizer.

$$\begin{aligned}
\frac{\partial}{\partial b}[\mathrm{Var}(g)] &= 0 && (35)\\
&= 2\mathbb{E}_{\rho_\pi,\pi}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)^T \nabla_\theta \log \pi_\theta(a_t|s_t)\right] b(s_t) && (36)\\
&\quad - 2\mathbb{E}_{\rho_\pi,\pi}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)^T \nabla_\theta \log \pi_\theta(a_t|s_t) \hat{Q}(s_t, a_t)\right] && (37)
\end{aligned}$$

$$\implies b^*(s_t) = \frac{\mathbb{E}_{\rho_\pi,\pi}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)^T \nabla_\theta \log \pi_\theta(a_t|s_t) \hat{Q}(s_t, a_t)\right]}{\mathbb{E}_{\rho_\pi,\pi}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)^T \nabla_\theta \log \pi_\theta(a_t|s_t)\right]} \tag{38}$$

5.8 Derivation of the optimal action-dependent baseline

We derive the optimal action-dependent baseline, which minimizes the variance of the policy gradient estimate. First, we write out the variance of the policy gradient under any action-dependent baseline. Recall the following notations: we define $z_i := \nabla_\theta \log \pi_\theta(a_t^i|s_t)$ and the component policy gradient:

$$\nabla \eta_i(\pi_\theta) := \mathbb{E}_{\rho_\pi,\pi}\left[\nabla_\theta \log \pi_\theta(a_t^i|s_t) \left(\hat{Q}(s_t, a_t) - b_i(s_t, a_t^{-i})\right)\right]. \tag{39}$$

Denote g_i to be the associated random variables:

$$g_i := \nabla_\theta \log \pi_\theta(a_t^i|s_t) \left(\hat{Q}(s_t, a_t) - b_i(s_t, a_t^{-i})\right), \quad a_t \sim \pi_\theta(a_t|s_t), s_t \sim \rho_\pi(s_t), \tag{40}$$

such that

$$\nabla_\theta \eta(\pi_\theta) = \nabla_\theta \left[\sum_{i=1}^{m} \eta_i(\pi_\theta)\right] = \mathbb{E}_{\rho_\pi,\pi}\left[\sum_{i=1}^{m} g_i\right]. \tag{41}$$

The overall variance is as follows:

$$\text{Var}(\sum_{i=1}^{m} g_i) = \sum_i \text{Var}(g_i) + \sum_i \sum_{j\neq i} \text{Cov}(g_i, g_j) \tag{42}$$

$$= \sum_i \text{Var}(g_i) + \sum_i \sum_{j\neq i} \mathbb{E}_{\rho_\pi,\pi}\left[g_i^T g_j\right] - \mathbb{E}_{\rho_\pi,\pi}\left[g_i\right]^T \mathbb{E}_{\rho_\pi,\pi}\left[g_j\right] \tag{43}$$

Note that the third term does not affect the optima, since the baseline is unbiased. A closer look at the second term reveals that only part of it affects the optima. Without loss of generality, a single subterm of the second term can be written as:

$$\mathbb{E}_{\rho_\pi,\pi}\left[g_i^T g_j\right] = \mathbb{E}_{\rho_\pi,\pi}\left[z_i^T z_j \left(\hat{Q}(s_t, a_t) - b_i(s_t, a_t^{-i})\right)\left(\hat{Q}(s_t, a_t) - b_j(s_t, a_t^{-j})\right)\right] \tag{44}$$

We can check the derivative with respect to one of the baselines (without loss of generality, choose b_i):

$$\frac{\partial}{\partial b_i}\mathbb{E}_{\rho_\pi,\pi}\left[g_i^T g_j\right] = \mathbb{E}_{\rho_\pi,\pi}\left[z_i^T z_j \left(-\hat{Q}(s_t, a_t) + b_j(s_t, a_t^{-j})\right)\right] \tag{45}$$

In particular, the latter term is always zero, and therefore never affects the optima, because:

$$\mathbb{E}_{\rho_\pi,\pi}\left[z_i^T z_j \left(b_j(s_t, a_t^{-j})\right)\right] = \mathbb{E}_{a^{-j}}\left[z_i^T \mathbb{E}_{a^j}\left[z_j\right]\left(b_j(s_t, a_t^{-j})\right)\right] \tag{46}$$

$$= \mathbb{E}_{a^{-j}}\left[z_i^T 0 \left(b_j(s_t, a_t^{-j})\right)\right] \tag{47}$$

$$= 0 \tag{48}$$

Then, assembling Equation 43 and the remainder of Equation 45, and following analysis

similar to Section 5.7, we have:

$$\frac{\partial}{\partial b_i}\mathrm{Var}(\sum_{i=1}^{m} g_i) = \mathbb{E}_{\rho_\pi, a_t^{-i}}\left[-2\mathbb{E}_{a_t^i}\left[z_i^\intercal z_i \hat{Q}(s_t, a_t)\right] + 2b_i(s_t, a_t^{-i})\mathbb{E}_{a_t^i}\left[z_i^\intercal z_i\right]\right] \tag{49}$$

$$+ 2\sum_{j\neq i}\mathbb{E}_{\rho_\pi,\pi}\left[z_i^\intercal z_j\left(-\hat{Q}(s_t, a_t)\right)\right] \tag{50}$$

$$= 2\mathbb{E}_{\rho_\pi, a_t^{-i}}\left[b_i(s_t, a_t^{-i})\mathbb{E}_{a_t^i}\left[z_i^\intercal z_i\right]\right] - 2\sum_{j}\mathbb{E}_{\rho_\pi,\pi}\left[z_i^\intercal z_j \hat{Q}(s_t, a_t)\right] \tag{51}$$

$$= 2\mathbb{E}_{\rho_\pi, a_t^{-i}}\left[b_i(s_t, a_t^{-i})\mathbb{E}_{a_t^i}\left[z_i^\intercal z_i\right]\right] - 2\mathbb{E}_{\rho_\pi,\pi}\left[z_i^\intercal z \hat{Q}(s_t, a_t)\right] = 0 \tag{52}$$

where $\sum_j z_j = z = \nabla_\theta \log \pi_\theta(a_t|s_t)$.

The optimal action-dependent baseline follows:

$$b_i^*(s_t, a_t^{-i}) = \frac{\mathbb{E}_{a^i}\left[z_i^\intercal z Q\right]}{\mathbb{E}_{a^i}\left[z_i^\intercal z_i\right]} \tag{53}$$

$$= \frac{\mathbb{E}_{a_t^i}\left[\nabla_\theta \log \pi_\theta(a_t^i|s_t)^\intercal \nabla_\theta \log \pi_\theta(a_t|s_t)\hat{Q}(s_t, a_t)\right]}{\mathbb{E}_{a_t^i}\left[\nabla_\theta \log \pi_\theta(a_t^i|s_t)^\intercal \nabla_\theta \log \pi_\theta(a_t^i|s_t)\right]} \tag{54}$$

5.9 Derivation of variance reduction improvement

We now turn to quantifying the reduction in variance of the policy gradient estimate under the optimal baseline derived above. Let $\mathrm{Var}^*(\sum_i g_i)$ denote the variance resulting from the optimal action-dependent baseline, and let $\mathrm{Var}(\sum_i g_i)$ denote the variance resulting from another baseline $b = (b_i(s_t, a_t^{-i}))_{i\in[m]}$, which may be suboptimal or action-independent. Recall the notation:

$$Z_i := Z_i(s_t, a_t^{-i}) = \mathbb{E}_{a_t^i}\left[\nabla_\theta \log \pi_\theta(a_t^i|s_t)^\intercal \nabla_\theta \log \pi_\theta(a_t^i|s_t)\right] \tag{55}$$

$$Y_i := Y_i(s_t, a_t^{-i}) = \mathbb{E}_{a_t^i}\left[\nabla_\theta \log \pi_\theta(a_t^i|s_t)^\intercal \nabla_\theta \log \pi_\theta(a_t^i|s_t)\hat{Q}(s_t, a_t)\right] \tag{56}$$

$$X_i := X_i(s_t, a_t^{-i}) = \mathbb{E}_{a_t^i}\left[\nabla_\theta \log \pi_\theta(a_t^i|s_t)^\intercal \nabla_\theta \log \pi_\theta(a_t^i|s_t)\hat{Q}(s_t, a_t)^2\right] \tag{57}$$

Finally, define the variance improvement $I_b := \mathrm{Var}(\sum_i g_i) - \mathrm{Var}^*(\sum_i g_i)$. Using these definitions, the variance can be re-written as:

$$\mathrm{Var}(\sum_i g_i) = \sum_i \mathbb{E}_{\rho_\pi, a_t^{-i}} \left[X_i - 2b_i(s_t, a_t^{-i}) Y_i + b_i(s_t, a_t^{-i})^2 Z_i \right] - M \tag{58}$$

Furthermore, the variance of the gradient with the optimal baseline can be written as

$$\mathrm{Var}^*(\sum_i g_i) = \sum_i \mathbb{E}_{\rho_\pi, a_t^{-i}} \left[X_i - \frac{Y_i^2}{Z_i} \right] - M \tag{59}$$

The difference in variance can be calculated as:

$$I_b := \sum_i \left(\mathbb{E}_{\rho_\pi, a_t^{-i}} \left[X_i - 2b_i(s_t, a_t^{-i}) Y_i + b_i(s_t, a_t^{-i})^2 Z_i \right] - (\mathbb{E}_{\rho_\pi, a_t^{-i}} \left[X_i - \frac{Y_i^2}{Z_i} \right]) \right) \tag{60}$$

$$= \sum_i \mathbb{E}_{\rho_\pi, a_t^{-i}} \left[-2b_i(s_t, a_t^{-i}) Y_i + b_i(s_t, a_t^{-i})^2 Z_i + \frac{Y_i^2}{Z_i} \right] \tag{61}$$

$$= \sum_i \mathbb{E}_{\rho_\pi, a_t^{-i}} \left[\left(b_i(s_t, a_t^{-i}) \sqrt{Z_i} - \frac{Y_i}{\sqrt{Z_i}} \right)^2 \right] \tag{62}$$

$$= \sum_i \mathbb{E}_{\rho_\pi, a_t^{-i}} \left[Z_i \left(b_i(s_t, a_t^{-i}) - \frac{Y_i}{Z_i} \right)^2 \right] \tag{63}$$

$$= \sum_i \mathbb{E}_{\rho_\pi, a_t^{-i}} \left[Z_i \left(b_i(s_t, a_t^{-i}) - b_i^*(s_t, a_t^{-i}) \right)^2 \right] \tag{64}$$

$$= \sum_i \mathbb{E}_{\rho_\pi, a_t^{-i}} \left[\mathbb{E}_{a_t^i} \left[\nabla_\theta \log \pi_\theta(a_t^i | s_t)^\top \nabla_\theta \log \pi_\theta(a_t^i | s_t) \right] \left(b_i(s_t, a_t^{-i}) - b_i^*(s_t, a_t^{-i}) \right)^2 \right] \tag{65}$$

5.10 DERIVATION OF SUBOPTIMALITY OF THE OPTIMAL STATE-DEPENDENT BASELINE

Using the notation from Section 5.9 and working off of Equation (64), we have:

$$I_{b=b^*(s)} := \sum_i \mathbb{E}_{\rho_\pi, a_t^{-i}} \left[Z_i \left(b^*(s_t) - b_i^*(s_t, a_t^{-i}) \right)^2 \right] \tag{66}$$

$$= \sum_i \mathbb{E}_{\rho_\pi, a_t^{-i}} \left[Z_i \left(\frac{\sum_j Y_j}{\sum_j Z_j} - \frac{Y_i}{Z_i} \right)^2 \right] \tag{67}$$

$$= \sum_i \mathbb{E}_{\rho_\pi, a_t^{-i}} \left[\frac{1}{Z_i} \left(\frac{Z_i}{\sum_j Z_j} \sum_j Y_j - Y_i \right)^2 \right] \tag{68}$$

5.11 BASELINES FOR GENERAL ACTIONS

In the preceding derivations, we have assumed policy actions are conditionally independent across dimensions. In the more general case, we only assume that there are m factors a_t^1 through a_t^m which altogether forms the action a_t. Conditioned on s_t, the different factors form a certain directed acyclic graphical model (including the fully dependent case). Without loss of generality, we assume that the following factorization holds:

$$\pi_\theta(a_t|s_t) = \prod_{i=1}^m \pi_\theta(a_t^i|s_t, a_t^{f(i)}) \tag{69}$$

where $f(i)$ denotes the indices of the parents of the ith factor. Let $D(i)$ denote the indices of descendants of i in the graphical model (including i itself). In this case, we can set the ith baseline to be $b_i(s_t, a_t^{[m]\setminus D(i)})$, where $[m] = \{1, 2, \ldots, m\}$. In other words, the ith baseline can depend on all other factors which the ith factor does not influence. The overall gradient estimator is given by

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\rho_\pi, \pi} \left[\sum_{i=1}^m \nabla_\theta \log \pi_\theta(a_t^i|s_t, a_t^{f(i)}) \left(\hat{Q}(s_t, a_t) - b_i(s_t, a_t^{[m]\setminus D(i)}) \right) \right] \tag{70}$$

In the most general case without any conditional independence assumptions, we have $f(i) = \{1, 2, \ldots, i-1\}$, and $D(i) = \{i, i+1, \ldots, m\}$. The above equation reduces to

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\rho_\pi, \pi}\left[\sum_{i=1}^{m} \nabla_\theta \log \pi_\theta(a_t^i | s_t, a_t^1, \ldots, a_t^{i-1}) \left(\hat{Q}(s_t, a_t) - b_i(s_t, a_t^1, \ldots, a_t^{i-1})\right)\right] \tag{71}$$

The above analysis for optimal baselines and variance suboptimality transfers also to the case of general actions.

The applicability of our techniques to general action spaces may be of crucial importance for many application domains where the conditional independence assumption does not hold up, such as language tasks and other compositional domains. Even in continuous control tasks, such as hand manipulation, and many other tasks where it is common practice to use conditionally independent factorized policies, it is reasonable to expect training improvement from policies without a full conditionally independence structure.

Computing action-dependent baselines for general actions. The marginalization presented in Section 5.3.4 does not apply for the general action setting. Instead, m individual baselines can be trained according to the factorization, and each of them can be fitted from data collected from the previous iteration. In the general case, this means fitting m functions $b_i(s_t, a_t^1, \ldots, a_t^{i-1})$, for $i \in \{1, \ldots, m\}$. The resulting method is described in Algorithm 2.

There may also exist special cases like conditional independent actions, for which more efficient baseline constructions exist. A closely related example to the conditionally independent case is the case of block diagonal covariance structures (e.g. in multi-agent settings), where we may wish to instead learn an overall Q function and marginalize over block factors. Another interesting example to explore is sparse covariance structures.

5.12 COMPATIBILITY WITH GAE

Temporal Difference (TD) learning methods such as GAE (Schulman et al., 2016) allow us to smoothly interpolate between high-bias, low-variance estimates and low-bias, high-variance estimates of the policy gradient. These methods are based on the idea of being able to predict future returns, thereby bootstrapping the learning procedure. In particu-

Algorithm 2 Policy gradient for general factorization policies using action-dependent baselines

Require: number of iterations N, batch size B, initial policy parameters θ

Initialize baselines $b_i(s_t, a_t^{[m]\setminus D(i)}) \equiv 0$, for $i \in \{1, \ldots, m\}$ and policy π_θ

for j in $\{1, \ldots, N\}$ **do**

Collect samples: $(s_t, a_t)_{t \in \{1,\ldots,B\}}$

Compute advantages: $\hat{A}_i(s_t, a_t) := \hat{Q}(s_t, a_t) - b_i(s_t, a_t^{[m]\setminus D(i)}), \forall t$

Perform a policy update step on θ using $\hat{A}_i(s_t, a_t)$ [Equation (70)]

Update baseline functions with current batch: $b_i(s_t, a_t^{[m]\setminus D(i)})$

end for

lar, when using the value function as a baseline, we have

$$A(s_t, a_t) = \mathbb{E}\left[r_t + \gamma V(s_{t+1}) - V(s_t)\right] = \left[r_t + \gamma b(s_{t+1}) - b(s_t)\right]] \tag{72}$$

if $b(s)$ is an unbiased estimator for $V(s)$. GAE uses an exponential averaging of such temporal difference terms over a trajectory to significantly reduce the variance of the advantage at the cost of a small bias (it allows us to pick where we want to be on the bias-variance curve). Similarly, if we use $b_i(s_t, a_t^{-i})$ as an unbiased estimator for $\mathbb{E}_{a_t^i}[\hat{Q}(s_t, a_t)]$, we have:

$$\mathbb{E}_{\pi,\mathcal{M}}\left[r_t + \gamma b_i(s_{t+1}, a_{t+1}^{-i}) - b_i(s_t, a_t^{-i})\right] = Q(s_t, a_t) - \mathbb{E}_{a_t^i}[\hat{Q}(s_t, a_t)] = A_i(s_t, a_t) \tag{73}$$

Thus, the temporal difference error with the action dependent baselines is an unbiased estimator for the advantage function as well. This allows us to use the GAE procedure to further reduce variance at the cost of some bias.

The following study shows that action-dependent baselines are consistent with TD procedures with their temporal differences being estimates of the advantage function. Our results summarized in Figure 20 suggests that slightly biasing the gradient to reduce variance produces the best results, while high-bias estimates perform poorly. Prior work with baselines that utilize global information (Foerster et al., 2017) employ the high-bias variant. The results here suggest that there is potential to further improve upon those results by carefully studying the bias-variance trade-off.

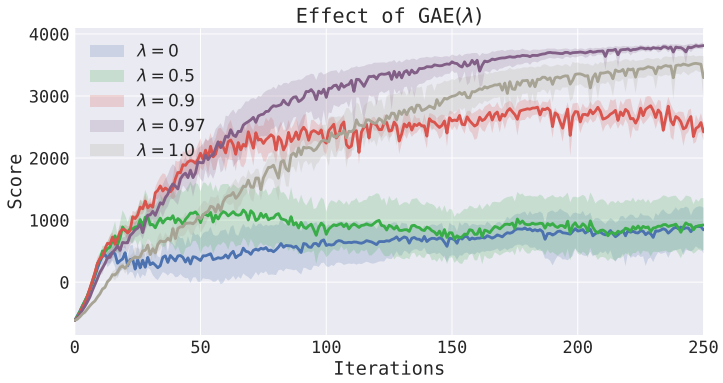


Figure 20: We study the influence of λ in GAE which allows us to trade off bias and variance as desired. High bias gradient corresponding to smaller values of λ do not make progress after a while. High variance gradient ($\lambda = 1$) has trouble learning initially. Allowing for a small bias to reduce the variance, corresponding to the intermediate $\lambda = 0.97$ produces the best overall result, consistent with the findings in Schulman et al. (2016).

5.13 HIGH-DIMENSIONAL ACTION SPACES: TRAINING CURVES

Figure 21 shows the resulting training curves for a synthetic high-dimensional target matching task, as described in Section 5.4. For higher dimensional action spaces (100 dimensions or greater), the action-dependent baseline consistently converges to the optimal solution 10% faster than the state-only baseline.

5.14 EXPERIMENT DETAILS

Parameters: Unless otherwise stated, the following parameters are used in the experiments in this work: $\gamma = 0.995, \lambda_{\text{GAE}} = 0.97$, $\text{kl}_{\text{desired}} = 0.025$.

Policies: The policies used are 2-layer fully connected networks with hidden sizes=(32, 32).

Initialization: the policy is initialized with Xavier initialization except final layer weights are scaled down (by a factor of 100x). Note that since the baseline is linear (with RBF features) and estimated with a Newton step, the initialization is inconsequential.

Per-experiment configuration: The following parameters in Tables 3 and 4 are for both state-only and action-dependent versions of the experiments. The m-DimTargetMatching

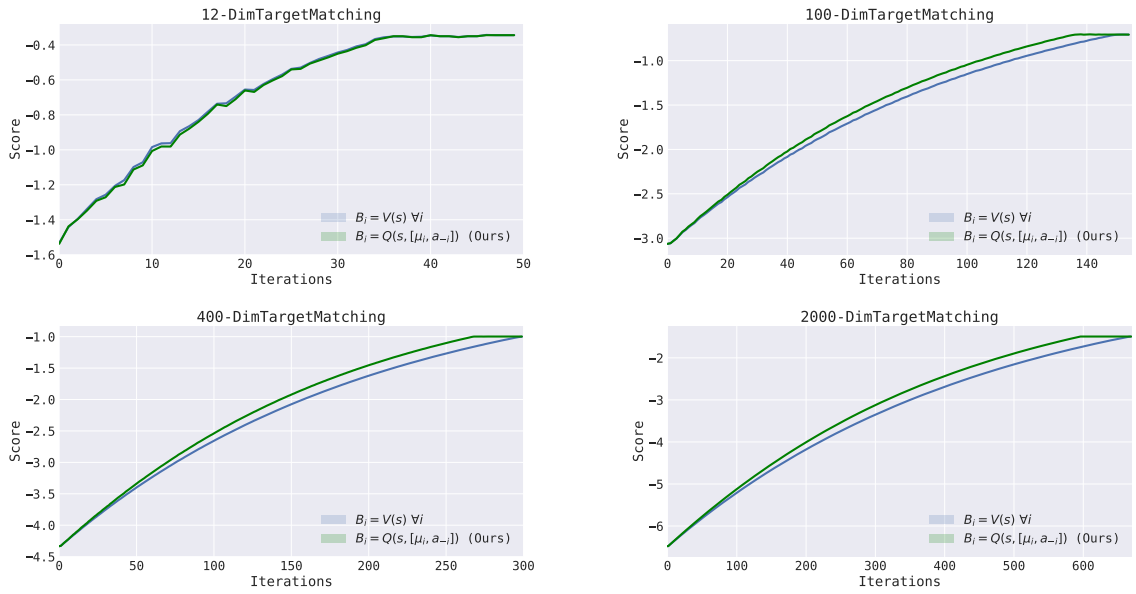


Figure 21: Shown is the learning curve for a synthetic high-dimensional target matching task (5 seeds), for 12 to 2000 dimensional action spaces. At high dimensions, the linear feature action-dependent baseline provides notable and consistent variance reduction, as compared to a linear feature state baseline. For 100, 400, and 2000 dimensions, our method converges 10% faster to the optimal solution.

experiments use a linear feature baseline. Table 5 details the dimensionality of the action space for each task.

Task	Benchmarks	Hand task	Peg Insertion
Trajectories	10	100	200
Horizon	1000	200	250
RBF features	100	250	250

Table 3: Experiment details

Task	CommunicateTarget	m-DimTargetMatching
Trajectories	300	150
Horizon	100	1
RBF features	250	N/A

Table 4: Experiment details

Task	Action dimensions
Hopper	3
HalfCheetah	6
Ant	8
Hand	30
Peg	7
CommunicateTarget	8 (4 per agent)
m-DimTargetMatching	m

Table 5: Action dimensionality of tasks

6

FLOW: A LIBRARY FOR REINFORCEMENT LEARNING AND MICROSIMULATION

> I thought cars were the dominant life-form. I was trying to introduce myself.
>
> Douglas Adams, The Hitchhiker's Guide to the Galaxy, 2005 movie

Powerful simulation systems which can capture the complexity of the interacting automated and human actors are a key component to the study of mixed autonomy systems. To this end, Flow is a new computational framework, built to support a key need triggered by the rapid growth of autonomy in ground traffic: controllers for autonomous vehicles in the presence of complex nonlinear dynamics in traffic. Leveraging recent advances in deep Reinforcement Learning (RL), Flow enables the use of RL methods such as policy gradient, derivative-free, and approximate dynamic programming methods, including those presented in Chapter 5, for traffic control. Flow additionally enables benchmarking the performance of classical (including hand-designed) controllers with learned policies (control laws). In particular, Flow integrates traffic microsimulator SUMO with deep reinforcement learning libraries, in particular rllab and Ray RLlib, and exposes a Task Designer, which enables the easy design of traffic tasks, including different networks configurations and vehicle dynamics. Flow supports advanced features, including distributed simulation, cloud support, hierarchical policies, and multi-agent environments. Flow is an advanced simulation system which is instrumental in studying emergent behaviors in mixed autonomy traffic, as introduced in Chapter 4, as well as other transportation applications which benefit from high-intensity computing.

6.1 OVERVIEW

Transportation accounts for 28% of energy consumption in the US. Workers spent on aggregate over three million driver-years commuting to their jobs (DOT, 2016), with significant impact on nation-wide congestion. Based on 2012 estimates, U.S. commuters experienced an average of 52 hours of delay per year, causing $121 billion of delay and fuel costs annually (Schrank et al., 2012). Depending on its use in traffic, automation has the potential to achieve many benefits or to exacerbate problems at the system level, with potential amelioration or worsening of various system metrics including *greenhouse gas* (GHG) emissions, *vehicle miles traveled* (VMT), *total travel time* (TTT). Estimates project that 2% of fuel consumption today is wasted due to congestion, a figure that rises to 4.2% in 2050 (Wadud et al., 2016). As such, the potential efficiency improvement provided by autonomous vehicles is 2-4% of total fuel consumption due to the alleviation of congestion alone.

Recent field operational tests by Stern et al. (2017) demonstrated a reduction in fuel consumption of 40% by the insertion of an autonomous vehicle in ring traffic to dampen the famous ring instabilities displayed by Sugiyama et al. (2008) in their seminal experiment. These tests are motivations for the present work: it demonstrates the power of automation and its potential impact on complex traffic phenomena such as *stop-and-go* waves (Garavello and Piccolli, 2006). These results are part of a broader core set of robotics challenges concerning the deployment of multi-agent automation systems, such as fleets of self-driving cars as seen in Pavone et al., 2012 and in Chapter 4 (an early version was published in Wu et al., 2017d), coordinated traffic lights (Xie et al., 2012; Belletti et al., 2018), or other coordinated infrastructure. Robotics has already demonstrated tremendous potential in improving transportation systems through autonomous vehicles research; highly related problems include localization (Sukkarieh et al., 1999; Dissanayake et al., 2001; Y. Cui and Ge, 2003), path planning (Shiller and Gwo, 1991; Bopardikar et al., 2015), collision avoidance (Minguez and Montano, 2009), and perception (Kanatani and Watanabe, 1990) problems. Considerable progress has also been made in recent decades in vehicle automation, including anti-lock braking systems (ABS), adaptive cruise control (ACC), lane keeping, lane changing, parking, overtaking, etc. (S. Drakunov et al., 1995; Van Arem et al., 2006; Lee et al., 2014; Son et al., 2015; Lefevre et al., 2014; Hatipoglu et al., 2003; Corporation, 1934; Paromtchik and Laugier, 1996; Milans et al., 2012), which also have great potential to improve energy efficiency and safety in traffic (reviewed in Section 4.9).

Down the road, the emergence of *automated districts*, i.e. districts where all vehicles

are automated and operate efficiently with collaborative path-planning, might push this paradigm to next generation mobility (Meyer and S. Shaheen, 2017). Fleets of autonomous vehicles have recently been explored in the context of shared-mobility systems, such as autonomous mobility-on-demand systems, which abstracts out the low-level vehicle dynamics and considers a queuing theoretic model. Low-level vehicle dynamics, however, are of crucial importance, as exhibited by Sadigh et al. (2016) and because many traffic phenomena, which affect energy consumption, safety, and travel time are exhibited at the level of low-level dynamics (Sugiyama et al., 2008; Lee et al., 2016; Rios-Torres and Malikopoulos, 2017a; Rios-Torres and Malikopoulos, 2017b). In some settings, model-based controllers enable analytical solutions, or tractable algorithmic solutions. However, often, due to the nonlinearity of the models, numerous guarantees are lost in the process of developing controllers (i.e. optimality, run-time, complexity, approximation ratio, etc.). For example, while the ring setting enables elegant controllers to work in practice, the extension of these results (both theoretical and experimental) to arbitrary settings (network topologies, number of lanes, heterogeneity of the fleet, etc.) is challenging.

Deep *reinforcement learning* (RL), which is the main enabler in our framework, is a powerful tool for control and has already had demonstrated success in complex but data-rich problem settings such as Atari games (Mnih et al., 2013), 3D locomotion and manipulation (Schulman et al., 2016; Schulman et al., 2015; Heess et al., 2015), chess (Lai, 2015), among others. RL testbeds exist for different problem domains, such as the *Arcade Learning Environment* (ALE) for Atari games (Bellemare et al., 2013), DeepMind Lab for a first-person 3D game (Beattie et al., 2016), OpenAI gym for a variety of control problems (Brockman et al., 2016), FAIR TorchCraft for Starcraft: Brood War (Synnaeve et al., 2016), MuJoCo for multi-joint dynamics with Contact (Todorov et al., 2012), TORCS for a car racing game (Wymann et al., 2000), among others. DeepMind and Blizzard will collaborate to release the Starcraft II AI research environment (Vinyals, 2016). Each of these RL testbeds enables the study of control through RL of a specific problem domain by leveraging of the data-rich setting of simulation. One of the primary goals of this chapter is to present a similarly suitable RL testbed for traffic dynamics by making use of an existing traffic simulator.

These recent advances in deep RL provide a promising alternative to model-based controller design, which the present chapter explores. One key step in the development of such paradigms is the ability to provide high fidelity microsimulations of traffic that can encompass accurate vehicle dynamics to simulate the action of these new RL-generated control policies, a pre-requisite to field experimental tests. This is precisely one of the

aims of the present chapter. RL promises an approach to design controllers using black box machine learning systems. However, note that there are a number of challenges to consider. It still requires the physical vehicle response to be incorporated in the simulation to learn controllers that match physical vehicle dynamics. Additional considerations extends beyond the vehicle for which the controller is to be designed. For example, although vehicle velocity is intuitive as a control variable, it is important to keep in mind that other variables, such as actuator torques, are those actually controlled; another example is that the input may consist of data from cameras, LIDAR, or radar. The conversion between the variables may not be direct and may require the design of additional controllers, the performance of which would also have to be considered.

In the present chapter, we propose the first computational framework and architecture to integrate deep RL and traffic microsimulation, thereby enabling the systematic study of autonomous vehicles in complex traffic settings, including mixed autonomy and fully autonomous settings. Our framework permits both RL and classical control techniques to be applied to microsimulations. As classical control is a primary approach for studying traffic dynamics, supporting benchmarking with such methods is crucial for measuring progress of learned controllers. As an illustration, the present chapter provides a benchmark of the relative performance of learned and explicit controllers (Stern et al., 2017) for the mixed autonomy ring road setting. The computational framework encompasses model-free reinforcement learning approaches, which complement model-based methods such as model-based reinforcement learning, dynamic programming, optimal control, and hand-designed controllers; these methods dramatically range in complexity, sometimes exhibiting prohibitive computational costs. Our initial case study investigates microscopic longitudinal dynamics (forwards-backwards) and lateral dynamics (left-right) of vehicles (Brackstone and McDonald, 1999; Zheng, 2014). We study a variety of network configurations, and our proposed framework largely extends to other reinforcement learning methods and other dynamics and settings, such as coordinated behaviors (Kotsialos et al., 1999), other sophisticated behavior models, and more complex network configurations.

The contribution of this chapter includes three components, (1) a computational framework and architecture, which provides a rich design space for traffic control problems and exposes model-free RL methods, (2) the implementation of several instantiations of RL algorithms that can solve complex control tasks, and (3) a set of use cases that illustrates the power of the building block and benchmark scenarios. Specifically, our contributions are:

- Flow, a computational framework for deep RL and control experiments for traffic

microsimulation. Flow integrates the traffic microsimulator SUMO (Behrisch et al., 2011; Krajzewicz et al., 2012) with standard deep reinforcement learning libraries rllab (Duan et al., 2016) and Ray RLLib (Moritz et al., 2017; E. Liang et al., 2017), thereby permitting the training of large-scale reinforcement learning experiments at scale on *Amazon Web Services* (AWS) *Elastic Compute Cloud* (EC2) for traffic control tasks using a variety of reinforcement learning methods. Our computational framework is open-source and available at `https://github.com/flow-project/flow`.

- An interface, provided by Flow for the design of traffic control tasks, including customized configurations of different road networks, vehicle types and vehicle dynamics, noise models, as well as other attributes provided by a standard MDP interface.
- Extensions of SUMO to support high frequency simulation and greater flexibility in controllers.
- Exposing model-free reinforcement algorithms for discounted finite (partially observed) MDPs such as policy gradient, with specific examples including *Trust Region Policy Optimization* (TRPO) (Schulman et al., 2015) and *Generalized Advantage Estimation* (GAE) (Schulman et al., 2016), to the domain of traffic control problems.
- Benchmarking of relative performance of learned and explicit controllers in rich traffic control settings. We present a benchmark on the mixed autonomy single-lane ring road network and find that a reinforcement learning agent is capable of learning policies exceeding the performance of state-of-the-art controllers. The particular case of Sugiyama instabilities is used to demonstrate the power of our tool (Sugiyama et al., 2008).
- Building block networks, including multi-lane ring road, figure-eight, merging, and intersection networks.

The rest of the chapter is organized as follows: Section 6.2 provides background on the RL framework used in the rest of the chapter. Section 6.3 describes the architecture of Flow and the processes it can handle in the three computational environments they are run (including SUMO, rllab, Ray RLlib). Section 6.4 presents the various building blocks used by SUMO for building general networks (underlying maps). Section 6.5 presents the various settings for the optimization, incl. action / observation space, reward functions and policies. This is followed by two experimental sections: in Section 6.6, in which we benchmark the performance of the RL-based algorithm to the *FollowerStopper* controller (Stern et al., 2017). Finally, Section 6.7 presents related work to place this in the broader context of traffic flow modeling, deep RL and microsimulations.

6.2 PRELIMINARIES

In this section, we define the notation used in subsequent sections. See Section 2.2 for a review of reinforcement learning.

The system described in this chapter solves tasks which conform to the standard interface of a finite-horizon discounted *Markov decision process* (MDP) (Bellman, 1957; Howard, 1964), defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma, T)$, where $\mathcal{S}$ is a (possibly infinite) set of states, $\mathcal{A}$ is a set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geqslant 0}$ is the transition probability distribution, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\rho_0 : \mathcal{S} \to \mathbb{R}_{\geqslant 0}$ is the initial state distribution, $\gamma \in (0, 1]$ is the discount factor, and T is the horizon. For partially observable tasks, which conform to the interface of a *partially observable Markov decision process* (POMDP), two more components are required, namely Ω, a set of observations, and $O : \mathcal{S} \times \Omega \to \mathbb{R}_{\geqslant 0}$, the observation probability distribution.

RL studies the problem of how agents can learn to take actions in its environment to maximize its cumulative reward. The Flow framework uses policy gradient methods (Sutton et al., 2000), a class of reinforcement learning algorithms which optimize a stochastic policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{\geqslant 0}$. These algorithms iteratively update the parameters of the policy through optimizing the expected cumulative reward using sampled data from SUMO. The policy usually consists of neural networks, and may be of several forms. Two policies used in this chapter are the *Multilayer Perceptron* (MLP) and *Gated Recurrent Unit* (GRU). MLP is a classical artificial neural network with multiple hidden layers and utilizes backpropagation to optimize its parameters (Haykin, 1994). GRUs are recurrent neural network capable of storing memory on the previous states of the system through the use of parametrized update and reset gates, which are also optimized by the policy gradient method (Chung et al., 2015). This enables GRUs to make decisions based on both current input and past inputs.

The autonomous vehicles in our system execute controllers which are parameterized policies, trained using policy gradient methods. For all experiments in this chapter, we use the TRPO (Schulman et al., 2015) policy gradient method for learning the policy, linear feature baselines as described in Duan et al. (2016), discount factor $\gamma = 0.999$, and step size 0.01. The experiment stabilizing the ring, described later, uses a hidden layer of shape (3,3) and `tanh` non-linearity. For experiments requiring memory, a GRU policy with hidden layers (5,) and `tanh` non-linearity is used.

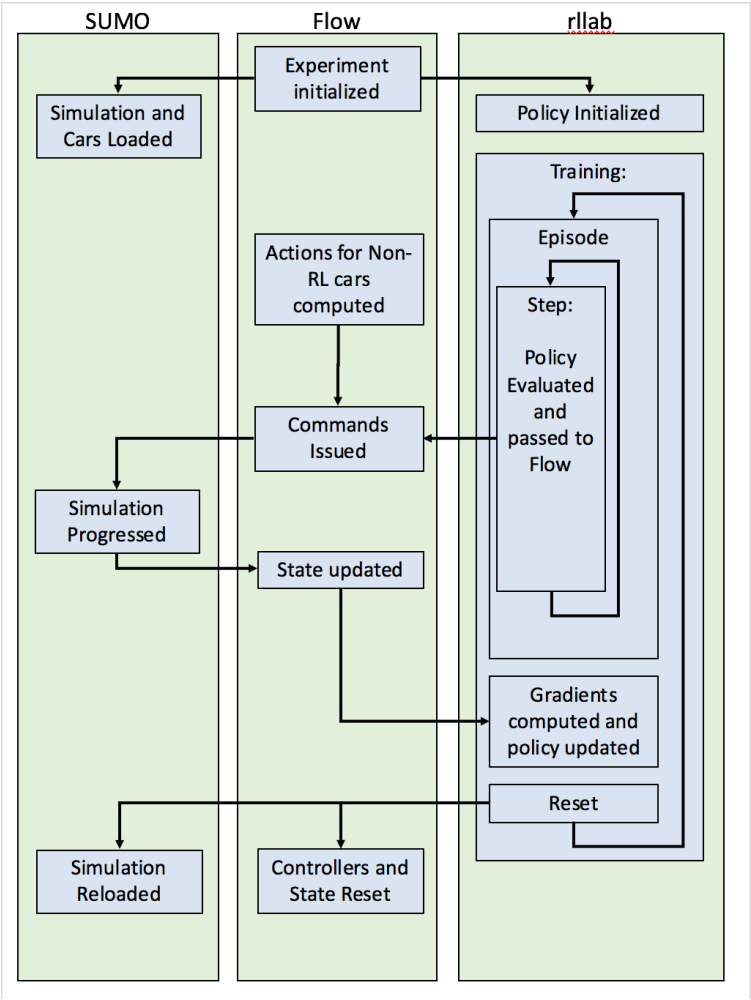


Figure 22: Flow Process Diagram. Flow interfaces SUMO via TraCI with rllab or Ray RLlib to permit the construction and simulation of traffic MDPs and for the training and evaluation of policies (control laws). After initializing the simulation in some initial configuration, rllab or Ray RLlib collects samples by advancing and observing the simulation. In each step, vehicles are provided actions through a pre-specified controller or through a policy (via rllab or Ray RLlib). These actions are then applied via TraCI and the simulation progresses. At the end of an episode, rllab or Ray RLlib issues a reset command to the environment, which returns vehicles to their initial (possibly random) position.

6.3 FLOW

Flow is created to fill the gap between modern machine learning and complex control problems in traffic. Flow is a computational framework for traffic microsimulation with

RL methods. Although the architecture is agnostic to specific machine learning and traffic software packages, we chose to integrate widely used open-source tools to promote access and extension.

The first of those open-source tools is SUMO (**S**imulation of **U**rban **MO**bility) (Krajzewicz et al., 2012). SUMO is a continuous-time and continuous-space microscopic traffic simulator. It is capable of handling large road networks and of modeling the dynamics of each vehicle in the simulation. SUMO was chosen particularly for its extensibility, as it includes an API called TraCI (**Tra**ffic **C**ontrol **I**nterface). TraCI allows users to extend existing SUMO functionality through querying and modifying the state of the simulation, at the single time-step resolution. This allows the user to easily provide intricate, custom commands that modify the simulation directly.

Secondly, we use rllab or Ray RLlib, open source framework that enables running and evaluating RL algorithms on a variety of different scenarios, from classic tasks such as cartpole balancing to more complicated tasks such as 3D humanoid locomotion (Duan et al., 2016; E. Liang et al., 2017). Flow uses rllab or Ray RLlib to facilitate the training, optimization, and application of control policies that manipulate the simulation. By modeling traffic scenarios as reinforcement learning problems, we use rllab or Ray RLlib to issue longitudinal and lateral controls to vehicles. Rllab or Ray RLlib further interfaces with OpenAI Gym, another framework for the development and evaluation of reinforcement learning algorithms. The SUMO environments built in Flow are also compatible with OpenAI Gym.

Flow encapsulates SUMO via TraCI to permit the definition and simulation of traffic MDPs for rllab or Ray RLlib to train and evaluate policies. After initializing the simulation in some initial configuration, the RL library collects samples by advancing and observing the simulation. In each step, vehicles are provided actions through a prespecified controller or through a policy. These actions are then applied via TraCI and the simulation progresses. After a specified number of timesteps (i.e. the end of a rollout) or after the simulation has terminated early (i.e. a vehicle has crashed), the RL library issues a reset command to the environment, which returns vehicles to their initial (possibly random) position. The interactions between Flow, SUMO/TraCI, and rllab or Ray RLlib are illustrated in Figure 22.

In addition to learned policies, Flow supports classical control (including hand-designed controllers and calibrated models of human dynamics) for longitudinal and lateral control. Flow also supports the car following models and lane-changing models that are provided in SUMO. These models work analogously to the policies generated by rllab or Ray RLlib, providing longitudinal and lateral controls to vehicles through ordinary dif-

ferential equations. Together, these controllers comprise the overall dynamics of mixed autonomy, fully human, or full autonomy settings.

Additionally, Flow provides various fail-safe mechanisms presented in Section 6.11, including the ones that are built into SUMO, to prevent the vehicles from crashing and the simulation from terminating early.

Flow can be used to perform both pure model-based control experiments by using only pre-specified controllers for issuing actions, or experiments with a mixture of pre-specified and learned controllers. Together, this permits the study of heterogeneous or mixed autonomy settings.

6.3.1 *Architecture of Flow*

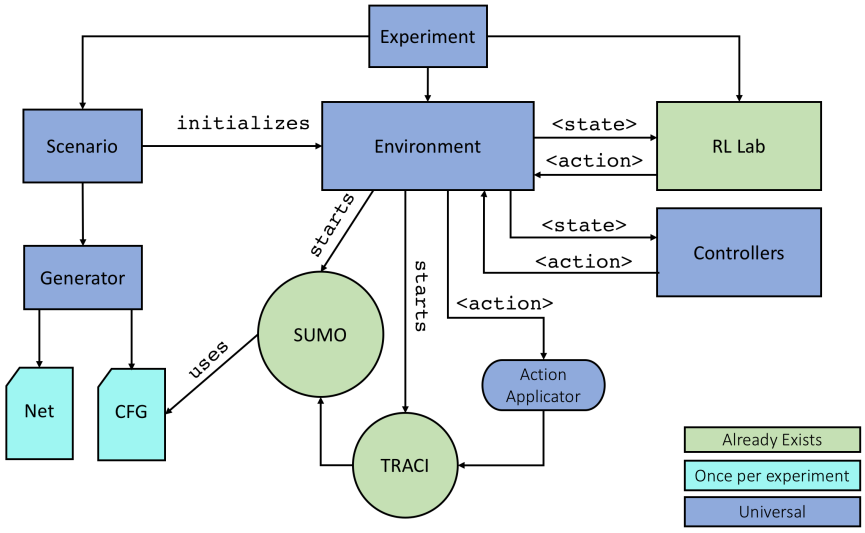


Figure 23: Flow Architecture. A Flow experiment involves a scenario and environment, interfaced with rllab or Ray RLlib and controllers. The experiment scenario runs a generator to create road networks for use in SUMO, which is started by the environment. Controllers and rllab or Ray RLlib take experiment states and return actions, which are applied through SUMO's TraCI API. (See Section 6.3.1).

An experiment using Flow requires defining two components: a scenario and an environment. These and several supporting components as well as their interactions are summarized in Figure 23.

The **scenario** for an experiment specifies network configuration in the form of network shape and attributes, for example two-lane loop road with circumference 200m, or by

importing OpenStreetMap data (see Figure 24). Based on the specifications provided, the net and configuration files needed by SUMO are generated. The user also specifies the number and types of vehicles (car following model and a lane-change controller), which will be placed in the scenario.

The **generator** is a predefined class, which allows for rapid generation of scenarios with user-defined sizes, shapes, and configurations. The experiments presented in this chapter include large loop roads generated by specifying the number of lanes and ring circumference, figure-eight networks with a crossing intersection, closed loops with merging networks, and standard intersections.

The **environment** encodes the MDP, including functions to step through the simulation, retrieve the state, sample and apply actions, compute the reward, and reset the simulation. The environment is updated at each timestep of the simulation and, importantly, stores each vehicle's state (e.g. position and velocity). Information from the environment is provided to a controller or passed to rllab or Ray RLlib to determine an action for a vehicle to apply, e.g. an acceleration. Note that the amount of information provided to either RL or to a controller can be restricted as desired, thus allowing fully observable or partially observable MDPs. This chapter studies both fully and partially observed settings.

When provided with actions to apply, Flow calls the **action applicator** which uses TraCI to enact the action on the vehicles. Actions specified as accelerations are converted into velocities, using numerical integration and based on the timestep and current state of the experiment. These velocities are then applied to vehicles using TraCI.

6.4 NETWORKS

Flow currently supports learning policies on a arbitrary (user-defined) networks. These include closed networks such as single and multi-lane ring roads, figure-eight networks, and loops with merge as well as open networks, such as intersections, and open networks with such as merge and highway networks with pre-defined in-flows of vehicles into the traffic system. See Figure 24 for various example networks supported by Flow. In each of these networks, Flow can be used to study the design or learning of controllers which optimize the system-level velocity or fuel consumption, in the presence of different types of vehicles, model noise, etc.

Single-lane ring roads: The ring road network consists of a circular lane with a specified length, inspired by the 230m track studied by Sugiyama et al. (2008). This network has been extensively studied and serves as an experimental and numerical baseline for

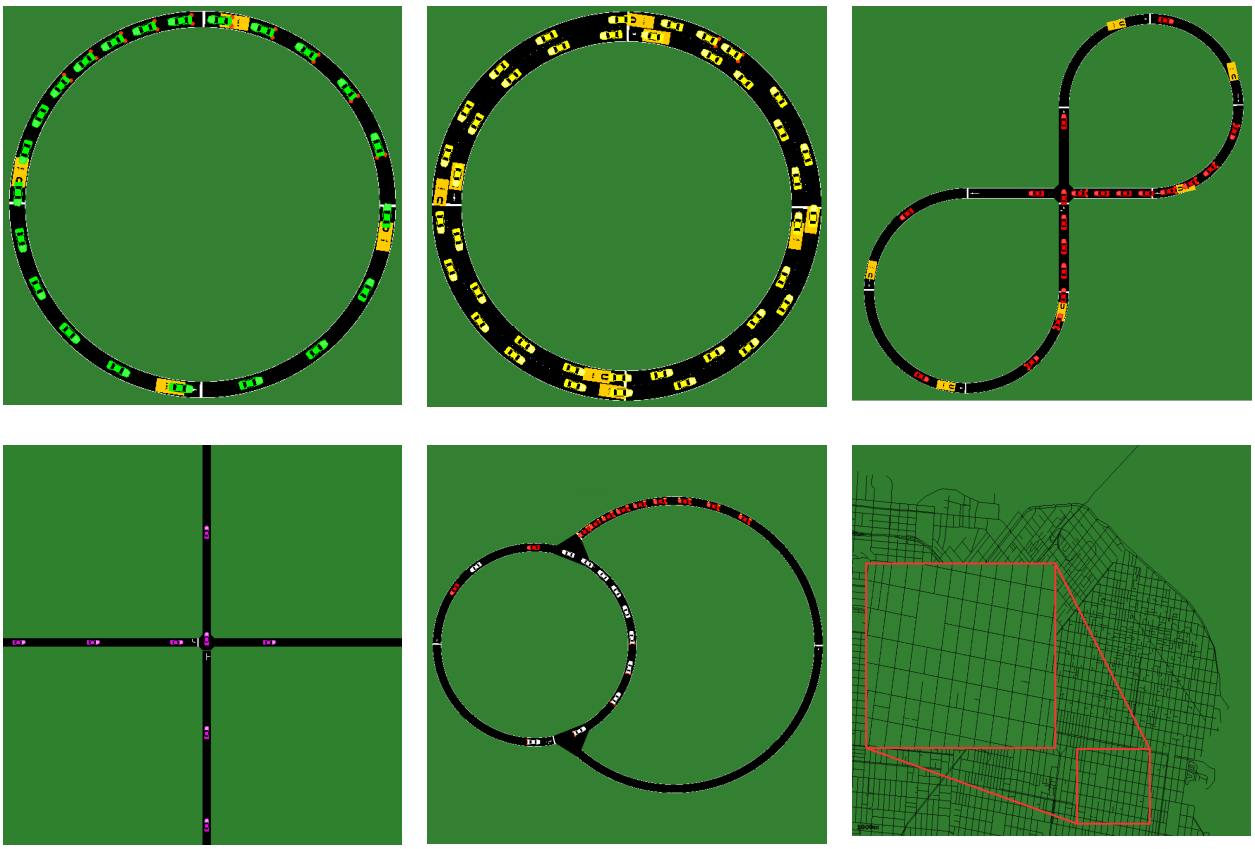


Figure 24: Various network building blocks supported by the Flow framework. **Top left**: Single-lane ring road network. **Top middle**: Multi-lane ring road network. **Top right**: Figure-eight road network. **Bottom left**: Intersection network. **Bottom Middle**: Closed loop merge network. **Bottom right**: Imported San Francisco network (currently operational for forward simulation). In Flow, maps can be generated from OSM data and visualized using SUMO.

benchmarking.

Multi-lane ring roads: Multi-lane ring roads are a natural extension to problems involving a single lane ring. The inclusion of lane-changing behavior in this setting makes studying such problems exceedingly difficult from an analytical perspective, thereby constraining most classical control techniques to the single-lane case. Many multi-lane models forgo longitudinal dynamics in order to encourage tractable analysis (Michalopoulos et al., 1984; Klar and Wegener, 1998; Sasoh and Ohara, 2002; Daganzo, 2002). Recent strides have been made in developing simple stochastic models that retain longitudinal dynamics while capturing lane-changing dynamics in a single lane setting (Wu et al.,

2017b). Modern machine learning methods, however, do not require a simplification of the dynamics for the problem to become tractable, as explored in Section 4.8.

Figure-eight network: The figure-eight network is a simple closed network with an intersection. Two ring roads, placed at opposite ends of the network, are connected by two perpendicular intersections. Vehicles that try to cross this intersection from opposite ends are constrained by a right-of-way model provided by SUMO to prevent crashes.

Loops with merge network: This network permits the study of merging behavior in closed loop networks. This network consists of two ring roads which are connected together. Vehicles in the smaller ring stay within this ring, while vehicles in the larger ring try to merge into the smaller ring and then back out to the larger ring. This typically results in congestion at the merge points.

Intersections: This network permits the study of intersection management in an open network. Vehicles arrive in the control zone of the intersection according to a Poisson distribution. At the control zone, the system speeds or slows down vehicles to either maximize average velocity or minimize experienced delay. The building block can be used to build a general schema for arbitrary maps such as the one shown in Figure 24 (bottom right).

6.5 TASK SPACE

Flow provides an interface for fine-grained traffic control task design. This section describes the options in the task design space, beyond the selection of a network configuration, as described in Section 6.4.

Action Space: When following a pre-defined route, a vehicle performs longitudinal (acceleration) and lateral (lane-changing) actions. Accordingly, for tasks with k autonomous vehicles, the action space is a set of accelerations $c \in \mathbb{R}^k$ and lane-changing decisions $d \in [-1, 1]^k$. The lane-changing values are rounded to the nearest integer (-1, 0, 1) denoting lane-change left, do not lane-change, and lane-change right, respectively; this keeps the action space representation continuous. In cases where the network only has one lane, the action space may be reduced to solely a set of accelerations.

Observation Space: The observation space may be any set of state information the user wishes to provide to the agent. This information may fully or partially describe the state of the environment. For instance, the autonomous vehicles may observe only the preceding vehicle, only nearby vehicles, or all vehicles and their corresponding position, relative position, velocity, lane information, etc.

Custom Reward Functions: The reward function can be any function of vehicle speed, position, fuel consumption, acceleration, distance elapsed, etc. Note that existing OpenAI Gym environments (atari and mujoco) come with a pre-specified reward function (Brockman et al., 2016). However, depending on the context, a researcher, control engineer, or planner may desire a different reward function or may even want to study a range of reward functions.

For all experiments presented in this chapter, we evaluate the reward on the average velocity of vehicles in the network. At times, this reward is also augmented with an added penalty to discourage accelerations or excessive lane-changes by the autonomous vehicles.

Heterogeneous Settings: Flow supports traffic settings with heterogeneous vehicle types, such as those with different controllers or parameters. Additionally, simulations can contain both learning agents (autonomous vehicles) and vehicles with pre-specified controllers or dynamics. This permits the use of Flow for mixed autonomy experiments.

Noise and Perturbations: Arbitrary vehicle-level perturbations can be specified in an experiment, choosing and randomly perturbing a vehicle by overriding its control inputs and commanding a deceleration for some duration. Gaussian noise may also be introduced to the accelerations of all human car-following models in Flow.

Vehicle Placement: Flow supports several vehicle placement methods that may be used to generate randomized starting positions. Vehicles may be distributed uniformly across the length of the network or perturbed from uniformity by some noise. In addition, vehicles may be bunched together to reduce the space they take up on the network initially, and spread out across one or multiple lanes (if the network permits it); these create configurations resembling traffic jams. Finally, the sequence in which vehicles are placed in the system may also be randomly shuffled, and thus their ordering in the state space may be randomized.

6.6 CONTROLLER DESIGN CASE STUDY: MIXED AUTONOMY RING

This section uses Flow to benchmark the relative performance of an explicit controller and the reinforcement learning approach to a given set of scenarios. The next section will show similar outcomes of our RL approaches, including examples for which there are no known explicit controllers.

The goal of this section is to demonstrate the performance of the reinforcement learning approach on the mixed autonomy single-lane ring, following the canonical single-lane ring setup of Sugiyama et al. (2008), consisting of 22 human-driven vehicles on a

230m ring track. This seminal experiment shows that such a dynamical system produces backward propagating waves, causing part of the traffic to come to a complete stop, even in the absence of typical traffic perturbations, such as those caused by lane changes and intersections. The field experiment of Stern et al. (2017) studies the case of 21 human-driven vehicles and one additional vehicle employing one of two proposed controllers, which we detail in Section 6.6.1. This setting invokes a cascade of nonlinear dynamics from n (homogeneous) agents. In this and following sections, we study the potential for machine learning techniques (RL in particular) to produce well-performing controllers, even in the presence of highly nonlinear and complex settings.

We begin by defining the experimental setup and the state-of-the-art controllers that had been designed for the mixed autonomy ring setting. We then benchmark the performance of the controller learned by Flow under the same experimental setup against the hand-designed controllers under a partially observed setting.

Experimental Scenario. In our numerical experiments, we similarly study 22 vehicles, one of which is autonomous, with ring lengths ranging between 180m and 380m, resulting in varying traffic densities. The vehicles are each 5m long and follow *Intelligent Driver Model* (IDM) dynamics with parameters specified by (Treiber and Kesting, 2013). The IDM dynamics are additionally perturbed by Gaussian acceleration noise of $\mathcal{N}(0, 0.2)$, calibrated to match measures of stochasticity to the IDM model presented by (Treiber and Kesting, 2017). We focus on the partially observed setting of observing only the velocity of the autonomous vehicle, the velocity of its preceding vehicle, and its *relative* position to the preceding vehicle. Each experiment runs for a finite time horizon, ranging from 150 to 300 seconds.

Definitions. We briefly present the important terms used in this case study. *Uniform flow* is an equilibrium state of the dynamical system (and a corresponding solution to the dynamics) where vehicles are traveling at a constant velocity. In this chapter, because the dynamical system has multiple equilibria, we use uniform flow to describe the unstable equilibrium in which the velocity is constant. We call this velocity the *equilibrium velocity* of the system. Uniform flow differentiates it from the stable equilibrium in which stop-and-go waves are formed, which does not exhibit a constant velocity.

Controllers. We compare the following controllers and observation settings for the single autonomous vehicle:

- Learned agent with GRU policy, with partial observation.
- Learned agent with MLP policy, with partial observation.
- Proportional Integral (PI) controller with saturation, with partial observation. This

controller is given in Stern et al. (2017) and is included in Section 6.6.1 for completeness.

- *FollowerStopper*, with partial observation, and desired velocity fixed at 4.15 m/s. The *FollowerStopper* controller is introduced in Stern et al. (2017) and is also detailed in Section 6.6.1. *FollowerStopper* requires an external desired velocity, so we selected the largest fixed velocity which successfully stabilizes the ring at 260m; this is further discussed in the results.
- Human driver using the *Intelligent Driver Model* (IDM), with partial observation, which is presented in more details in Chapter 6.9. This setting yields traffic jams as in Sugiyama et al. (2008) and serves as a baseline comparison.

6.6.1 *Explicit Controllers*

In this section, we describe the two state-of-the-art controllers for the mixed autonomy ring, against which we benchmark our learned policies generated using Flow.

FollowerStopper. Recent work by Stern et al. (2017) presented two control models that may be used by autonomous vehicles to attenuate the emergence of stop-and-go waves in a traffic network. The first of these models is the *FollowerStopper*. This model commands the autonomous vehicles to maintain a desired velocity U, while ensuring that the vehicle does not crash into the vehicle behind it. Following this model, the command velocity v^{cmd} of the autonomous vehicle is:

$$v^{cmd} = \begin{cases} 0 & \text{if } \Delta x \leqslant \Delta x_1 \\ v\frac{\Delta x - \Delta x_1}{\Delta x_2 - \Delta x_1} & \text{if } \Delta x_1 < \Delta x \leqslant \Delta x_2 \\ v + (U - v)\frac{\Delta x - \Delta x_2}{\Delta x_3 - \Delta x_2} & \text{if } \Delta x_2 < \Delta x \leqslant \Delta x_3 \\ U & \text{if } \Delta x_3 < \Delta x \end{cases} \tag{74}$$

where $v = \min(\max(v^{lead}, 0), U)$, v^{lead} is the speed of the leading vehicles, Δx is the headway of the autonomous vehicle, subject to boundaries defined as:

$$\Delta x_k = \Delta x_k^0 + \frac{1}{2d_k}(\Delta v_-)^2, \quad k = 1, 2, 3 \tag{75}$$

The parameters of this model can be found in Stern et al. (2017).

PI with Saturation. In addition to the *FollowerStopper* model, Stern et al. (2017) presents

a model called the *PI with Saturation Controller* that attempts to estimate the average equilibrium velocity U for vehicles on the network, and then drives at that speed. This average is computed as a temporal average from its own history: $U = \frac{1}{m}\sum_{j=1}^{m} v_j^{AV}$. The target velocity at any given time is then defined as:

$$v^{target} = U + v^{catch} \times \min\left(\max\left(\frac{\Delta x - g_l}{g_u - g_l}, 0\right), 1\right) \tag{76}$$

Finally, the command velocity for the vehicle at time $j+1$, which also ensures that the vehicle does not crash, is:

$$v_{j+1}^{cmd} = \beta_j(\alpha_j v_j^{target} + (1-\alpha_j)v_j^{lead}) + (1-\beta_j)v_j^{cmd} \tag{77}$$

The values for all parameters in the model can be found in Stern et al. (2017).

6.6.2 *Results*

Through this detailed case study of the mixed autonomy single-lane ring, we demonstrate that Flow enables the fine-grained benchmarking of classical and learned controllers. Videos and additional results are available at `https://sites.google.com/view/ieee-tro-flow`.

Performance. Figure 27 shows several key findings. This traffic density versus velocity plot shows the performance of the different learned and hand-designed controllers. First, we observe that GRU and MLP controllers (in partially observed settings) are capable of matching the uniform flow speed very closely for all trained densities, thereby effectively stabilizing traffic in all densities in the training range. The PI with Saturation controller, on the other hand, is only capable of properly performing at densities less than or equal to the density at which it was calibrated (less congested settings).

Figure 25 shows the velocity profiles for the different learned and hand-designed controllers for the 260m ring and additionally includes the *FollowerStopper* controller. We observe that although all controllers are able to stabilize the system, the GRU controller allows the system to reach the uniform flow equilibrium velocity most quickly. The GRU and MLP policies stabilize the system with less oscillatory behavior than the *FollowerStopper* and PI with Saturation controllers, as observed in the velocity profiles. In addition, the *FollowerStopper* controller is the least performant; the controller can only stabilize a 260m ring road to a speed of 4.15 m/s, well below the 4.82 m/s uniform flow velocity.

Finally, Figure 26 shows the space-time curves for all vehicles in the system, using a

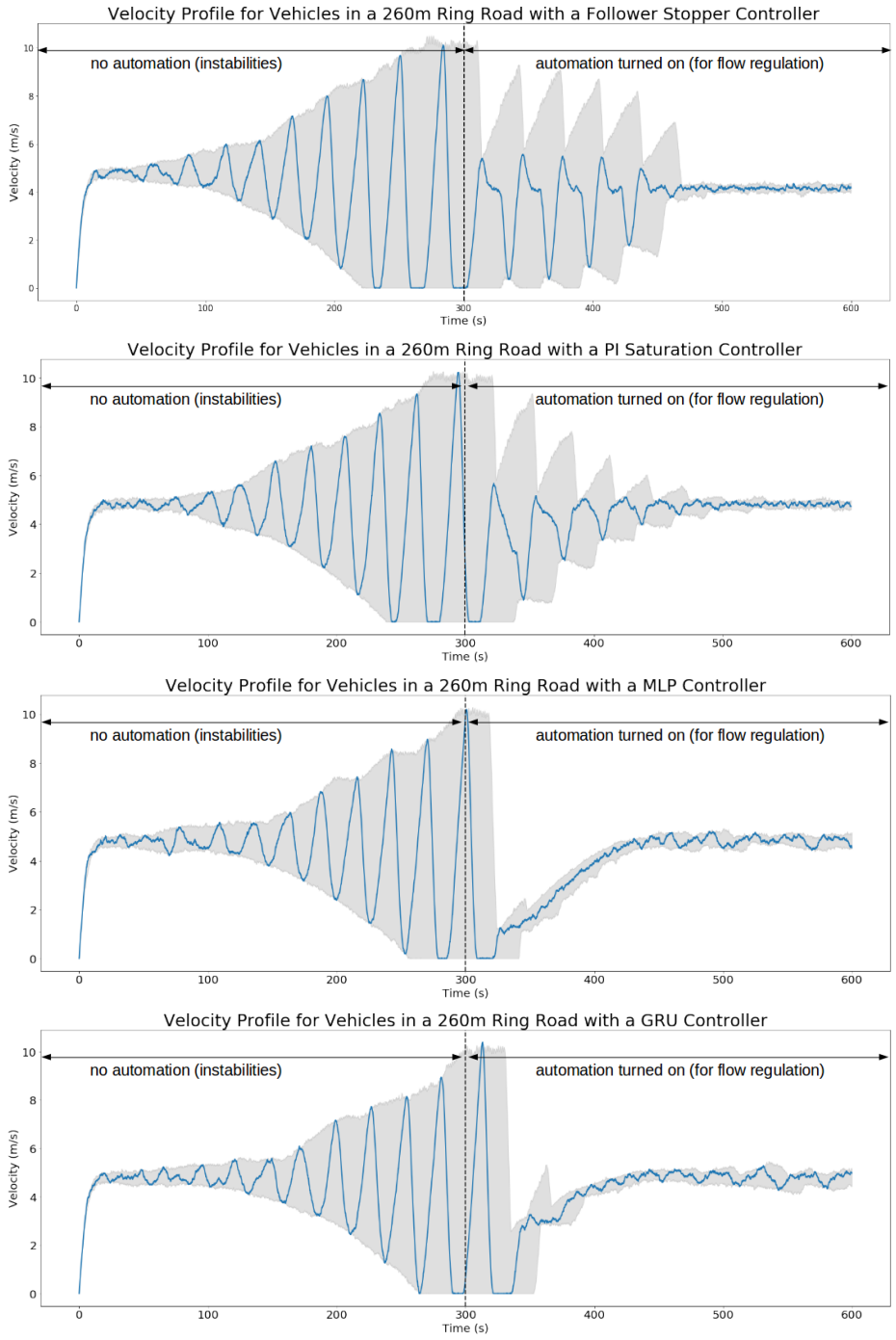


Figure 25: All experiments are run for 300 seconds with the autonomous vehicle acting as a human driver before it is activated. As we can see, an autonomous vehicle trained in a partially observable ring road setting with variable traffic densities is capable of stabilizing the ring in a similar fashion to the *FollowerStopper* and PI with Saturation Controller. Among the four controller, the GRU controller allows the system to reach the uniform flow equilibrium velocity most quickly. In addition, the *FollowerStopper* controller is the most brittle and can only stabilize a 260m ring road to a speed of 4.15 m/s, well below the 4.82 m/s uniform flow velocity.

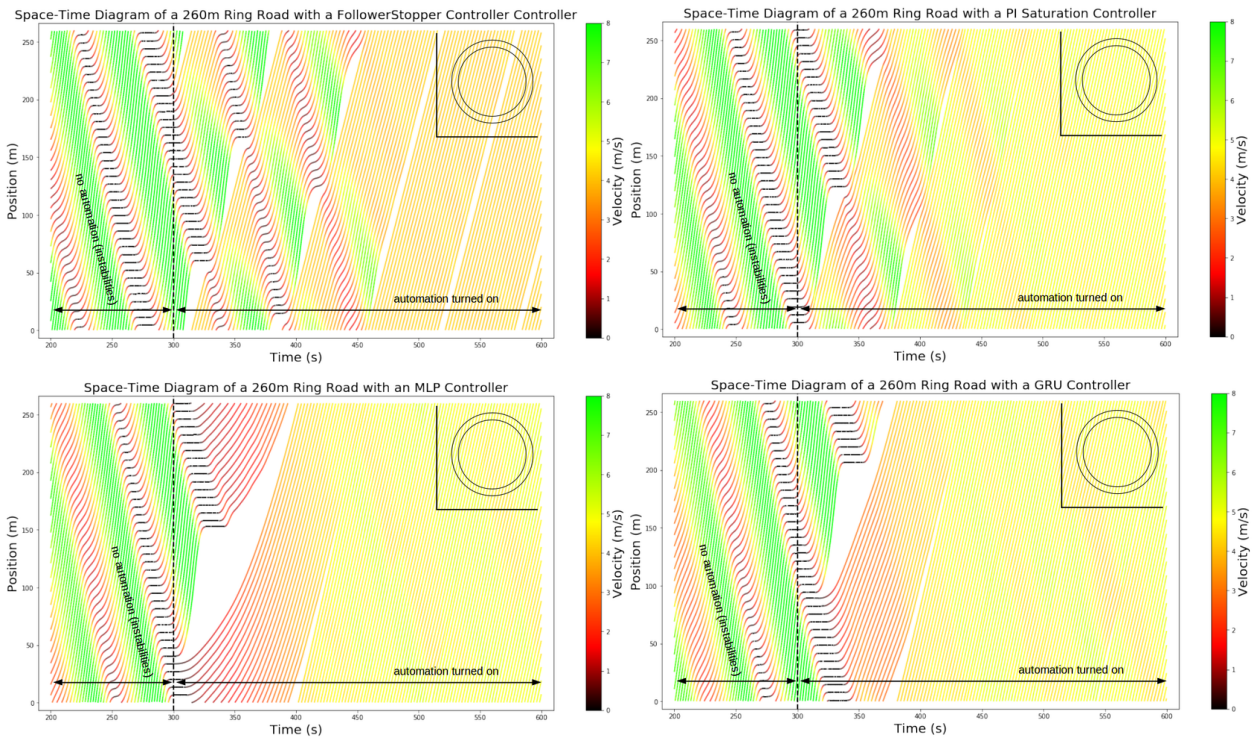


Figure 26: Prior to the activation of the single automated vehicle in the ring road network, all settings exhibit backward propagating waves resulting from stop-and-go behavior. The automated vehicle then employs different policies aimed at attenuating these waves. **Top left**: Space-time diagram for an AV employing a *FollowerStopper* Controller. **Top Right**: Space-time diagram for an AV employing a PI with Saturation Controller. **Bottom left**: Space-time diagram for an AV employing a MLP Controller. **Bottom right**: Space-time diagram for an AV employing a GRU Controller.

variety of controllers. We observe that the PI with Saturation and *FollowerStopper* controllers leave much smaller openings in the network (smaller headways) than the MLP and GRU policies. The MLP policy exhibits the largest openings, as can be seen by the large white portion of the MLP plot within Figure 26. If this were instead applied in a multi-lane ring study, then the smaller openings would have the benefit of preventing opportunistic lane changes, so this observation can lead to better reward design for more complex mixed autonomy traffic studies.

Robustness. One of the strengths of our GRU and MLP policies is that it does not rely on external calibration of parameters that is specific to a particular traffic setting, such as density.

Although the PI with Saturation controller can conceptually adjust to different densities, with its moving average filter, we experimentally found that its performance is sensitive to its parameters. Using parameters calibrated for 260m ring roads (as described in Stern et al. (2017)), the PI with Saturation controller indeed performs the best at 260m among the density range considered in this study. However, this controller's performance quickly degrades at higher density (more congested settings), dipping close to the stop-and-go equilibrium velocity.

Similarly, the *FollowerStopper* Controller suffers from the same calibration deficiencies as the PI with Saturation Controller. Additionally, the desired velocity must be provided beforehand. Interestingly and moreover, we found that this controller often fails to stabilize the system if provided too high of a desired velocity, even if it is well below the equilibrium velocity. Instead, if a lower desired velocity is first provided as an intermediate control target, then the desired velocity may then subsequently be achieved. This suggests that a simple control law such as the *FollowerStopper* cannot optimally stabilize a mixed autonomy ring, and additionally, that there is additional tuning and augmentation necessary to use the *FollowerStopper* controller.

Generalization of the Learned Control Policy. Training on different vehicle densities encourages the learning of a more robust policy. We found the policy to generalize even to densities outside of the training regime. Figure 27 shows the average velocity vehicles in the network achieve for the final 100s of simulation time; the gray regions indicate the test-time densities. Surprisingly, we found that even when training on different densities but in the *absence* of acceleration error in the human driver models, the learned policies successfully stabilized settings *with* human model noise during test time.

Discussion. This benchmark study demonstrates that deep RL can be used to learn a controller which performs better than state-of-the-art hand-designed controllers for the given setting, in terms of velocity. In particular, policy gradient methods, using the same state information provided to the hand-designed controllers and with access to samples from the overall traffic system (via a black box simulator), can learn a near-optimal controller in terms of average system-level velocity performance.

This study focuses on the partially observed setting, since it is the more realistic setting for near-term deployments. Furthermore, there are hand-designed controllers in the literature for this setting, with which we can benchmark. We would expect that the fully observed setting (with a MLP policy) would perform as well if not better than our learned policies in the partially observed setting. Since our policies already closely track the equilibrium velocity curve, we do not explore the fully observed setting.

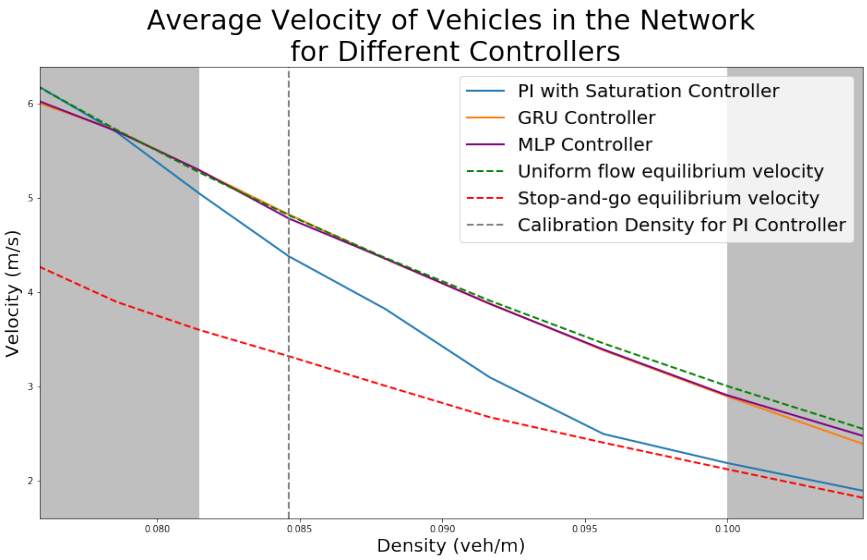


Figure 27: The performance of the MLP, GRU, and PI Saturation controllers for various densities are averaged over ten runs for each of the tested densities. The GRU and MLP controllers are capable of matching the uniform flow speed very closely for all trained densities. The PI with Saturation controller, on the other hand, is only capable of properly performing at densities less than or equal to the density at which it was calibrated. Remarkably, the GRU and MLP controllers are also reliable enough to stabilize the system at velocities close to the uniform flow equilibrium even for densities outside the training set.

6.7 RELATED WORK

Traffic Dynamics. Modeling and analysis of traffic dynamics is notoriously complex and yet is historically considered a prerequisite for traffic control (Treiber and Kesting, 2013; Papageorgiou et al., 2003). Researchers classically trade away the complexity of the model (and thus the realism of the model) in favor of the tractability of analysis, using high level abstraction with the goal of designing optimal controllers or other controllers with desirable properties, such as safety or comfort (P. A. Ioannou and Chien, 1993; Vahidi and Eskandarian, 2003; Technical Committee ISO/TC 204, Intelligent transport systems, 2010; Martin et al., 2013). Consequently, results in traffic control can largely be classified as small-scale simulation-based numerical analysis (C.-Y. Liang and H. Peng, 2000; Bose and P. A. Ioannou, 2003; P. A. Ioannou and Stefanovic, 2005; Kamal et al., 2014, e.g.) or theoretical analysis on simple settings such as assuming non-oscillatory responses (Swaroop, 1997, e.g.) or focusing on a single-lane ring road (Orosz et al., 2010; Orosz et al., 2011; I. G. Jin and Orosz, 2014; Horn, 2013; L. Wang et al., 2016; Wu et al.,

2017a, e.g.).

In particular, with the advent of autonomous vehicles, new frameworks and techniques are urgently needed to establish a foundation for studying the control and the effects of autonomous vehicles, thereby preparing the world for their adoption. Modern reinforcement learning techniques indicate promise towards the goal of obtaining controllers with desirable (though perhaps not optimal) properties while simultaneously studying complex settings.

Deep RL and Traffic. Several recent studies incorporated ideas from deep learning in traffic optimization. Deep RL has been used for traffic prediction (Lv et al., 2015; Polson and Sokolov, 2017) and control (L. Li et al., 2016; Belletti et al., 2018). A deep RL architecture was used in Polson and Sokolov (2017) to predict traffic flows, demonstrating success even during special events with nonlinear features; to learn features to represent states involving both space and time, Lv et al. (2015) additionally used hierarchical autoencoding in the traffic flow prediction problem. Deep Q Networks (DQN) was employed for learning traffic signal timings in L. Li et al. (2016). A multi-agent deep RL algorithm was introduced in Belletti et al. (2018) to learn a policy for ramp metering. For additional uses of deep learning in traffic, we refer the reader to Karlaftis and Vlahogianni (2011), which presents an overview comparing non-neural statistical methods versus neural networks in transportation research. These recent results demonstrate that deep learning and deep RL are a promising approach to traffic problems. This chapter aims to bridge the gap between deep RL and traffic control problems by providing a computational framework for learning well-performing controllers; a preliminary prototype of our architecture is published in Wu et al. (2017e).

Traffic Simulation. Traffic microsimulators include Quadstone Paramics (Cameron and Duncan, 1996), VISSIM (Fellendorf, 1994; Fellendorf and Vortisch, 2010), AIMSUN (Casas et al., 2010), MATSIM (Horni and (eds.), 2016), POLARIS (Auld et al., 2016), and SUMO (Krajzewicz et al., 2012). The first three are closed-source commercial software, whereas the latter three are open source commercial-grade software. Of the three open source solutions, the first two are designed for agent-based modeling (B. Chen and Cheng, 2010) and the third is designed for traffic microsimulation. Each of these tools are capable of large-scale traffic simulation and can handle a variety of policies and control strategies. Each tool offers an *Application Programming Interface* (API) which permits overriding or extending the default models such as car following, lane changing, route choice, etc. Each of these simulators are widely used in the research community. These tools differ in their precise offerings and features, such as visualization tools, supported models, and simulation speed. Because most studies focus their study on a single simulator, a

comprehensive comparison of these tools is unfortunately lacking.

In the present work, we choose to integrate SUMO, an open-source, extensible, microscopic simulator that can simulate large road networks. SUMO discretizes time and progresses the simulation for a user-specified timestep; furthermore, because SUMO is microscopic, individual vehicles are controlled by car following models—functions of the vehicle's headway, velocity and the velocity of the preceding vehicle. The acceleration provided by the car following model is applied as a change of velocity over the course of the next timestep. SUMO's car following models include IDM, IDMM, and Wiedermann.

SUMO has several current issues which limit its suitability for RL. First, all SUMO built-in car following models are configured with a minimal time headway, τ, that is used to ensure safety (Erdmann, 2016), and do not support time delays. Second, SUMO's car following models are calibrated for a simulation timestep of 1.0 seconds, and their behavior for smaller timesteps is known to produce unnatural behaviors (SUMO Team, 2016), whereas we would like to simulate at 10-100ms timesteps, consistent with human physiological reaction times. Finally, there does not yet exist an interface between commercial-grade microsimulation and RL libraries, although studies have demonstrated the potential of integrating microsimulation and powerful optimization methods such as genetic algorithms (Sanchez-Medina et al., 2010). Similarly, because reinforcement learning is fully data-driven and thus its success relies on the realism of the model/simulator, we require traffic models that capture realistic fine-grained dynamics, including operating at a higher granularity (smaller simulation step), with a different model of time delays, with acceleration-based control, etc.

Our work aims to address each of these limitations. Flow extends SUMO to permit rich custom controllers which may operate at smaller simulation steps and with time delays. These richer control actions allow Flow to support a larger class of controllers, thus permitting a more realistic and suitable testbed for reinforcement learning in traffic dynamics. SUMO also includes a Python API called *TRAffic Control Interface* (TraCI), from which the user can retrieve information about the vehicles' current states and issue precise commands to set the vehicles' velocities, positions, and lanes. This API allows us to interface SUMO with RL libraries, read out state information, issue actions, define our own car following models, etc.

6.8 CHAPTER SUMMARY

Flow is a computational framework built on open source tools; it enables learning policies for autonomous vehicles in complex traffic settings involving nonlinear vehicle dynamics and arbitrary network configurations. This chapter demonstrates its capabilities and provides several concrete examples and a case study which effectively benchmarks learned policies against established control results. The expansion and combination of benchmark networks to additional network types, including arbitrary grid networks, more complex intersections, and importing arbitrary map networks, is the subject of ongoing work, and will be operational soon (it is already functional for simulation). Supporting more advanced RL algorithms and non-RL methods in Flow will be important for benchmarking and overcoming the poor sample efficiency of current RL methods, due to the presence of combinatorial structures such as graphs (road networks) (Dai et al., 2017) and multiple agents (Lowe et al., 2017). Interesting and promising future directions include extending Flow to support additional features, such as evaluating safety (in addition to efficiency), using Flow as a tool to design and deploy specific controllers (which can be interpreted or for which properties such as optimality can be proven), and using it to inform public policy in preparation for the increased adoption of autonomous vehicles. Finally, as seen in many traffic management project led by State agencies, microsimulation tools are the last step before field implementation, which we hope to see for this work as well.

6.9 CLASSICAL CONTROLLERS

Flow enables the study of automated vehicles in complex traffic settings, which includes interactions with human-driven vehicles. This section presents several classical controllers that are commonly used in the literature for prescribing human and automated driving behavior (e.g. including ACC). We provide these implementations in Flow for several reasons: 1) to support studies of heterogeneous traffic control with mixtures of vehicles of different types, and 2) to enable numerical experiments which are backwards compatible with the literature.

6.9.1 *Longitudinal controllers*

Longitudinal Controllers: Longitudinal dynamics are usually defined by car following models (Orosz et al., 2010). Standard *car following models* (CFMs) are of the form:

$$a_i = \dot{v}_i = f(h_i, \dot{h}_i, v_i), \tag{78}$$

where the acceleration a_i of vehicle i is some typically nonlinear function of $h_i, \dot{h}_i, v_i$, which are respectively the headway, relative velocity, and velocity for vehicle i. A general model may include time delays from the input signals $h_i, \dot{h}_i, v_i$ to the resulting output acceleration a_i. Example CFMs include the *Intelligent Driver Model* (IDM) (Treiber et al., 2000) and the *Optimal Velocity Model* (OVM) (Bando et al., 1994; Bando et al., 1995). Our presented system implements several known CFMs and provides an easy way to implement custom CFMs.

Custom longitudinal controllers can be implemented in Flow using methods similar to the general car following model in Equation 78, in which a vehicle's acceleration is some function of its speed, headway, and relative velocity. Car following models are not limited to those inputs, however; full access to the state of the environment at each timestep is provided to controllers.

Out of the box, Flow supports a variety of car following models, including SUMO default models and custom models not provided by SUMO. Each model specifies the acceleration for a vehicle at a given time, which is commanded to that vehicle for the next time-step using TraCI.

Controllers with arbitrary time delays between perception and action are supported in Flow. Delays are implemented by storing control actions in a queue. For delayed controllers, a new action is computed using the state at each timestep and enqueued, and an action corresponding to some previous state is dequeued and commanded. Descriptions of supported car-following models follow below.

6.9.1.1 *Second-order linear model*

The first, and simplest, car following model implemented is the forward-looking car following model specified in (Orosz et al., 2010). The model specifies the acceleration of vehicle i as a function of a vehicle's current position and velocity, as well as the position and velocity of the vehicle ahead. Thus: $\dot{v}_i = k_d(d_i - d_{des}) + k_v(v_{i-1} - v_i) + k_c(v_i - v_{des})$ where v_i, x_i are the velocity and position of the i-th vehicle, $d_i := x_{i-1} - x_i$ is the headway for the i-th vehicle, k_d, k_c, k_v are controller gains for the difference between the distance

to the leading car and the desired distance, relative velocity, and the difference between current velocity and desired velocity, respectively. In addition, d_{des}, v_{des} are the desired headways and velocities respectively.

6.9.1.2 *Intelligent Driver Model*

As described in Chapter 4 and included here for completeness, the *Intelligent Driver Model* (IDM) is a microscopic car-following model commonly used to model realistic driver behavior (Treiber et al., 2000). Using this model, the acceleration for a vehicle following IDM dynamics is defined by its bumper-to-bumper headway s (distance to preceding vehicle), velocity v, and relative velocity Δv, via the following equation:

$$a_{IDM} = \frac{dv}{dt} = a\left[1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s}\right)^2\right] \tag{79}$$

where s^* is the desired headway of the vehicle, denoted by:

$$s^*(v, \Delta v) = s_0 + \max\left(0, vT + \frac{v\Delta v}{2\sqrt{ab}}\right) \tag{80}$$

where $s_0, v_0, T, \delta, a, b$ are given parameters. Typical values for these parameters can be found in (Treiber et al., 2000) or in Section 4.2.

6.9.1.3 *Optimal Velocity Model (OVM)*

Another car following model implemented in Flow is the optimal velocity model from I. G. Jin and Orosz (2014). A variety of optimal velocity functions exist for use in specifying car following models (Batista and Twrdy, 2010; Treiber and Kesting, 2013); I. G. Jin and Orosz (2014) uses a cosine-based function to define optimal velocity $V(h)$ as a function of headway:

$$V(h) = \begin{cases} 0 & h \leqslant h_{st} \\ \frac{v_{max}}{2}(1 - \cos(\pi \frac{h - h_{st}}{h_{go}} - h_{st})) & h_{st} < h < h_{go} \\ v_{max} & h \geqslant h_{go} \end{cases}$$

The values h_{st}, h_{go} correspond to headway thresholds for choosing an optimal velocity, so that for headways below h_{st}, the optimal velocity is 0, and for headways above h_{go},

the optimal velocity is some maximum velocity v_{max}. The optimal velocity transitions using a cosine function for headways between h_{st} and h_{go}. $V(h)$ is used in the control law for the acceleration of the i-th vehicle, where $\dot{v}_i = \alpha[V(h_i) - v_i] + \beta[v_{i-1} - v_i]$ at each timestep. This controller can also be implemented with delay to simulate perception and reaction times for human drivers, in which case $\dot{v}_i(t)$ would be a function of states $h_i(t-\tau), v_i(t-\tau), v_{i-1}(t-\tau)$.

6.9.1.4 *Bilateral Control Model (BCM)*

The bilateral controller presented by Horn (2013) and L. Wang et al. (2016) considers not only the relation of a subject vehicle to the vehicle ahead but also to the vehicle behind it. In their controller, the subject vehicle's acceleration depends on the distance and velocity difference to both the vehicle ahead and behind, with

$$\dot{v}_i = k_d h_i + k_v((v_{i-1} - v_i) - (v_i - v_{i+1})) + k_c(v_i - v_{des})$$

where $h_i := (x_{i-1} - x_i) - (x_i - x_{i+1})$. Horn (2013) and L. Wang et al. (2016) argue that bilateral controllers can stabilize traffic.

6.9.2 *Lateral controllers*

SUMO has lateral dynamics models dictating when and how to lane change (Erdmann, 2015); however, to extend lateral control to the RL framework, Flow permits the easy design of new and higher fidelity lane changing models. The current implementation of Flow includes a proof of concept lane-changing model in which vehicles change lanes stochastically based on speed advantage when adjacent lanes satisfy a set of constraints. Vehicles in Flow do not check to change lanes at each timestep, as that might lead to an excessive number of lane changes. Instead, at some time interval, the vehicle determines if it should lane change. SUMO's existing lane-changing models can also be used in a Flow experiment in place of custom models.

As with longitudinal controllers, custom lateral controllers can also be built in Flow. These lane-changing models have access to the full state of the environment at each time step to use as potential inputs. This allows, for example, a vehicle to identify all nearby vehicles in adjacent lanes and their speeds, and then send a lane-change command if a lane is clear and offers potentially higher speed. Due to the rich development interface available, Flow supports the integration of complex lateral controllers.

6.10 ADDITIONAL EXPERIMENTS

This section uses Flow to benchmark the controllers described in Section 6.9 in single-lane ring traffic. Note that the experiments described in this section are forward simulations only and have no learning components.

6.10.1 *OVM from uniform initial state (Figure 28a)*

The first experiment runs the Sugiyama setup from an initial state in which all 22 vehicles were spaced evenly around the ring road and start with the same velocity. Each of the vehicles was using a *Optimal Vehicle Model* (OVM) controller (see Section 6.9.1.3). The experiment begins from a stopped state, gets up to speed, and proceeds free of traffic shockwaves for its duration.

6.10.2 *OVM with a perturbation (Figure 28b)*

In this experiment, 22 OVM vehicles are run from a uniform, evenly-spaced starting state. No traffic shockwaves form until the system is perturbed 9 seconds into the experiment, once the vehicles have roughly reached their equilibrium velocities from the unperturbed setting. One vehicle is randomly chosen and an acceleration of -5 m/s^2 is applied for 1.5 seconds. The braking of that vehicle forces the vehicles behind it to slow down as well, and the system degrades into stop-and-go traffic.

6.10.3 *OVM from a nonuniform motion state (Figure 28c)*

This experiment simulates the Sugiyama setup but from a non-uniform initial configuration. Starting with the first vehicle, the subsequent position of each vehicle is drawn from a Gaussian distribution with mean equal to the length of track divided by number of vehicles and a standard deviation given by one fifth the mean. The unstable starting state also incorporates a bunching factor, in which no vehicles are placed on some segment of the track, with the length of that segment being a user-defined variable. All 22 vehicles use the OVM controller. Instability is apparent from the beginning, with traffic rapidly degrading into traffic shockwaves and failing to recover.

6.10.4 *BCM with a perturbation (Figure 29a)*

22 vehicles implementing the bilateral car following model (BCM) (see Section 6.9.1.4), are implemented in this simulation. The simulation begins from a uniform, evenly-spaced starting state. As with the experiment above, a random vehicle is perturbed at an acceleration of -5m/s^2, 9 seconds into the simulation for 1.5 seconds. Some braking results, but unlike the OVM case described above, the BCM vehicles recover from this perturbation and traffic returns to uniform motion shortly after.

6.10.5 *BCM from a nonuniform state (Figure 29b)*

Again, 22 BCM vehicles are run in this simulation, but from the same nonuniform starting state as in the nonuniform motion OVM case, in which vehicles are spaced randomly subject to a bunching factor. There is some initial instability and small traffic shockwaves, but again the BCM vehicles recover from this non-stable state and return to uniform motion.

6.10.6 *Mixed BCM/OVM from a nonuniform initial state (Figure 29c)*

Here, 11 BCM vehicles and 11 OVM vehicles begin from a randomly spaced, and bunched starting state as described above. The proportion of bilateral control vehicles proves sufficient to prevent the stop-and-go waves seen in the unstable OVM setting. Some velocity variation persists, however, unlike the full-BCM unstable setting which returns to a completely uniform motion state.

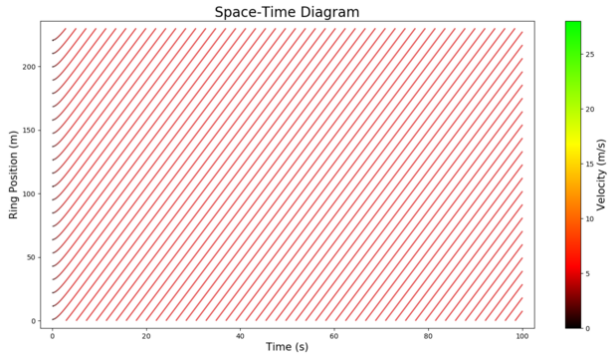
6.11 FAIL-SAFES

Flow supplements its car following models with safe driving rules that prevent the inherently unstable car following models from crashing. As SUMO experiments terminate when a collision occurs, Flow provides a fail-safe mechanism, called the *final position rule*, which runs constantly alongside other controllers. Fail-safes are passed in the action commanded by the vehicle controller, regardless of whether it is an action specified by RL or a control model. Fail-safes are a standard feature in any traffic simulator that is required to handle large perturbations and string unstable traffic. The conservativeness of the fail-safe affects the braking behavior of the traffic. In general, fail-safes operate according to the principle of maintaining a minimum safe distance from the leading vehicle

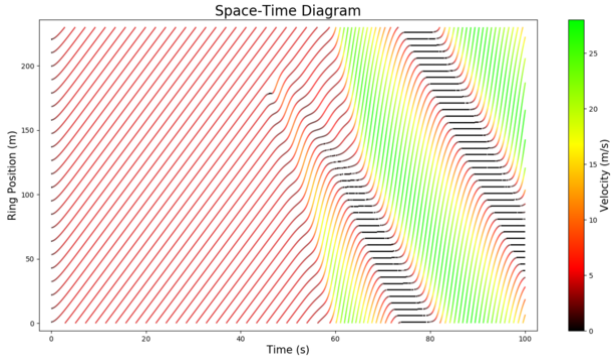
where the maximum acceleration and deceleration of the leading vehicle is stochastically generated (**dowling2004traffic**; Yeo et al., 2008).

Final Position Rule. This fail-safe aims to keep a velocity such that if the preceding vehicle suddenly starts braking with max deceleration a, then even if the following vehicle has a delay τ it can still slow down such that it comes to rest at the final position of the rear bumper of the preceding vehicle. If the preceding vehicle is initially at position $x_{i-1}(0)$, and decelerates maximally, it will come to rest at position $x_{i-1}(0) + \frac{v_{i-1}^2(0)}{2a}$. Because the fail-safe issues the maximum velocity, if the ego vehicle has delay τ, it will first travel a distance of $v_{safe}\tau$ and then begins to brake with maximum deceleration, which brings it to rest at position $x_i(0) + v_{safe} \cdot \left(\tau + \frac{v_{safe}}{2a}\right)$.

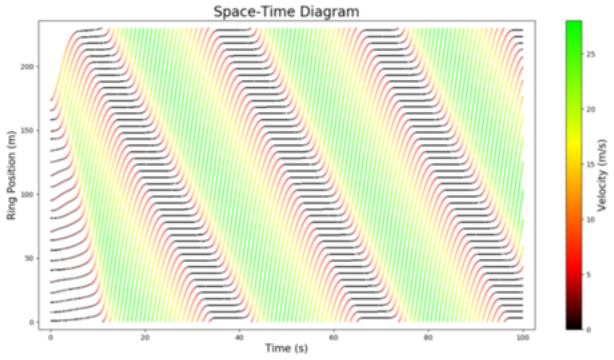
SUMO-Imposed Safety Behavior. In addition to incorporating its own safe velocity models, Flow leverages various safety features from SUMO, which may also be used to prevent longitudinal and lateral collisions. These fail-safes serve as bounds on the accelerations and lane-changes human and autonomous vehicles may perform, and may be relaxed on any set of vehicles in the network to allow for the prospect of more aggressive actions to take place.



(a) 22 OVM vehicles from uniform state on a ring road, showing an average speed of 4.87 m/s across all vehicles.

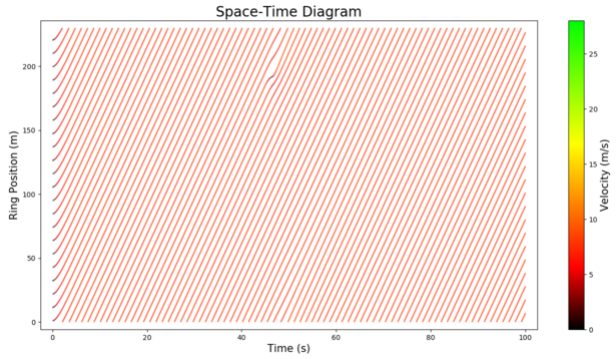


(b) 22 OVM vehicles on a ring road with a perturbation, breaking down from uniform motion into stop-and-go traffic with an average speed of 7.5 m/s.

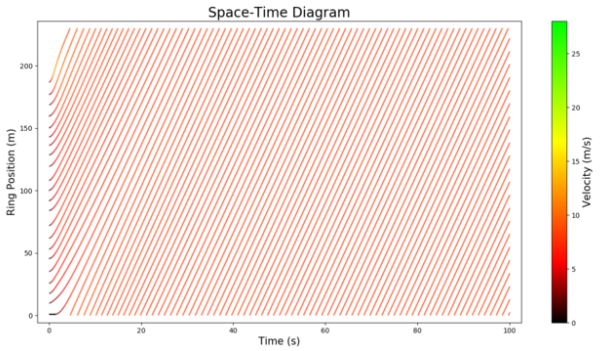


(c) 22 OVM vehicles from a nonuniform ring road initial state, showing stop-and-go traffic with an average speed of 7.8 m/s.

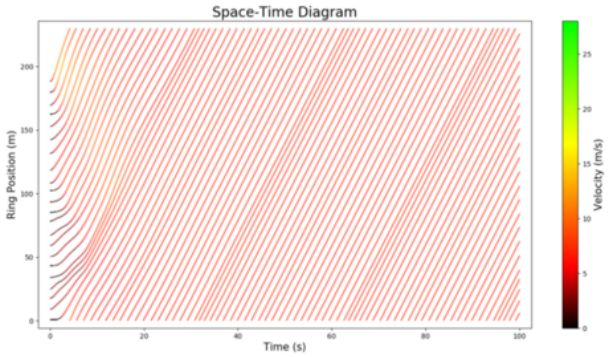
Figure 28: Spatial-temporal dynamics of numerical experiments with Optimal Velocity Model (OVM) vehicles in a ring road network with different starting conditions.



(a) 22 BCM vehicles on a ring road with a perturbation, showing an average speed of 7.9 m/s.



(b) 22 BCM vehicles from a nonuniform ring road initial state, showing an average speed of 7.9 m/s.



(c) 11 BCM and 11 OVM vehicles, from a nonuniform ring road initial state, showing an average speed of 7.1 m/s.

Figure 29: Spatial-temporal dynamics of numerical experiments showing different mixtures of Bilateral Control Model (BCM) and Optimal Velocity Model (OVM) vehicles in a ring road network.

Part II

STATE ESTIMATION

7

CELLPATH: FUSION OF CELLULAR AND TRAFFIC SENSOR DATA FOR ROUTE FLOW ESTIMATION VIA CONVEX OPTIMIZATION

> And, sure, fine, I do check my phone about every two minutes, but so do a lot of people, and it's better than smoking, that's what I say. It's the new, lung-safe cigarette.
>
> Aimee Bender,
> The Color Master: Stories (2013)

Part 2 of the thesis explores the sensing challenges and requirements to enable mixed-autonomy systems. In mixed autonomy systems, only parts of the system are automated and thus naturally observed, but information about the other parts of the system may be required in order to achieve optimal system performance. Thus being able to measure or estimate relevant quantities is critical to the performance of mixed autonomy systems. In mobility, from a control theoretic viewpoint, vehicle density determines the optimal velocity of a linear vehicle system (see Chapter 4), and thus is a critical quantity to estimate, but has classically been hindered by sparse sensing of the transportation infrastructure. Owing to advances in traffic *velocity* estimation due to mobile phones and the natural relationship between throughput, speed, and density, the problem reduces to that of estimating link-level traffic throughput, also called flow or demand (that is, vehicles/hour on a link in the network). By casting the problem into the framework of convex optimization, we determined that currently available transportation sensing infrastructure, augmented with also currently available aggregate data from cellular networks enables accurate route-level (and link-level) flow estimation. The structure of cellular network data and the large scale of urban systems motivates the design of a new algorithm for projected gradient descent with a block simplex constraint. The contributions of this

part have implications for near-term transportation management, as opposed to long-term planning such as land-use planning, and infrastructure design, as well as scalable convex optimization.

In particular, a new convex optimization framework is developed for the route flow estimation problem from the fusion of vehicle count and cellular network data. The issue of highly underdetermined link flow based methods in transportation networks is investigated, then solved using the proposed concept of *cellpaths* for cellular network data. With this data-driven approach, our proposed approach is versatile: it is compatible with other data sources, and it is model agnostic and thus compatible with user equilibrium, system-optimum, Stackelberg concepts, and other models. Using a dimensionality reduction scheme, we design a projected gradient algorithm suitable for the proposed route flow estimation problem. The algorithm solves a block isotonic regression problem in the projection step in linear time. The accuracy, computational efficiency, and versatility of the proposed approach are validated on the I-210 corridor near Los Angeles, where we achieve 90% route flow accuracy with 1033 traffic sensors and 1000 cellular towers covering a large network of highways and arterials with more than 20,000 links. In contrast to long-term land use planning applications, we demonstrate the first system to our knowledge that can produce route-level flow estimates suitable for short time horizon prediction and control applications in traffic management. Our system is open source and available for validation and extension.

7.1 OVERVIEW

While there is a wealth of literature in transportation science that is aimed at modeling, computing, and estimating the movement of people in terms of *link* flows and *origin-destination (OD)* flows, there is relatively little work focused on *route* flow estimation. The route flow estimation problem is particularly important because route flow estimates can capture phenomena in traffic behavior that link flows and OD flows (also called OD demands) cannot. For instance, route flows would enable analysis and re-routing of commuters who would be most affected by a link closure. Additionally, route flows provides a rich state estimate of the network which may be used to compute link flows, OD flows, turning ratios, etc., thereby providing backwards compatibility with past and ongoing work that builds upon those estimates.

Simultaneously accurate and efficient methods for estimating route flows are crucial for large scale urban network analysis and planning demands. However, the first step for many approaches to estimating route flow requires enumerating all feasible routes,

which is an unreasonable task for many urban road networks because it may require exponential time to compute (Ford and Fulkerson, 1962, §1.2). Classically, the set of potential routes may be extracted from the induced equilibrium in network flow models. At the cost of restrictive assumptions, *deterministic user equilibrium* (UE) (Wardrop and Whitehead, 1952) permits the modeling of unique link flows and feasible route (or path) flows without requiring full route enumeration, see Sheffi (1985, §3.3) and M. G. H. Bell and Iida (1997, §5.2). The *stochastic user equilibrium* (SUE) (probit-based (Daganzo and Sheffi, 1977; Maher and Hughes, 1997) and logit-based (Fisk, 1980; M. G. H. Bell and Iida, 1997)) addresses some of the shortcomings of the UE by modeling imperfect knowledge of the network and variation in drivers' preferences, which makes the estimation of route flows possible (M. Bell et al., 1997). However, frequent perturbations in traffic networks indicate that real-world transportation networks may not be in equilibrium (or only approximately so) (Hato et al., 1999), so we develop a data-driven framework that focuses on effectively utilizing the large amount of data available for estimation in traffic networks. Indeed, in recent years, the growing number of mobile sensors in urban areas enables the use of probe vehicles for route inference from GPS traces (Hunter et al., 2009; Rahmani and Koutsopoulos, 2013).

7.1.1 *Traffic data sources*

Traditional traffic sensing systems such as loop detectors embedded in the pavement and cameras provide accurate volume and speed estimates, but their placements are typically sparse and their information content is too coarse. Most importantly, they measure total counts of vehicles passing through a road segment without distinguishing between vehicles following different routes. In order to partially address the shortage of information on the routes followed by vehicles, other types of static sensors have been deployed on the road network: cameras that measure split ratios at different intersections (Veeraraghavan et al., 2003) and plate scanning systems (Castillo et al., 2008; Castillo et al., 2010). However, these systems require costly infrastructure and only provide highly localized traffic information. Meanwhile, given the large penetration of mobile phones among the driving population and the ubiquitous coverage of service providers in urban areas, mobile phones have become an increasingly popular source of location data for the transportation community. For example, dynamic probing by means of in-car GPS traces (Horvitz et al., 2005; Work et al., 2008; Herrera et al., 2009; Hunter et al., 2009; Ban et al., 2011; Sun and Ban, 2013) is a promising technology for congestion prediction, trajectory recovery, travel time estimation, queue length estimation, and vehicle-type classification.

However, due to the read-only nature of GPS signals (Farrell and Barth, 1999), the low penetration rate of GPS-enabled devices running a dedicated sensing application currently limits the ability to accurately estimate traffic volumes, and it is also unlikely that such data would become available to public agencies (Patire et al., 2013).

Cellular network data, in contrast to GPS traces, benefit from dedicated communication between mobile phones and cellular network base stations, and the (coarse) location data are available directly from cellular communication network operators. Cellular network infrastructures record a variety of phone to cell communication events, such as *handovers* (HO), *location updates* (LU) and *call detail records* (CDR) (Volinsky et al., 2011), and this data has already been shown to be effective in studying urban environments (Candia et al., 2008; Jiang et al., 2013; Toole et al., 2012). Since typical cellular networks in urban areas include thousands of cells, HO/LU/CDR events are dense enough to be used effectively to estimate the route choice of agents without requiring any additional infrastructure. When an agent is moving, HOs transfer ongoing calls or data sessions from one cell to another without disconnecting the session, and LUs allow a mobile device to inform the cellular network when the device move from one location (or cell) to the next. CDRs (mainly used by service providers for billing purposes) contain timestamped summaries of the cell through which each data transmission came, and therefore contain abundant mobility traces for a majority of the population. Due to the granularity of sensing, these records alone are not sufficient for recovering agent routes precisely. The spatial resolution of CDR, HO, and LU data varies with the density of antennas and is roughly proportional to the daytime population density. In the present work, we use a standard localization approach when dealing with cellular data based on Voronoi tessellation, a simple model solely based on the locations of the cell towers (Baert and Seme, 2004; Candia et al., 2008).

7.1.2 *Related work*

Several problems within traffic estimation have already benefited from incorporating data from cellular networks: OD matrix computation using cell phone location data (Caceres et al., 2007; Calabrese et al., 2011) such as CDRs (White and Wells, 2002), link flow estimation (Yadlowsky et al., 2014), and travel time and type of road congestion (Janecek et al., 2012). These studies vary in scale and assumptions, but they indicate the promise of non-pervasive sensing to provide a richer understanding of mobility. In particular, cellular network data has been used to improve the accuracy of OD matrix estimation (Caceres et al., 2007; Calabrese et al., 2011). There are many surveys on the

subject in the past decades (M. G. H. Bell and Iida, 1997; Abrahamsson, 1998; Ortuzar and Willumsen, 2001), and the accuracy of OD estimates will continue to improve. More generally, the growing variety of region-scale data sources shows promise for improved performance of transportation systems (J. Zhang et al., 2011). Additionally, convex optimization techniques have been used quite frequently by the transportation community for diverse purposes, including several of these problems. For example, the classical Wardrop equilibrium approach to the traffic assignment problem can be formulated as a convex optimization program given some typical assumptions on the link performance (or delay) functions (Sheffi, 1985). Recent works often combine convex optimization with machine learning techniques (Blandin et al., 2009; Shen and Wynter, 2012; Mardani and Giannakis, 2013).

An early study on the use of cellular network data for traffic assignment (Tettamanti et al., 2012) estimates the route choice for each user in the cellular network using a distance measure to determine the best matching route. Their small experiment (2-4 routes) performed via a macro-simulator indicates the potential of cellular network data for solving this problem. However, a recent survey on the use of wireless signals for road traffic detection (Mathew and Xavier, 2014) concludes that there is thus far no existing system that can estimate traffic densities in a practical sense, that is, in terms of scalability, coverage, cost, and reliability, thus motivating our work on estimating route flows.

7.1.3 *Contributions of this chapter*

One of the key innovations of the present work is generalizing the common notion of an OD matrix to a general form of coarse (route) flow measurements (here collected from cellular network data). As mentioned above, the problem of traffic assignment is historically highly underdetermined because the OD matrix and link flows (even when all the links are observed) contains relatively little information as compared to the number of available routes. We introduce the concept of *cellpaths*, which generalizes 2-point network flow, which we call *OD flow*, to n-point network flow, which we call *cellpath flow*. Where OD flow is characterized by two centroids (illustrated in Figure 32), cellpath flow can be characterized by n region centroids through with vehicles pass on a single trip. In this chapter, the centroids for cellpath flow correspond to cellular base stations, and the centroids for OD flows correspond to centroids of Traffic Analysis Zones (TAZ). Since our approach includes a "strict" generalization of ODs to cellpaths, the methodology presented in this chapter can be applied to a variety of traffic modeling and estimation

problems.

Now, we define our problem as follows: given a large-scale road network in the quasi-static regime, a set of OD demands, a set of admissible routes between each OD pairs, cellpath flow measurements along the network, and link flow measurements on a subset of links in the network, our goal is to develop a method to estimate the distribution of flow over the set of routes. We pose the route flow estimation problem as a mathematical program optimizing the fit to link sensor data over feasible route flow distributions, constrained to those which are consistent with measured cellpath flows in the network.

Our analysis of the structure of the constraints in the optimization program allows us to present a more efficient solution method that scales to full-sized networks. By recognizing the constraints as block-simplex constraints, we apply a standard equality constraint elimination technique (Boyd and Vandenberghe, 2004, §4.2.4) with a particular change of variables to convert the non-negativity constraints on the variables into ordering constraints. In the new space induced by the change of variables, we show that the projection on the feasible set (characterized by the ordering constraints) can be performed in linear time via bounded isotonic regression (see Tibshirani et al. (2011) for a short survey on isotonic regression), where n is the number of routes per OD pair. This is an improvement over the $O(n \log n)$ time required by the projection onto the simplex (Duchi et al., 2008b; W. Wang and Carreira-Perpinán, 2013). Then we solve our convex optimization program with a first-order projected descent algorithm. The change of variables presents two main advantages: our projection requires $O(n)$ time and the dimensionality is reduced (sometimes by a factor of $1/3$), which is critical for large-scale problems. In addition, it is worth noting that a wide variety of problems can benefit from this methodology. First, the use of algorithms that feature a projection step, e.g. projected descent methods and alternating direction methods, is very popular since they often provide a simple and efficient way to solve constrained convex optimization problem as opposed to more specialized active set methods. There is also a great deal of applications that feature simplex constraints, such as the aforementioned traffic assignment problem, many game theory settings for the computation of strategy distributions, and ℓ_1-based approach in machine learning (Duchi et al., 2008b).

Practical considerations that traffic flow in urban areas may not be in equilibrium motivate our emphasis on a data-driven approach that benefits from the sheer amount of cellular network data without relying on equilibrium-based models or other route choice models. Aiming at a real-world production system pipeline summarized in Figure 30, we demonstrate the versatility and data-driven nature of the proposed approach via validation on three datasets produced by two simulators of route assignment, where

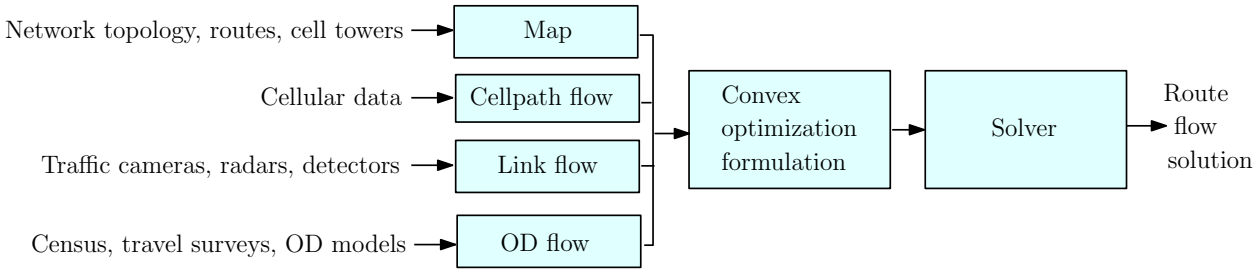


Figure 30: Proposed route flow estimation pipeline, from raw data to route flows, including: 1) a scheme for route selection and resolution cellular and road networks into a unified map; 2) a trip analysis step to filter driver cellular traces from other traces and infer cellpath flows; 3) an aggregation of the link flow obtained from static sensors over a sizable duration (*e.g.* 1 hour) suitable to address the static estimation problem; 4) a state-of-the-art OD matrix estimation method; 5) a problem formulation that handles data fusion from disparate sources; and 6) the route inference method.

the positions of the cell towers are sampled randomly on the urban networks. To assess our approach on a variety of possible route choice models that may be realized in a real-world setting, we develop a small equilibrium-based model that generates user equilibrium (UE) and system-optimal (SO) flows on the I-210 corridor near Los Angeles, CA. We also use MATSim, an agent-based transport simulator[1], on a large-scale urban road network near Los Angeles, CA (with more than 20K links and 290K routes) to showcase the performance of our methodology on large datasets. We demonstrate that our full pipeline, from the simulators to the procedures to estimate static route flows on small and large-scale urban networks, can be extended easily to incorporate other types of data such as link capacities, split ratios etc. Hence we hope that our framework will be a benchmark for many future studies of estimation problems in transportation science.[2]

We summarize the contributions of the presented work:

- We propose a convex optimization formulation for the route flow estimation problem which uses a new data fusion approach for loop detectors counts and cellular signal traces (ubiquitous among the driving population).
- We demonstrate that our formulation is also compatible with several other approaches to this problem, including equilibrium concepts, which may be used in conjunction for improved estimation.

1 MATSim is an open source project: http://www.matsim.org/publications.

2 Our full system is open source and available at https://megacell.github.io/ for validation and extension.

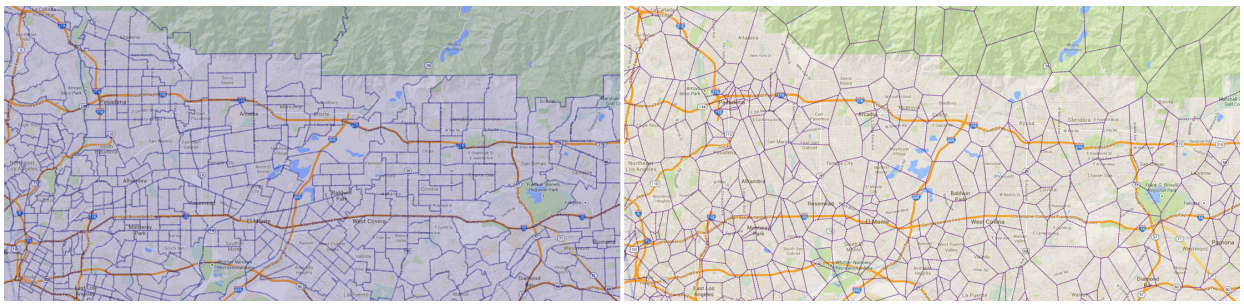


Figure 31: I-210 corridor in Los Angeles county used for the numerical work presented in Section 7.5. Left subfigure: The 700 regions are origin/destinations areas called Traffic Analysis Zones (TAZ) used for the numerical experiments. Right subfigure: Corresponding Voronoi partition of the cellular network based on 1000 cell towers. Best viewed in color.

- We introduce the concept of *cellpaths* and demonstrate its application to traffic estimation problems. We address the issues with highly underdetermined link flow based methods (which was already raised in the traffic assignment literature) by formalizing cellular data as cellpaths and incorporating them as constraints. Though we focus on the route flow estimation problem, many traffic problems may benefit from such an approach.
- Using a reduction scheme, we design an algorithm to solve the route flow estimation problem and large-scale traffic assignment problems. In the resulting formulation, the projection step can be performed in $O(n)$ via isotonic regression, an improvement over $O(n \log n)$, where n is the number of routes per OD pair.
- We present a full system pipeline from cellular network and link flow data to estimate the static route flow (and as a by-product, link flow) on a large-scale urban network. We demonstrate the first system to our knowledge that can produce route-level flow estimates suitable for short time horizon prediction and control applications in traffic management from the fusion of data from the cellular network and static sensors along roads.
- We present numerical results from different sets of small and large-scale datasets for the Greater Los Angeles Area (see Figure 31). In particular, the emphasis is placed on a data-driven approach: it is versatile to different types of underlying agent behavior models.

The remainder of the chapter is organized as follows: In Section 7.2, we present the problem setup and assumptions, then formulate our route estimation problem in the framework of convex optimization. We also provide a re-formulation necessary for the algorithmic approach described in Section 7.3. Further in Section 7.3, we develop a spe-

cialized projected gradient method to solve convex optimization programs with simplex constraints. Section 7.4 is dedicated to our experiment settings. Section 7.5 presents our numerical results. Section 7.6 concludes the chapter by placing the presented method within a general data-driven traffic estimation framework and identifying future directions.

7.2 PROBLEM FORMULATION

7.2.1 *Problem setup and assumptions*

We define the terminology used in the chapter, the notation is presented in Table 6, and the setup is illustrated in Figure 32. It is important to distinguish between four types of flows: cellpath flow, link flow, route flow, and OD flow. Our setup consists of:

- **Origins**: traffic regions each with an associated centroid, defined by a partitioning of the *road network*. Each region is both an *origin* (its centroid is a source from which trips emanate) and a *destination* (its centroid is a sink at which trips terminate). To demonstrate a possible implementation, the numerical work in this chapter uses the Traffic Analysis Zones (TAZ) (see Figure 31) as origins/destinations. We define *OD flow* to be the flow (vehicle count per time) that originates and terminates with an OD pair.
- **Cells**: regions defined by the Voronoi partition of the locations of the *cellular network base stations*; they are generally a different set of regions from the origins.
- **Cellpath**: a sequence of cells; we define *cellpath flow* to be the flow along a cellpath.
- **Link**: a segment of road in the network, and the *link flow* is the flow through a link.
- **Route**: a sequence of links from an origin to a destination. Each route also has a particular associated cellpath, as well as a particular associated OD; this insight is important for the structure of our convex optimization formulation. The *route flow* is the flow on the route.

The link-route incidence matrix A encodes the network topology (which routes $r \in \mathcal{R}$ contains which links $l \in \mathcal{L}$); the cellpath-route incidence matrix U encodes the collection of routes with the same cellpaths (which routes $r \in \mathcal{R}$ is associated to which cellpath $p \in \mathcal{P}$); and the OD-route incidence matrix T encodes which routes $r \in \mathcal{R}$ is between OD pairs $k \in \mathcal{O}^2$.[3]

3 The lowercase letters l, r, p, k written as subscripts refer to the indices associated to links, routes, cellpaths, and ODs respectively.

Table 6: Notation for route estimation problem. We have m observed links, q cellpaths, n routes.

Notation	Description
$\mathcal{O}$, $\mathcal{D}$	Set of origins/destinations $\mathcal{D} = \mathcal{O}$
$\mathcal{L}$	$\lvert\mathcal{L}\rvert = m$, links with observed flow
$\mathcal{P}$	$\lvert\mathcal{P}\rvert = q$, observed cellpaths
$\mathcal{R}$	$\lvert\mathcal{R}\rvert = n$, set of routes
$\mathbb{A}$	Set of all links in the network
$d \in \mathbb{R}_{\geqslant 0}^{\lvert\mathcal{O}\rvert^2}$	Vector of OD flows, $d = (d_k)_{k\in\mathcal{O}^2}$
$b \in \mathbb{R}_{\geqslant 0}^{\lvert\mathcal{L}\rvert}$	Observed link flow vector, $b = (b_l)_{l\in\mathcal{L}}$
$f \in \mathbb{R}_{\geqslant 0}^{\lvert\mathcal{P}\rvert}$	Cellpath flows vector, $f = (f_p)_{p\in\mathcal{P}}$
$x \in \mathbb{R}_{\geqslant 0}^{\lvert\mathcal{R}\rvert}$	Vector of route flows $x = (x_r)_{r\in\mathcal{R}}$
$v \in \mathbb{R}_{+}^{\lvert\mathbb{A}\rvert}$	Full link flow vector, $v = (v_a)_{a\in\mathbb{A}}$
$A \in \{0,1\}^{\lvert\mathcal{L}\rvert\times\lvert\mathcal{R}\rvert}$	Link-route incidence matrix
$A^{\text{full}} \in \{0,1\}^{\lvert\mathbb{A}\rvert\times\lvert\mathcal{R}\rvert}$	Full link-route incidence matrix
$U \in \{0,1\}^{\lvert\mathcal{P}\rvert\times\lvert\mathcal{R}\rvert}$	cellpath-route incidence matrix
$T \in \{0,1\}^{\lvert\mathcal{O}\rvert^2\times\lvert\mathcal{R}\rvert}$	OD-route incidence matrix
Subset $\mathcal{R}^p$	Subset of $n_p := \lvert\mathcal{R}^p\rvert$ routes with cellpath p
$\tilde{x}^p \in [0,1]^{\lvert\mathcal{R}^p\rvert}$	Ratios of flows across routes $r \in \mathcal{R}^p$
$x^p \in \mathbb{R}_{+}^{n_p}$	x_r^p is the flow of route $r \in \mathcal{R}^p$
$\mathcal{R}^k \subset \mathcal{R}$	Subset of n_k routes between OD pair k

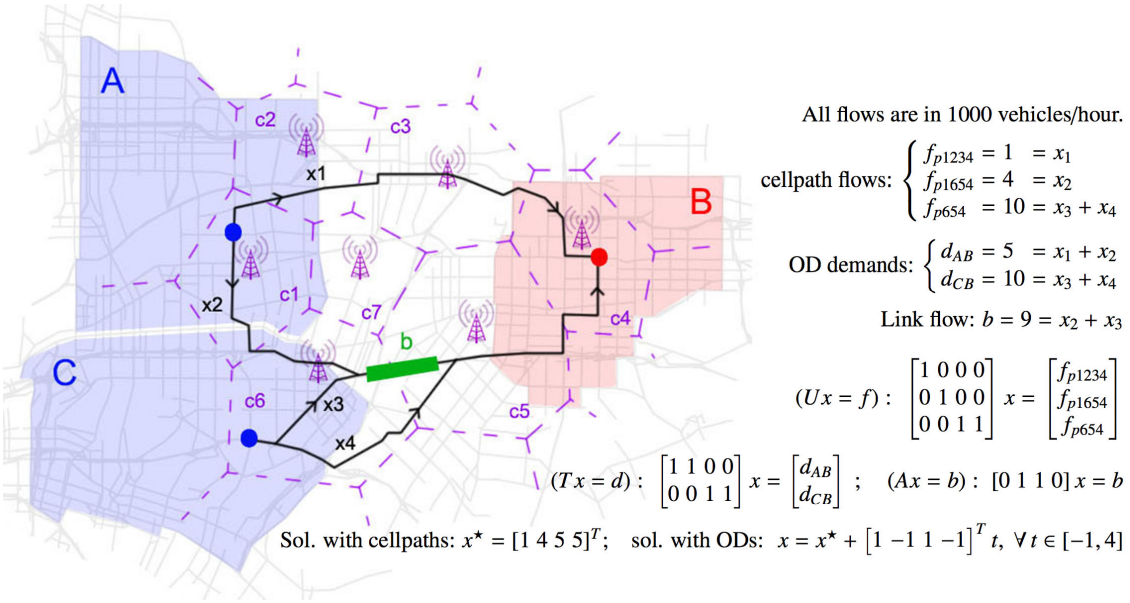


Figure 32: In this illustration of the cellular and loop data fusion, we have two origins A and C (the blue traffic regions and their centroid as blue dots) and one destination B (the red traffic region). We have routes r_1, r_2, r_3, r_4 with flows $x = (x_1, x_2, x_3, x_4)$ such that r_1, r_2 go from A to B and r_3, r_4 go from C to B. Cells $c_1, \cdots, c_7$ are shown in purple dashed regions. Since route r_1 goes through cells c_1, c_2, c_3, c_4, its associated cellpath is p_{1234}. Similarly, routes r_2, r_3, r_4 have cellpaths $p_{1654}, p_{654}, p_{654}$ respectively. Let f_{p1234}, f_{p1654}, f_{p654} be the cellpath flows (obtained from cellular network data), *i.e.* there are f_{p1234}=1000 veh/h going through c_1, c_2, c_3, c_4. Let d_{AB} and d_{CB} be the OD demands. Cellpaths p_{1234} and p_{1654} disambiguate routes between AB: $f_{p1234} = x_1$, $f_{p1654} = x_2$, contrary to the ODs: $d_{AB} = x_1 + x_2$. However, cell towers are not dense along r_3, r_4, hence $d_{CB} = f_{p654} = x_3 + x_4$. The cellpath-route incidence matrix generalizes OD matrices since we consider the sequence of intermediate regions (cells here) that intersect with trips. We also have $x_2 + x_3 = b$, with b the flow on the green link (from loop detectors). There is a unique route flow inducing flows b, f_{p1234}, f_{p1654}, f_{p654} that is $x^\star = [1\ 4\ 5\ 5]$, while there are infinitely many flows inducing b, d_{AB}, d_{CB}: $x = x^\star + [1\ -1\ 1\ -1]^T t$, $\forall\, t \in [-1, 4]$, so the problem has *one degree of freedom* and is *underdetermined* with only the OD demands as data. Best viewed in color.

$$\text{link-route: } A_{lr} = \begin{cases} 1 & \text{if } l \in r \\ 0 & \text{else} \end{cases} \tag{81}$$

$$\text{cellpath-route: } U_{pr} = \begin{cases} 1 & \text{if } r \in \mathcal{R}^p \\ 0 & \text{else} \end{cases} \tag{82}$$

$$\text{OD-route: } T_{kr} = \begin{cases} 1 & \text{if } r \in \mathcal{R}^k \\ 0 & \text{else} \end{cases} \tag{83}$$

The model assumptions are as follows:

- We consider a quasi-static setting, where traffic demands (flows) remain constant over time, and we focus on the noiseless case, with a short commentary on the noisy case in Section 7.5.
- Since enumerating all routes is not tractable, we consider the top routes between each OD pair following different criteria depending on the setting of the numerical experiment (see Section 7.4).
- We can reliably determine the cellpath flow f_p from the cellular traces along each cellpath p.
- All cellpaths $p \in \mathcal{P}$ are *contiguous*: each pair of consecutive cells in p shares a boundary.
- The set of cellpaths $\mathcal{P}$ is *well-posed*: each route $r \in \mathcal{R}$ corresponds to exactly 1 cellpath $p \in \mathcal{P}$, and we have a cellpath flow measurement f_p for each $p \in \mathcal{P}$.

7.2.2 *Formulation and analysis of the model*

The fusion of cellular and loop data for route flow estimation is one of the key contributions of this chapter. We pose the route flow estimation problem as a mathematical program optimizing the fit to link sensor data over feasible route flow distributions, constrained to those which are consistent with measured cellpath flows in the network. We formulate this in the framework of convex optimization as a minimization of a quadratic program:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|Ax - b\|_2^2 \\ \text{s.t.} \quad & Ux = f,\ x \succeq 0 \end{aligned} \tag{84}$$

The problem is a constrained linear inverse problem in which we want to estimate a signal of length n (the route flows) given that we have m measurements (the observable link flows). We additionally have q cellpath flow constraints: for each cellpath $p \in \mathcal{P}$, there are n_p routes corresponding to p, such that their flow must sum up to the cellpath flow f_p:

$$Ux = f : \quad \sum_{r \in \mathcal{R}^p} x_r^p = f_p \quad \forall p \in \mathcal{P} \tag{85}$$

We note that the subsets of routes $\mathcal{R}^p$ are disjoint (each route has at most one cellpath associated to it), hence (85) along with the nonnegativity constraint in (84), together forms a block simplex constraint, which we further analyze in Section 7.3.

In general, $m \ll n$ and $q \leqslant n$, thus typically the Hessian $A^T A$ of our convex quadratic objective is singular ($A^T A \in \mathbb{R}^{n \times n}$ but $\operatorname{rank}(A^T A) \leqslant m \ll n$). Thus the problem might have multiple optimal solutions (underdetermined) or might have more observations than unknowns (overdetermined), depending on the number of cellpath flow constraints. Our cellpath formulation encodes more constraints than methods that consider less detailed flow measurements (e.g. OD flow), thereby constraining the solution space. Moreover, when there are uncorrelated measurement errors on the vector flow b (absence of interactions between the detection process of the link sensors), the ordinary least squares is the best unbiased estimator of the route flow.[4]

We now make the following observation, which is key for our algorithm described in Section 7.3.

Theorem 1. *Problem (84) can be reduced to a least-squares problem with (separable) standard simplex constraints:*

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\tilde{A}\tilde{x} - b\|_2^2 \\ s.t. \quad & \mathbf{1}^T \tilde{x}^p = 1,\ \tilde{x}^p \succeq 0, \quad \forall p \in \mathcal{P} \end{aligned} \qquad \text{where} \quad \tilde{A} \in \mathbb{R}_+^{|\mathcal{L}| \times |\mathcal{R}|} : \tilde{A}_{lr} = \begin{cases} f_p & \text{if } l \in r \in \mathcal{R}^p \\ 0 & \text{else} \end{cases} \tag{86}$$

where $\mathbf{1} = [1, \cdots, 1]^T \in \mathbb{R}^{n_p}$ *and* $\tilde{A}$ *is a modified link-route incidence matrix containing the cellpath flows* f_p.

Proof. The constraints $Ux = f$ in (84) can be written explicitly: $\sum_{r \in \mathcal{R}^p} x_r^p = f_p$, $\forall p \in \mathcal{P}$. With the change of variables $\tilde{x}^p := x^p / f_p$ for all p, the constraints become $\sum_{r \in \mathcal{R}^p} \tilde{x}_r^p =$

4 The errors must also have zero-mean and constant variance, then the result holds as link flows linearly depend on route flows: $\hat{b} = Ax + \epsilon$, from the Gauss-Markov theorem.

1, $\forall p \in \mathcal{P}$, or in matrix form: $\mathbf{1}^T \tilde{x}^p = 1$, $\forall p \in \mathcal{P}$. Since $f_p > 0$ for all p, then the inequalities $x^p \succeq 0$ are equivalent to $\tilde{x}^p = x^p / f_p \succeq 0$. Finally, the vector Ax has entries $v_l = \sum_{r\,:\,l \in r} x_r$ for $l \in \mathcal{L}$, where v_l is the flow on link l. The sum can be decomposed between the different cellpaths p: $v_l = \sum\limits_{r\,:\,l\in r} x_r = \sum\limits_p \left\{ \sum\limits_{r\,:\,l\in r\in\mathcal{R}^p} x_r^p \right\} = \sum\limits_p \left\{ \sum\limits_{r\,:\,l\in r\in\mathcal{R}^p} f_p \tilde{x}_r^p \right\} = (\tilde{A}\tilde{x})_l$, hence the objectives are the same. □

7.2.3 *Compatibility of our formulation*

Our formulation is related to the *traffic assignment problem* (also called the route assignment problem) used to solve traffic equilibrium problems (see Wardrop and Whitehead (1952), Sheffi (1985, §3), M. G. H. Bell and Iida (1997, §5)), where $\mathcal{A}$ is the set of all links (arcs) in the network, $A^{\text{full}} \in [0,1]^{|\mathcal{A}|\times|\mathcal{R}|}$ is the full link-route incidence matrix, and φ is the Beckmann objective function (Beckmann et al., 1956):

$$\min\ \varphi(A^{\text{full}} x) \quad \text{s.t.} \quad Tx = d,\ x \succeq 0 \tag{87}$$

This is a standard formulation in traffic assignment in which a local minimum of (87) is a Wardrop equilibrium of a congestion game (Monderer and Shapley, 1996); the relation between OD and route flows given by $Tx = d$ is equivalent to the "approach proportions" formulation of Bar-Gera (Bar-Gera, 2002). If the cellpath-route incidence matrix U is reduced to an OD-route incidence matrix (see Fig. 32), both (84) and (87) share the same constraints. Furthermore, the cellpath constraints can be added to (87) to restrict its solution space as well. The main difference lies in the minimization objective: in (84) it is the link flows measurement residual while in (87) the Beckmann objective φ expresses the incentives of all vehicles (or players) to take the shortest route.

In the context of game theory, each cellpath can be seen as a player who chooses a strategy or a probability distribution with weights $(\tilde{x}_r^p)_{r\in\mathcal{R}^p}$ over the n_p routes, and a set defined by $S^p := \{\tilde{x}^p \in [0,1]^{n^p} \mid \sum_{r\in\mathcal{R}^p} \tilde{x}_r^p = 1\}$ is a *strategy set* or a *probability simplex* over the routes $r \in \mathcal{R}^p$.

We observe that the traffic assignment problem (87) can also be reduced in a similar fashion, where $\mathcal{O}^2$ is the set of all OD pairs:

$$\begin{array}{ll} \min & \varphi(\tilde{A}^{\text{full}}\tilde{x}) \\ \text{s.t.} & \mathbf{1}^T\tilde{x} = 1,\ \tilde{x}^k \succeq 0,\ \forall k \in \mathcal{O}^2 \end{array} \qquad \text{where} \quad \tilde{A}^{\text{full}} = \begin{cases} d_k & \text{if } l \in r \in \mathcal{R}^k \\ 0 & \text{else} \end{cases} \tag{88}$$

Our formulation in (84) is also compatible with several other types of data, *e.g.* turning ratios, link capacities, OD flows. We demonstrate this by augmenting our problem formulation with turning ratios as follows: If, at some node (intersection) $j \in \mathcal{N}$, we know the flow of vehicles coming from link $a = (i, j) \in \mathbb{A}$ and turning into link $a' = (j, k)$, we denote the pair of successive links by $t = (a, a')$, denote $\mathcal{T}$ the set of monitored traffic turns (intersections), let $G \in \{0, 1\}^{|\mathcal{T}| \times |\mathcal{R}|}$ be the turn-route incidence matrix, and denote h the vector of flow that passes through each monitored intersection. Then, the objective of (84) can be generalized to include turning ratios:

$$\begin{array}{ll} \min & \frac{1}{2}\|A'x - b'\|_2^2 \\ \text{s.t.} & Ux = f,\ x \succeq 0 \end{array} \qquad \text{where} \quad A' = \begin{bmatrix} A \\ G \end{bmatrix}, \quad b' = \begin{bmatrix} b \\ h \end{bmatrix} \quad \text{and} \tag{89}$$

$$G_{tr} = \begin{cases} 1 & \text{if} \quad t = (a, a') : a, a' \in r \\ 0 & \text{otherwise} \end{cases} \tag{90}$$

Similarly, the objective of (84) can be generalized to include OD flows, and we later demonstrate the incorporation of this information in our numerical experiments:[5]

$$\min \frac{1}{2}\|A'x - b'\|_2^2 \quad \text{s.t.} \quad Ux = f,\ x \succeq 0 \qquad \text{where} \quad A' = \begin{bmatrix} A \\ T \end{bmatrix} \quad \text{and} \quad b' = \begin{bmatrix} b \\ d \end{bmatrix} \tag{91}$$

Suppose we know the link capacities $\tilde{m}_a$, then the constraints $A^{\text{full}}x \preceq \tilde{m}$, where $\tilde{m} := (\tilde{m}_a)_{a \in \mathcal{A}}$ is the link capacities vector, can be added to program (84). To approximate the new problem as a program with simplex constraints, we can make the added constraints implicit in the objective:

$$\min \frac{1}{2}\|Ax - b\|_2^2 + \sum_{a \in \mathbb{A}} \Phi(L_a^T x - \tilde{m}_a) \quad \text{s.t.} \quad Ux = f,\ x \succeq 0 \tag{92}$$

where the barrier Φ is an approximation of the indicator function of $\mathbb{R}_-$ given as $I_-(u) = (0 \text{ if } u \leqslant 0 \text{ else })$, and the vectors L_a^T, $a \in \mathbb{A}$ are the rows of A^{full}. A common choice for Φ is the logarithm barrier $\Phi(u) = -\alpha \log(-u)$ where $\alpha > 0$ is a parameter that sets the

5 Since the inequalities $Ux = f,\ Tx = d,\ x \succeq 0$ might not define simplexes, we chose formulation (91) over: $\min \frac{1}{2}\|Ax - b\|_2^2$ s.t. $Ux = f,\ Tx = d,\ x \succeq 0$ to have the same constraints as in (84) for our algorithmic approach. Besides, with dense cellular networks, satisfying $Tx = d$ is redundant with the constraints $Ux = f$ because OD demands are included in cellular network data, hence both formulations reduce to (84).

accuracy of the approximation (Boyd and Vandenberghe, 2004, §11.2.1).

In summary, we pose the route flow estimation problem as a mathematical program optimizing the fit to link sensor data (and possibly other sources of data) over feasible route flow distributions, constrained to those which are consistent with measured cell-path flows in the network. Our data-driven approach is compatible with other types of data and also similar in formulation to route-based traffic assignment models.

7.3 DIMENSIONALITY REDUCTION AND PROJECTION VIA ISOTONIC REGRESSION

In this section, we present an efficient constraint elimination technique relying on the choice of a particular nullspace, which is suitable for both the proposed route flow estimation problem (84) and the traffic assignment problem (87). The projection on the inequality constraints is performed in linear time via isotonic regression.

7.3.1 *Exploiting the structure of the equality constraints*

We consider the reduced route flow estimation problem (86) and the reduced traffic assignment problem (87):

$$\begin{aligned} &\text{route flow estimation problem:} && \min_x \tfrac{1}{2}\|\tilde{A}\tilde{x} - b\|_2^2 \quad \text{s.t.} \quad \mathbf{1}^T\tilde{x}^p = 1,\ \tilde{x}^p \succeq 0, \quad \forall\, p \in \mathcal{P} \\ &\text{traffic assignment problem:} && \min_x \varphi(\tilde{A}^{\text{full}}\tilde{x}) \quad \text{s.t.} \quad \mathbf{1}^T\tilde{x}^k = 1,\ \tilde{x}^k \succeq 0, \quad \forall\, k \in \mathcal{O}^2 \end{aligned} \tag{93}$$

We consider a general objective function f and the simplices $S^p = \{\tilde{x}^p \in [0,1]^{n^p} \mid \sum_{r\in\mathcal{R}^p} \tilde{x}^p_r = 1\}$ as constraints, but the following analysis applies for both problems. We use standard linear algebra operations to eliminate the equality constraints (Boyd and Vandenberghe, 2004, §4.2.4). Since the constraints have disjoint support, we treat each one of them separately. For all $p \in \mathcal{P}$, we find a direction e^p which is a particular solution of $\mathbf{1}^T\tilde{x}^p = 1$, and a matrix N^p whose range is the *orthogonal complement* of the vector $\mathbf{1} \in \mathbb{R}^{n_p}$, denoted $\{t\,\mathbf{1} \mid t \in \mathbb{R}\}^\perp$. With the vectors $\{e^p\}_{p\in\mathcal{P}}$ stacked into an overall vector $\tilde{x}_0 := (e^p)_{p\in\mathcal{P}}$, and the matrices $\{N^p\}_{p\in\mathcal{P}}$ encoded in an overall block-diagonal matrix $N := \operatorname{diag}((N^p)_{p\in\mathcal{P}})$,

the resulting problem is:

$$\begin{array}{ll}\min\limits_{z} & \frac{1}{2}f(\tilde{x}_0 + Nz) \\ \text{s.t.} & \tilde{x}_0 + Nz \succeq 0\end{array} \quad ; \quad \text{or with blocks made explicit:} \quad \begin{array}{ll}\min\limits_{z} & f((e^p + N^p z^p)_{p\in\mathcal{P}}) \\ \text{s.t.} & e^p + N^p z^p \succeq 0, \quad \forall\, p \in \mathcal{P}\end{array} \tag{94}$$

Vectors of the form $[\cdots, 1, -1, \cdots]^T$ are orthogonal to $\mathbf{1} \in \mathbb{R}^{n_p}$. We also choose a simple e^p solution of $\mathbf{1}^T x^p = 1$:

$$e^p := [0, \cdots, 0, 1]^T \in \mathbb{R}^{n_p}; \quad N^p = \begin{bmatrix} 1 & & \\ -1 & 1 & \\ & -1 & \ddots \\ & & \ddots \end{bmatrix} \in \mathbb{R}^{n_p \times (n_p - 1)} \quad \forall\, p \in \mathcal{P} \tag{95}$$

where the columns of N^p form a basis of $\{t\,\mathbf{1} \,|\, t \in \mathbb{R}\}^{\perp}$. These choices result in a simplification of the constraints in (94), and we can interchangeably operate on variables x^p in (84) and variables z^p in (86) since they are simply related:

$$\begin{aligned} \tilde{x}^p &= e^p + N^p z^p = [z_1^p,\, z_2^p - z_1^p,\, \cdots,\, z_{n_p}^p - z_{n_p-1}^p,\, 1 - z_{n_p}^p]^T, \forall\, p \in \mathcal{P} \\ z^p &= [\tilde{x}_1^p,\, \tilde{x}_1^p + \tilde{x}_2^p, \cdots,\, \textstyle\sum_{i=1}^{n-2} \tilde{x}_i^p,\, \sum_{i=1}^{n-1} \tilde{x}_i^p,]^T, \forall\, p \in \mathcal{P} \end{aligned} \tag{96}$$

The constraint $e^p + N^p z^p \succeq 0$ becomes an ordering constraint $0 \leqslant z_1^p \leqslant \cdots \leqslant z_{n_p-1}^p \leqslant 1$. The program (94) is now:

$$\min_{z} f((e^p + N^p z^p)_{p\in\mathcal{P}}) \quad \text{s.t.} \quad 0 \leqslant z_1^p \leqslant \cdots \leqslant z_{n_p-1}^p \leqslant 1, \quad \forall\, p \in \mathcal{P} \tag{97}$$

The main advantage of this constraint elimination is the reduction of the dimension from n to $n - q$, where n is the number of routes and q the number of cellpaths (see Table 6). If each cellpath has maximum k routes, then we have $n \leqslant kq$, hence $n - q \leqslant n(1 - 1/k)$. For our target problem, we generally have $k \in [3, 50]$ hence the dimension can be reduced by as much as $1/3$.

The problem (97) can be solved quite efficiently with a simple (accelerated) first order or second order projection algorithm, or an Augmented Lagrangian method. In particular, the basic descent projection algorithm (see Algorithm 3) iteratively takes a step in a descent direction Δz (line 2) from the current point z, projects the new point $z + \Delta z$ onto the constraint set $z^+ := \Pi(z + \Delta z)$ (line 3), and performs a line search (line 4). The projection step is performed with q Euclidean projections of $z^p + \Delta z^p$ onto ordering

constraints:

$$\Pi^p(y^p): \quad \min_{u^p} \|u^p - y^p\|_2^2 \quad \text{s.t.} \quad 0 \leqslant u_1^p \leqslant u_2^p \leqslant \cdots \leqslant u_{n_p-1}^p \leqslant 1 \quad \forall\, p \in \mathcal{P} \tag{98}$$

Algorithm 3 `Proj-descent(·)` General projected descent method

Require: initial point $z = (z^p)_{p\in\mathcal{P}}$ in the feasible set $\mathcal{X}$.

1: **while** stopping criteria not met **do**
2: Determine a descent direction $\Delta z = (\Delta z^p)_{p\in\mathcal{P}}$
3: Projection: $(z^p)^+ := \underset{u^p}{\operatorname{argmin}}\{\|z^p + \Delta z^p - u^p\|^2 : 0 \leqslant u_1^p \leqslant u_2^p \leqslant, \cdots \leqslant u_{n_p-1}^p \leqslant 1\}$, $\forall\, p \in \mathcal{P}$
4: Line search on the projected arc: $\gamma :\approx \operatorname{argmin}\{f(z + t(z^+ - z)) : t \in [0,1]\}$
5: $z := z + \gamma(z^+ - z)$
6: **end while**
7: **return** z

In line 4 of Algorithm 3, we perform a backtracking line search (Boyd and Vandenberghe, 2004, §9.2). This is an Armijo-rule based step size selection that ensures sufficient descent, it approximately minimizes the objective along the projected arc $\{z + t(z^+ - z) \,|\, t \in [0,1]\}$. Since the feasible set is convex, the projected arc is feasible, hence the method also ensures feasibility of the next iterate. We apply backtracking with objective $f(z) = \|A(\tilde{x}_0 + Nz)\|_2^2$ and descent direction $d = z^+ - z$.

7.3.2 *A simple projection using isotonic regression*

The projections (98) have general form (99), given data points $y := [y_1, \cdots, y_n] \in \mathbb{R}^n$, weights $w := [w_1, \cdots, w_n] \succ 0$, and bounds $L < U$.[6] Without bounds, we have an isotonic regression problem (100) (see Tibshirani et al. (2011) and references therein).

$$\text{ISO}_{1\to n}^{[L,U]}(y,w): \quad \min_u \textstyle\sum_{i=1}^n w_i(y_i - u_i)^2 \quad \text{s.t.} \quad L \leqslant u_1 \leqslant u_2 \leqslant \cdots \leqslant u_n \leqslant U \tag{99}$$

$$\text{ISO}_{1\to n}^{\mathbb{R}}(y,w): \quad \min_u \textstyle\sum_{i=1}^n w_i(y_i - u_i)^2 \quad \text{s.t.} \quad u_1 \leqslant u_2 \leqslant \cdots \leqslant u_n \tag{100}$$

where we use the notation $\text{ISO}_{s\to t}^{I}(y,w)$ such that subscript $s \to t$ means we only consider data points with indices from s to t, and superscript I is the interval in which the

6 For subsection 7.3.2 only, $U \in \mathbb{R}$ is the upper bound in problem (99). In the rest of the chapter, U is the cellpath-route incidence matrix.

variables $u_s, u_{s+1}, \cdots, u_t$ lie. Since both problems are strongly convex, they both have unique solutions. The solution to (100), denoted u^{iso}, can be computed in linear time using the *Pool Adjacent Violators* (PAV) algorithm (Best and Chakravarti, 1990, §3), so one hopes that the solution to (99), denoted $u^\star$, derives easily from u^{iso}. In fact, we prove the following result:

Theorem 2. *The solution $u^\star$ to (99) is the Euclidean projection of the solution u^{iso} to (100) onto $[L, U]^n$.*

Although isotonic regression is generally studied in the form (100), the bounded version (99) has appeared in Grotzinger and Witzgall (1984). The simple connection presented in Theorem 2 is new to the best of our knowledge. This result can be written $u^\star = \Pi_{[L,U]^n}(u^{\text{iso}})$ where $\Pi_{\mathcal{K}}$ is the Euclidean projector onto space $\mathcal{K}$. When $\mathcal{K} = [L, U]^n$, the projected vector $p := \Pi_{[L,U]^n}(u)$ is obtained from $u \in \mathbb{R}^n$ by simply projecting each entry u_i onto $[L, U]$, *i.e.* $p_i = u_i$ if $u_i \in [L, U]$, $p_i = L$ if $x_i < L$, and $p_i = U$ if $x_i > U$. We first give a lemma.

Lemma 1. *Given u^{iso} the solution to (100), if there exists k such that $u_k^{iso} < u_{k+1}^{iso}$ then (100) reduces to two subproblems:*

$$\begin{array}{ll} ISO^{\mathbb{R}}_{1\to k}(y, w): & \min_u \sum_{i=1}^k w_i(y_i - u_i)^2 \quad s.t. \quad u_1 \leqslant \cdots \leqslant u_k \\ ISO^{\mathbb{R}}_{k+1\to n}(y, w): & \min_u \sum_{i=k+1}^n w_i(y_i - u_i)^2 \quad s.t. \quad u_{k+1} \leqslant \cdots \leqslant u_n \end{array} \tag{101}$$

such that $[u_1^{iso}, \cdots, u_k^{iso}]$ is the solution to the first one and $[u_{k+1}^{iso}, \cdots, u_n^{iso}]$ is the solution to the second one. The same result holds for (99) and $u^\star$, with resulting subproblems $ISO^{[L,+)}_{1\to k}(y, w)$ and $ISO^{(-,U]}_{k+1\to n}(y, w)$.

Proof. Since the constraint $u_k \leqslant u_{k+1}$ is not active at u^{iso}, it may be removed from (100) without altering the solution. Then the resulting program can be separated into the two programs in (101) with respective solutions $[u_1^{\text{iso}}, \cdots, u_k^{\text{iso}}]$ and $[u_{k+1}^{\text{iso}}, \cdots, u_n^{\text{iso}}]$. □

Proof of Theorem. We start with two simple cases.
Case 1: $[u_i^{\text{iso}} \leqslant L, \forall i]$. Suppose $\exists k, u_k^\star > L$. We choose k the smallest of such indices, then either $k = 1$ or $L = u_{k-1} < u_k$. In both cases, $[u_k^\star, \cdots, u_n^\star]$ is the unique solution

to $\mathrm{ISO}_{k\to n}^{(-,U]}(y,w)$ from Lemma 1. Since $[u_k^{\mathrm{iso}},\cdots,u_n^{\mathrm{iso}}]$ is also feasible for $\mathrm{ISO}_{k\to n}^{(-,U]}(y,w)$, we have $\sum_{i=k}^n w_i(y_i-u_i^{\mathrm{iso}})^2 > \sum_{i=k}^n w_i(y_i-u_i^\star)^2$, and adding $\sum_{i=1}^{k-1} w_i(y_i-u_i^{\mathrm{iso}})^2$ on both sides yields $\sum_{i=1}^n w_i(y_i-u_i^{\mathrm{iso}})^2 > \sum_{i=1}^{k-1} w_i(y_i-u_i^{\mathrm{iso}})^2 + \sum_{i=k}^n w_i(y_i-u_i^\star)^2$. Since $[u_1^{\mathrm{iso}},\cdots,u_{k-1}^{\mathrm{iso}},u_k^\star,\cdots,u_n^\star]$ is also feasible for (100) ($u_{k-1}^{\mathrm{iso}} \leqslant l < u_k^\star$), this contradicts the optimality of u^{iso}. Hence $u_k^\star = L$, $\forall\, k$, *i.e.* $u^\star = \Pi_{[L,U]^n}(u^{\mathrm{iso}})$.

Case 2: $[u_i^{\mathrm{iso}} \geqslant U, \forall\, i]$. The analysis is similar to case 2. We have: $u_k^\star = U$, $\forall\, k$, *i.e.* $x^\star = \Pi_{[L,U]^n}(u^{\mathrm{iso}})$.

General case: Without loss of generality, we suppose there exist two indices s,t such that: $u_1^{\mathrm{iso}} \leqslant \cdots \leqslant u_s^{\mathrm{iso}} \leqslant L < u_{s+1}^{\mathrm{iso}} \leqslant \cdots \leqslant u_{t-1}^{\mathrm{iso}} < U \leqslant x_t^{\mathrm{iso}} \leqslant \cdots \leqslant x_n^{\mathrm{iso}}$. From Lemma 1, $[u_1^{\mathrm{iso}},\cdots,u_s^{\mathrm{iso}}]$, $[u_{s+1}^{\mathrm{iso}},\cdots,u_{t-1}^{\mathrm{iso}}]$, and $[u_t^{\mathrm{iso}},\cdots,u_n^{\mathrm{iso}}]$ are then solutions to

$$\mathrm{ISO}_{1\to s}^{\mathbb{R}}(y,w),\ \mathrm{ISO}_{s+1\to t-1}^{\mathbb{R}}(y,w),$$

and $\mathrm{ISO}_{t\to n}^{\mathbb{R}}(y,w)$ respectively. From case 1, the vector $[L,\cdots,L] \in \mathbb{R}^s$ is solution to $\mathrm{ISO}_{1\to s}^{[L,+)}(y,w)$ and from case 2, the vector $[U,\cdots,U] \in \mathbb{R}^{n-t+1}$ is solution to $\mathrm{ISO}_{t\to n}^{(-,U]}(y,w)$. Then the global vector $x^* := [L,\cdots,L,u_{s+1}^{\mathrm{iso}},\cdots,u_{t-1}^{\mathrm{iso}},U,\cdots,U]$ is the solution to the global program:

$$\begin{array}{ll} \min_u & \sum_{i=1}^n w_i(y_i-u_i)^2 \\ \text{s.t.} & L \leqslant u_1 \leqslant \cdots \leqslant u_s, \quad u_{s+1} \leqslant \cdots \leqslant u_{t-1}, \quad u_t \leqslant \cdots \leqslant u_n \leqslant U \end{array} \tag{102}$$

Adding the constraints $u_s \leqslant u_{s+1}$ and $u_{t-1} \leqslant u_t$ to (102) does not alter the solution since they are inactive. Hence $[L,\cdots,L,u_{s+1}^{\mathrm{iso}},\cdots,u_{t-1}^{\mathrm{iso}},U,\cdots,U]$ is the solution to (99), *i.e.* $u^\star = \Pi_{[L,U]^n}(u^{\mathrm{iso}})$. □

Algorithm 4 `PAV-proj(`y^p`)` Projection onto ordering constraints in line 3 of Algorithm 3

Require: vector $y^p \in \mathbb{R}^{n_p-1}$

1: compute $y^{p,\mathrm{iso}} := \underset{u^p}{\mathrm{argmin}}\{\|u^p-y^p\|_2^2 : u_1^p \leqslant u_2^p \leqslant \cdots \leqslant u_{n_p-1}^p\}$ with the PAV algorithm (Best and Chakravarti, 1990)

2: project $y^{p,\mathrm{iso}}$ onto $[0,1]^{n_p-1}$: $\tilde{y}_k^p = y_k^{p,\mathrm{iso}}$ if $y_k^{p,\mathrm{iso}} \in [0,1]$; $\tilde{y}_k^p = 0$ if $y_k^{p,\mathrm{iso}} \leqslant 0$; $\tilde{y}_k^p = 1$ if $y_k^{p,\mathrm{iso}} \geqslant 1$.

3: **return** return $\tilde{y}^p$

In Algorithm 4, we give an efficient algorithm to perform the projections (98) in line 3 of Algorithm 3. We note that without the constraint elimination described earlier, a projected descent method applied to (86) would require q projections onto the probability simplices $\{\tilde{x}^p \in \mathbb{R}^{n_p} \,|\, \mathbf{1}^T\tilde{x}^p = 1,\ \tilde{x}^p \succeq 1\}$ at each iteration. The complexity of these projections is $O(n_p \log n_p)$ (Duchi et al., 2008a; W. Wang and Carreira-Perpinán, 2013), which is less attractive than the $O(n_p)$ complexity of Algorithm 4.

Problems (86) and (88) are both convex, and can be solved efficiently with including interior point methods, augmented Lagrangian, gradient projection, and conjugate gradient. In particular, we choose the *Barzilai and Borwein* (BB) method for the accelerated gradient method, where z is the current iterate and z^- and previous iterate:

$$\Delta z = -((y^T s)/(y^T y))\Delta f(z) \quad \text{where} \quad y = \nabla f(z) - \nabla f(z^-), \quad s = z - z^- \tag{103}$$

The change of variable reduces the dimensionality, at the cost of losing some of the intuitive structure of the route choice problem. While long-standing algorithms such as the Frank-Wolfe assignment (LeBlanc et al., 1975) and the Origin-based assignment (Bar-Gera, 2002) and their modifications may have diminished efficiency since the all-or-nothing assignment step is no longer available, their slow convergence is known (Ortuzar and Willumsen, 2001, §11.2.3.1). We propose that the estimation problem (86) and the traffic assignment problem (88) can be reduced to the form (94), and then be solved efficiently with quasi-Newton methods (*e.g.* L-BFGS (Nocedal and Wright, 2006)), accelerated gradient methods, or alternating direction methods. These algorithms are proven to have fast convergence, and the proposed projection step is efficient as discussed above.

7.4 EXPERIMENTAL SETTING AND VALIDATION PROCESS

We demonstrate our approach by providing numerical results on networks of varying sizes, applying different traffic assignment models and sensor configurations, all based on the I-210 highway corridor in Los Angeles. To demonstrate the versatility to the underlying data-driven approach, we investigate the following three scenarios (see Figure 33):

1. *Highway network in user equilibrium* (UE), with varying cell densities and static sensor coverage.
2. *Highway network in system optimum* (SO), with varying cell densities and static sensor coverage.
3. *Activity-based agent model on full network*, with varying cell densities, 5% static sensor

coverage.

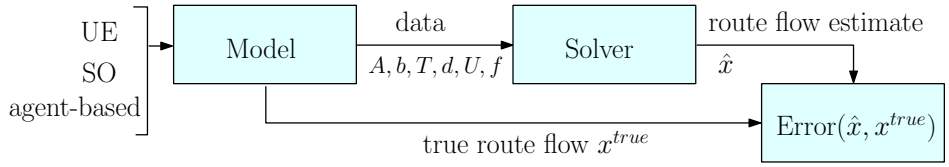


Figure 33: Our experiment flow block diagram, where the model is comprised of a network, traffic assignment model, and sensor configuration. The solver is presented in Section 7.3. The error metric represented here is a function of the estimated and actual route flow. We may compute the percent flow error or, using additional information (e.g. network topology and actual link flow), we may also compute the link flow GEH error.

We have intentionally selected three different traffic assignment models (UE, SO, agent-based) to test the versatility of our method; we aim to demonstrate that our method is not only accurate (provided enough measurements) and appropriate for full-scale networks but also model agnostic, thereby highlighting a major advantage of our approach to those that require more rigid assumptions on agent behavior. Thus, we study networks of different sizes and complexities, different driver behavior models, and trade-offs for different sensor placements. We additionally present preliminary investigation on the effect of measurement and model error on the accuracy of the approach.

7.4.1 *Sensor configurations*

We have two main types of data: link sensors data (loop based) and cellpath sensors (cell based). We consider link sensors on a subset of the links in the network (ranging from 5% to 100% coverage). For the highway network with UE/SO flow, the subsets of links are chosen such that the *most congested* links are observed, *i.e.* links with highest traffic volumes or flows, whereas in the full large-scale network, we use locations of real highway (PeMS, see Choe et al. (2002)) and arterial loop sensors where the coverage is 5%.

Although the use of real cellular network data from a service provider would demonstrate even stronger applicability of our framework, its availability is restricted for privacy issues. Our team at the present time is not able to share findings based on collaborations with companies such as AT&T. Nevertheless, the use of well-designed simulators remains necessary for demonstrating how of our framework may apply to different networks and settings, and also for the ease of validation, as route flows are not yet measurable in real-world settings. Our model for cell placement is based on employee

population density and locations of major roads. Most notably, many ordinances prohibit towers in residential areas but promote towers in industrial and commercial centers. For both networks, the locations $(X_i, Y_i) \in \mathbb{R}^2$ of the cell towers are randomly sampled on the plane such that the distribution models realistically represent the coverage based on region demographics. The overall sensor configuration (104,105,106) consists of $N = N^B + N^S + N^L$ total cell towers, where N^B, N^S, N^L are predetermined for each experiment and the weights of the multinomial distributions are determined by demographics and geometry. Our sensor configurations are drawn from three distribution models:

1. Within the whole region delimited by a *bounding box*: N^B cell tower locations $\{(X_i^B, Y_i^B)\}_{i=1,\cdots,N^B}$ are sampled uniformly (104).
2. Within sub-regions $\mathcal{S}$ comprising the full region: For each sub-region $s \in \mathcal{S}$, delimited by a rectangle (X^s_{min}, Y^s_{min}), (X^s_{max}, Y^s_{max}), N^s additional cell tower locations $\{(X_i^s, Y_i^s)\}_{i=1,\cdots,N^s}$ are sampled (105). The number of base stations N^s for each sub-region s is sampled from a multinomial distribution with N^S trials and weights proportional to demographic information for each region (e.g. employee population). That is, $N^S = \sum_{s\in\mathcal{S}} N^s$ is the total number of cell towers among all the sub-regions (excluding those sampled from the entire bounding box).
3. Near major links in the network: Along each link $a \in \tilde{\mathcal{A}} \subset \mathcal{A}$ (also called *arcs*), where $\tilde{\mathcal{A}}$ denotes a pre-selected subset of major links in the network, N^a cell tower locations are sampled uniformly along the link with Gaussian noise (106) where (X_s^a, Y_s^a) is the location of the start of link a, and (X_t^a, Y_t^a) is the location of the end of link a. The numbers of base stations N^a along links $a \in \mathbb{A}$ are sampled from a multinomial distribution with N^L trials and weights proportional to the length of a, where N^L is the total number of cells along links.

$$\text{Bounding box}: \quad X_i^B \sim U([X^B_{min}, X^B_{max}]),\ Y_i^B \sim U([Y^B_{min}, Y^B_{max}]), \quad \text{for} \quad i = 1, \cdots, N^B \tag{104}$$

$$\text{Sub-region } S: \quad X_i^s \sim U([X^s_{min}, X^s_{max}]),\ Y_i^s \sim U([Y^s_{min}, Y^s_{max}]), \quad \text{for} \quad i = 1, \cdots, N^s \tag{105}$$

$$\text{Link } a: \quad \begin{cases} X_i^a \sim X_s^a + t_i(X_t^a - X_s^a) + N(0, \sigma) \\ Y_i^a \sim Y_s^a + t_i(Y_t^a - Y_s^a) + N(0, \sigma) \end{cases} \quad \text{s.t. } t_i \sim U([0, 1]), \text{ for } i = 1, \cdots, N^a \tag{106}$$

7.4.2 *Scenarios 1 and 2: UE and SO on the highway network*

We consider first the *highway network* of the I-210 region in Los Angeles[7]. The roads are extracted from OpenStreetMaps (OSM) and we retain only links with at least five (up to 11) lanes. This results in a directed graph $\mathcal{G} = (\mathcal{N}, \mathbb{A})$ with $|\mathcal{N}| = 44$ nodes and $|\mathbb{A}| = 122$ directed links. We calibrate the free flow delay τ_a for each link $a \in \mathbb{A}$ using the link's length and free flow speed (provided by OSM) and empirical delays values (provided by Google Maps). An illustration of the network is provided in Fig. 34.

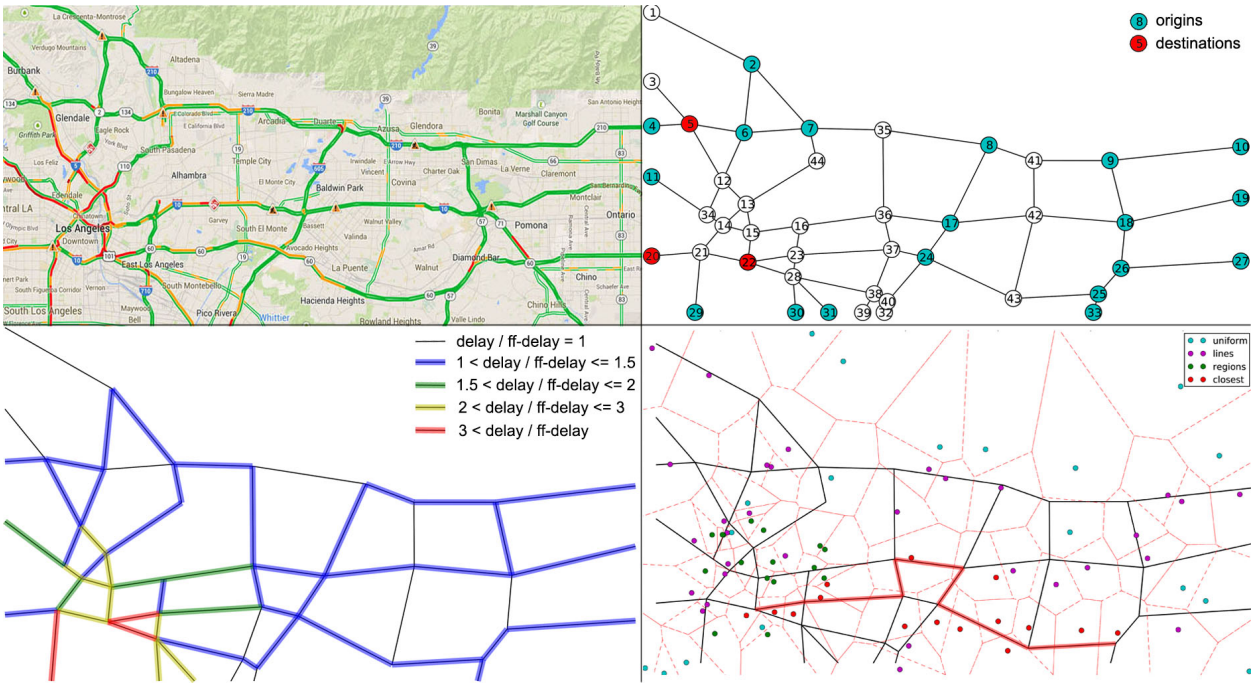


Figure 34: Benchmark (small-scale) example used for the first numerical run: The four subfigures present the highway network of the I-210 highway corridor in L.A. county. Starting from the top left and in clockwise order: 1) The state of traffic on 2014-06-12 at 9:14 AM from Google Maps; 2) The nodes in blue and red are nodes from which positive flows emanate, nodes in red are nodes from which positive flows terminate; 3) Network with 80 sampled cells, with a higher concentration of cells near downtown. A random path from 25 to 22 is shown in red with the closest cell towers. 4) The highway network in user equilibrium with the resulting delays. Best viewed in color.

The OD demands are based on census data and employment concentration in L.A.

7 The region has bounding box [-118.328299, 33.984601, -117.68132, 34.255881] in latitude and longitude coordinates.

county, which are extracted from the Census Bureau. The OD demand model is simplified to a static morning rush hour model[8] of the region such that: i) only 21 origins have positive flows emanating from them; ii) all the trips terminate at three destinations: near Burbank at node 5, towards Santa Monica at node 20, and in Downtown L.A. at node 22; iii) we only have 42 OD pairs with positive flows ranging from 1200 veh/hour to 12,000 veh/hour.[9]

In our equilibrium-based numerical study, we consider the traffic assignment model presented in Sheffi (1985, §3.1) to generate route flows and cellpath flows. The delay on a given link a is assumed to be a strictly increasing function $c_a(\cdot)$ of the traffic volume (flow) v_a on that link. We choose the widely used delay function estimated by the Bureau of Public Roads, where τ_a is the free flow delay (sec) and m_a is the number of lanes on link a, and provide the Beckmann objective function φ^{UE} associated to the overall model (Beckmann et al., 1956)):

$$\text{link delay:} \quad c_a(v_a) = \tau_a(1 + 0.15(v_a/m_a)^4), \ \forall\, a \in \mathbb{A} \tag{107}$$

$$\text{UE objective:} \quad \varphi^{\mathrm{UE}}(v) = \sum_{a \in \mathbb{A}} \int_0^{v_a} c_a(u)\,du \tag{108}$$

In this section only, we use the following notation: the nodes of the network are indexed by $i \in \mathcal{N}$, the 42 OD pairs with positive OD flow are indexed by $k \in \{1, \cdots, Q\}$, $A^{\mathrm{full}} \in \mathbb{R}^{|\mathbb{A}| \times |\mathcal{R}|}$ is the link-route incidence matrix, and $N \in \{-1, 0, 1\}^{|\mathcal{N}| \times |\mathbb{A}|}$ is the node-link incidence matrix. For each OD pair $k = (s_k, t_k)$ we define an associated vector $e^k \in \mathbb{R}^{|\mathcal{N}|}$ such that $e^k_i = -d_k$ at node $i = s_k$ (the origin), $e^k_i = d_k$ at node $i = t_k$ (the destination), and $e^k_i = 0$ otherwise. Under the assumptions of our experiment, the path-flow traffic assignment (PTA) is equivalent to the link-flow traffic assignment (LTA), *i.e.* they give the same *unique* link flow solution (Ford and Fulkerson, 1962):

$$\begin{aligned} \text{PTA}: \ & \min \varphi^{\mathrm{UE}}(A^{\mathrm{full}} x) \\ & \text{s.t. } Tx = d, \ x \succeq 0 \end{aligned} \tag{109}$$

8 Based on observed flows on 2014-06-12 at 9:14 AM from Google Maps.

9 Highway experiment implementation (Python) is available at https://github.com/megacell/traffic-estimation-wardrop.

$$\text{LTA}: \ \min \varphi^{\text{UE}}(v) \tag{110}$$
$$\text{s.t. } v \in \mathcal{K} := \left\{ v \in \mathbb{R}_+^{|\mathbb{A}|} \mid \exists w^k \in \mathbb{R}_+^{|\mathbb{A}|}, v = \sum_{k=1}^{Q} w^k, Nw^k = e^k, \forall k \in \{1, \cdots, Q\} \right\}$$

Since PTA is not tractable due to the computational cost of enumerating all the possible routes, we solve LTA in the first step, then perform the following steps to generate a set of routes $\mathcal{R}$ with an associated UE route flow vector $x^{\text{UE}} \in \mathbb{R}_+^{|\mathcal{R}|}$, and a set $\mathcal{P}$ of cellpaths with a feasible UE cellpath flow vector $f^{\text{UE}} \in \mathbb{R}_+^{|\mathcal{P}|}$:

1. We solve LTA and obtain the UE link flow $v^{\text{UE}} \in \mathbb{R}_+^{|\mathbb{A}|}$ and resulting link delays.
2. We find the K-shortest paths with the UE delays for each of the 42 OD pairs, using Yen's algorithm (J. Y. Yen, 1971). Note that K is chosen large enough such that *at least* all used routes are extracted, *i.e.* all the routes with the (same) shortest delays as characterized by the Wardrop equilibrium. We choose $K = 5$ and extract 207 candidate routes.
3. We solve PTA with the 207 *candidate routes* starting from a random initial point. Let x^{UE} be a route flow solution (the resulting link flow $A^{\text{full}}x^{\text{UE}}$ should equal to v^{UE} since the UE link flow is unique).
4. We sample cells on the highway network following the model presented in Section 7.4.1 (see Fig. 34).
5. Among the 207 routes, we found $\left|\{r \mid x_r^{\text{UE}} > 0\}\right| = 90$ *used routes*. We compute the sequence of cells that intersects with each used route to determine the cellpath flows, given by: $f_p^{\text{UE}} = \sum_{r \in \mathcal{R}^p} x_r^p$.

On a network with SO flow, the total delay is minimized (Wardrop and Whitehead, 1952; Kelly, 1991), hence the objective function to be minimized is φ^{SO} in (111) subject to the constraints in (109) and (110), for the path-flow and link-flow formulations, respectively. In fact, the SO link flow corresponds to the UE link flow with the modified delay function $\tilde{c}_a(\cdot)$ in (111), called the marginal delay function (Roughgarden, 2003) (where the prime indicates the derivative function):

$$\text{link marginal delay}: \quad \tilde{c}_a = c_a(v_a) + v_a c_a'(v_a) \ ; \quad \text{SO objective:} \quad \varphi^{\text{SO}}(v) = \sum_{a \in \mathbb{A}} v_a c_a(v_a) \tag{111}$$

Steps 1 to 5 are performed for the SO objective φ^{SO} via the link marginal delay formulation to generate a SO route flow x^{SO} and a SO cellpath flow f^{SO} on the highway network described above, with a few minor differences:

- In step 2, we find the K-shortest paths under the marginal delays induced by the SO link flow. We choose $K = 10$ and we extract 411 candidate routes.
- In step 5, we found 164 routes with positive flow on it.

7.4.3 *Scenario 3: activity-based agent model on the large-scale full network*

Figure 35: Full-scale network including highway and arterial networks of the I-210 corridor used for MATSim data generation, and for the estimation problem. See Figure 31 for the Voronoi tessellation model of the cellular network and the 700 origins given by the TAZ.

We additionally consider a large *full network*, consisting of both the highway network and the arterial networks in the region. We use the OpenStreetMaps network of the greater Los Angeles area, excluding residential links. Our network consists of 20,513 edges (links) and 10,538 nodes (intersections). We take the origins to be the Traffic Analysis Zones (TAZ) given by the US census, of which there are 778 in the region (see Fig. 35). We use a commercially available OD model for the region, called the Census Transportation Planning Products (CTPP) model.

On this large-scale network, we utilize an *activity-based agent model* for simulating the traffic assignment. MATSim is a well-known open-source traffic simulation framework (Illenberger and Nagel, 2007), which takes in a set of K agent home and work locations and outputs a set of K trajectories (time-stamped sequences of links) that each agent

performed. MATSim searches for a user equilibrium in terms of utility functions defined for the agents using a co-evolutionary optimization algorithm. In our setting, we consider agent utility as a function of travel time. MATSim differs from the user equilibrium model above in that it is quasi-static, by allowing slight variation in the departure times for each agent. MATSim is suitable for performing large-scale agent simulations; we simulate the morning and evening rush-hours using 500,000 agents, as those are the most vital times to understand the state of traffic. The home and work locations for each agent are distributed randomly according to census demographics. Since these locations are selected randomly within origins and destination (as opposed to selected randomly among the region centroids), typically all of the trajectories generated are unique. Viewing the full set of trajectories as our set of possible routes lends itself to be a trivial problem in our formulation. Instead of all routes, we consider the "important" routes in the network. Therefore, we examine trajectories between each OD pair and group them by similarity as follows: 1) Find the trajectory which matches with the most other trajectories ($\geqslant$80% match in length). Add this trajectory to the list of routes for the OD pair; 2) Remove all trajectories that match with this route and repeat. We stop when 50 routes are selected or when there are no more trajectories. 50 routes empirically accounts for 99.4% of the 500K trajectories. This procedure yields a set of 304,695 routes.

7.4.4 *Implementation*

Our full system is available at `https://megacell.github.io/` for validation and extension, including the optimization routine, the small-scale network experiments, and the large-scale pipeline. We hope that our framework will be a benchmark for many future studies of estimation problems in transportation science. The software to run the full-scale experiments was developed mostly in `python 2.7`, using the `GEOS (v.3.4.2)` library for geometric computations. All data is managed and stored in a `PostGIS 2.1.3` database. The geometries and other data about routes, cell tower Voronoi tesselations, and the links of the road network are all stored in the database with spatial indices on all geometry columns, allowing `PostGIS` spatial queries to be performed efficiently for extracting cellpath information associated with each route. The convex optimization program[10] was developed in Python, using `scipy.sparse` and `numpy` for matrix computation. Finally, the PAV projection algorithm was written in `C`, and bindings were written so that it could be called from the Python optimization algorithm.

10 Implementation (Python, C) is open source and available at `https://github.com/megacell/block-simplex-least-squares`.

The full network dataset for the I-210 corridor contains 280x691 routes, 778 origins, 1033 sensors, and was tested with a variety of different cells, ranging in number from 250 to 8000. The incidence matrices U (roughly 250K-by-300K matrices) are generated by finding the cellpath for each route from the database by ordering the sequence of Voronoi cells that intersect with the respective route; the link-route incidence matrices A and are formed by finding all routes whose distance from the sensor locations was less than some threshold empirically selected such that the maps matched well (≈ 10 meters tolerance for the PeMS loop sensor locations). All incidence matrices are saved in the `scipy.sparse` format.

7.5 NUMERICAL RESULTS

We validate our approach by measuring our accuracy in terms of the route flow estimates, denoted $\hat{x}$, given different scenarios. Note that we solve the reduced problem presented in (86) according to our algorithmic approach, and the solution in z-space is converted to $\tilde{x}$-space following the simple relation in (96), and is subsequently rescaled to $\hat{x} = \text{diag}(f^T U)\tilde{x}$. We additionally present our accuracy in terms of link flow estimates, to serve as a comparison to classical approaches to link flow estimation:

- Route flow error: $\epsilon_r = \left\|x^{true} - \hat{x}\right\|_1 / \left\|x^{true}\right\|_1$, with x^{true} the true route flow and $\hat{x}$ the estimated route flow. This relative error may be thought of as the percent error of flow allocation among all routes.
- Link flow error for observed links and all links, respectively:

$$\epsilon_l^{obs} = \frac{\left|\epsilon_{GEH}(b_i^{true}, \hat{b}_i) < 5, \forall i \in \left\{1, \cdots |\hat{b}|\right\}\right|}{|b^{true}|}, \text{ with } b^{true} = Ax^{true},\ \hat{b} = A\hat{x} \quad (112)$$

$$\epsilon_l^{full} = \frac{\left|\epsilon_{GEH}(v_i^{true}, \hat{v}_i) < 5, \forall i \in \{1, \cdots |\hat{v}|\}\right|}{|b^{true}|}, \text{ with } v^{true} = A^{full}x^{true},\ \hat{v} = A^{full}\hat{x} \quad (113)$$

where $|\cdot|$ denotes cardinality and $\epsilon_{GEH}(y, \hat{y}) = \sqrt{\frac{(y-\hat{y})^2}{0.5(y+\hat{y})}}$.

The $\epsilon_{GEH}(\cdot,\cdot)$ is called the GEH statistic, a heuristic formula commonly used to compare two sets of traffic volumes, e.g. for calibration of microsimulation models (Dowling et al., 2004, §5.6) and for validating hourly traffic flows (Transportation (WisDOT), 2013, §11-13). For an individual link, a GEH value of less than 5.0 is considered to be a good match. For a vector of links, a fraction $\epsilon_l \geqslant 0.85$ of good matches is considered a good match overall between modeled and observed volumes.

Note that our method always achieves the optimal link flow error $\epsilon_l^{obs} = 1$ for all networks, traffic assignment models, and sensor configurations, since our formulation minimizes the error to the observed link flows. However, we include this metric because it is a metric upon which we can validate real network settings, without relying on traffic simulators.

7.5.1 *Highway network*

Using the highway network scenarios in Section 7.4.2, we vary the link coverage from 10% to 100% and the cell density from 10 to 120 cells such that the proportions are $N^B : N^L : N^S :: 1 : 2 : 1$. We always observe the most congested links, and regions $\mathcal{S}$ contains only 1 region and is roughly downtown Los Angeles (see 7.4.1). We analyze how the relative error ϵ_r in route flows vary when sensors are more sparse. Since we choose random initial points in PTA (109) and in the solver (84) to generate synthetic route flows and compute the estimate respectively, and since the cellular network is sampled randomly, all the results presented in this section have been averaged over 100 trials. Figure 36 presents the numerical results when link flows and optionally OD demands are known, and cellular network data are assimilated into the model. That is, we solve and compare both the problem without OD flows in (84) and with OD flows in (91).

In the left column of Fig. 36, we consider Problem (84), where we compare the performance of route flow estimation via cellpath flow vs OD flow alone. As expected, the accuracy increases as we observe more links and/or more cells. We observe that in the regime where we have low link sensor coverage, having even very few cells outperforms OD demands. It is interesting that observing the additional links in the 40-70% region makes a significant difference in the accuracy for the OD demands. In this region (and beyond), it is possible to achieve an accuracy of 98.8% and 98.4% (for UE and SO, respectively) with a sufficient selection of cells. Finally, we note that 80 cells and 120 cells respectively achieves 96.0% and 98.7% accuracy for UE and 93.5% and 98.0% accuracy for SO, in the absence of OD demands and with only 10% links observed. This indicates that with a sufficient selection of cells, route flow estimation may be possible even without other kinds of sensor data.

In the middle column of Fig. 36, we consider Problem (91), which considers cellpath flows and OD demands together for estimation. As expected, adding information from any number of cells performs strictly better than having no cells. With both types of information, the highway network can achieve beyond 99.0% accuracy (with at least 70%

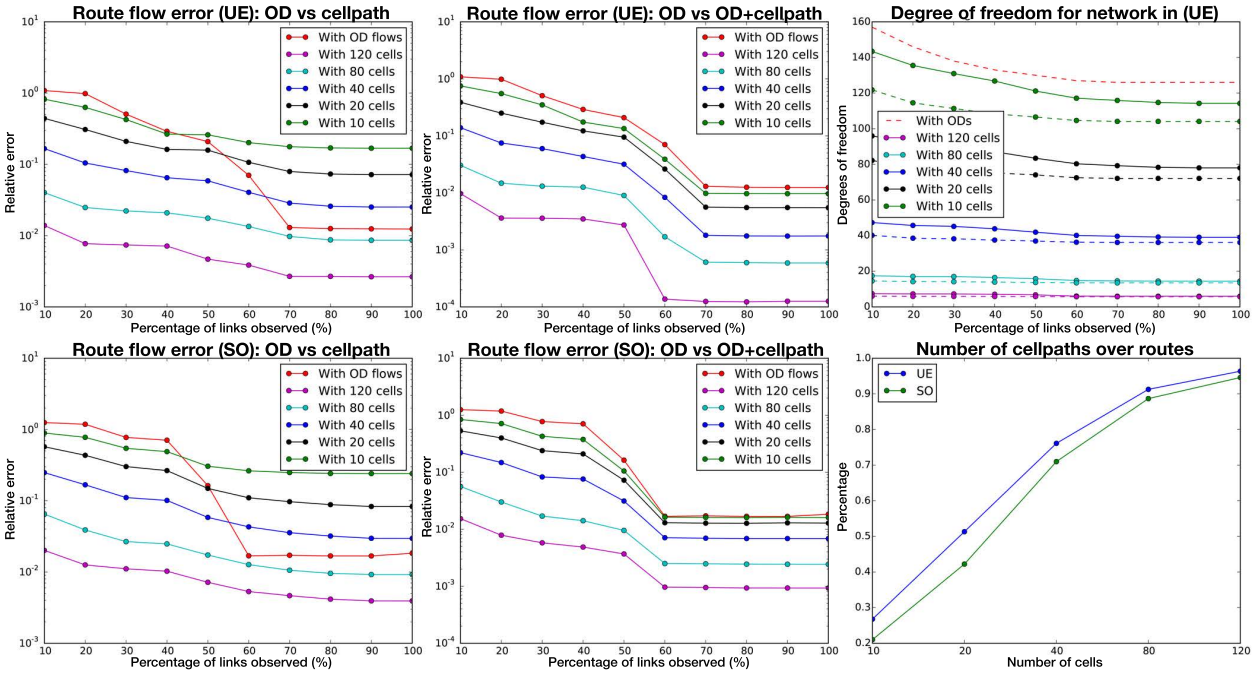


Figure 36: The six subfigures present the numerical results for the highway network. Top row, from the left to right: 1) the route flow error ϵ_r from OD demands (red curve) and cellpath flows only (other curves) with different link coverage values and different numbers of cells for the network in UE; 2) the route flow error ϵ_r from OD demands (red curve) and OD demands & cellpath flows (other curves) with different link coverage values and different numbers of cells for the network in UE; 3) lower bound on the degree of freedom for the program with OD demands (red curve), cellpath flows only (other curves, solid), and OD demands & cellpath flows (other curves, dotted) for the network in UE; Bottom row, left to right: 4) ϵ_r from OD demands (red curve) and cellpath flows only (other curves) for different configurations of the network in SO; 5) ϵ_r from OD demands (red curve) and OD demands & cellpath flows (other curves) for different configurations of the network in SO; 6) ratio of the number of observed cellpaths to the number of candidate routes. Best viewed in color.

links observed) and 98.4% accuracy (with at least 60% links observed), for UE and SO, respectively.

For both experiments above, the accuracy in the UE settings is generally better than that of the SO settings. In the bottom right subfigure of Fig. 36, we see that the ratio of the number of cellpaths observed to the number of routes used is greater for UE than SO for all the cell counts in our experimental setup; this is due to the tendency of agents to consider more routes in SO, and thus the same number of cells provides less resolution into the route choice of the agents. This provides evidence that the SO setting is more difficult for estimation.

The accuracy of the estimates is closely related to the degree of freedom of the solution in Problems (84) and (91). We compute an upper bound on the degree of freedom as $n - \text{rank}[A^T, U^T]$ and $n - \text{rank}[A^T, T^T, U^T]$, respectively, where n is the dimension of the problem. It is an upper bound because we do not consider how the non-negativity constraint $x \succeq 0$ limits the solution space. In the top right subfigure of Figure 36, we observe that in all cases, the use of cellpath information limits the degree of freedom moreso than with only OD information. As expected, the combined information from cellpaths and ODs (the dotted lines) limits the degree of freedom more than cellpath information alone (the solid lines). As the number of cells is increased, the degree of freedom tends towards zero, at which point we can fully recover the route flow. These numerical results confirms the utility of cellular network data for addressing the traditionally highly underdetermined route flow estimation problem.

7.5.2 *Full network, activity-based model*

Using the full network scenario in Section 7.4.3, we perform experiments using the actual locations of PeMS static highway count sensors on 1033 links (about 5% coverage). We vary cell density from 250 to 8000 cells such that the proportions are $N^B : N^L : N^S :: 3 : 1 : 16$.[11] The sub-regions $\mathcal{S}$ is given by the bounding boxes for the TAZ within the whole region. We analyze how the errors in route flows and link flows vary with the density of cell towers. Additionally, we study the effect of performing inference on only a subset of routes from our dataset.

Figure 37 presents the numerical results for Problem (84), where we compare the performance of route flow estimation via cellpath flow vs OD flow alone. To select a particular estimate from the solution space, we add an ℓ_2 regularization term to the objective. In our dataset, selecting the top 50 routes per OD pair was sufficient to account for 99.4% of trajectories; however, in general, the corresponding number of routes needed will vary based on the network, time of day, underlying driver behavior, etc. Thus, to represent these different settings, we present trade-off curves for accuracy when varying the number of routes from 3 up to 50. As expected, as more routes are considered by agents, the route flow accuracy ϵ_r declines, since the solution space (and its corresponding nullspace) grows. Fortunately, the accuracy increases with the number of cells. Thus, Figure 37 (top left) shows that the same level of accuracy may be attained when con-

11 The I-210 region is 688mi^2 and, with cell towers spaced $\frac{1}{4}$ to 2 miles apart for suburban and urban areas, a reasonable range of cell towers for modern urban areas is 180 to 5500. We select 1000 for our baseline model.

sidering different numbers of routes (per OD pair) by varying also the number of cells. Our method performs comparably for the morning (shown in Figure 37) and evening (not shown) rush hours, achieving 89.5% and 89.9% route flow accuracy respectively and well exceeding the GEH test (with 1000 cells and 50 routes per OD), indicating the versatility of our method for diverse traffic settings. Figure 8 (bottom right) shows that we always achieve the link flow error $\epsilon_l^{full} = 1$ on all links (including those not observed) for various link volume classes, indicating that our method is effective for estimating link flows on unobserved links in noiseless settings.

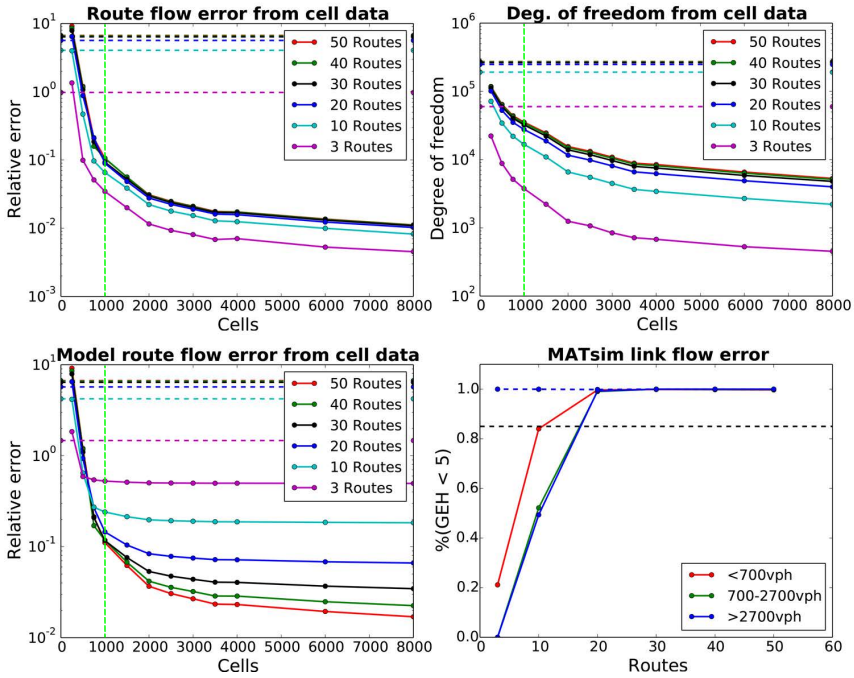


Figure 37: Full (highway and arterial) network experiment results, corresponding to the regularized solution for the morning commute (rush hour). The top row is specific to the noiseless setting; the bottom row includes experiments with modeling error (noise). The green dotted vertical line highlights the results for 1000 cells, which is a reasonable setting for urban areas today. Top left: Route flow ϵ_r from OD demands (dotted) and cellpath flows (solid) for varying cell counts. The different curves indicate the number of routes (per OD) considered; Top right: Approximate degrees of freedom for the program with OD demands (dotted) and cellpath flows (solid) for varying cell counts. Bottom left: Including modeling error, the route flow ϵ_r from OD demands (dotted line) and cellpath flows (curves) for varying cell counts. Bottom right: Link flow error evaluated on all links ϵ_l^{full} without model error (dotted) and with model error (solid), shown for different link flow volume classes for 1000 cells. Best viewed in color.

Similarly to the highway network experiment, the accuracy in the estimates is closely

related to the degree of freedom in Problem (84). For computational reasons, we compute an approximate measure of the degrees of freedom by $\text{nullity}(AN) \geqslant |z| - \text{rank}(A)$, using notation from (94). Although the problem remains underdetermined (based on equality constraints in the noiseless setting), the accuracy increases substantially as the degrees of freedom decreases (Figure 37, top right). In all scenarios except the lowest cell configuration (250 cells), we observe that performing inference using cellpath flows (compared to using OD flow information) greatly improves the estimates of route flow.

However, selecting the top routes between each OD pair for a real network relies on sophisticated models and techniques. Though this chapter focuses on the noiseless setting, here we present preliminary results for a noisy setting, motivated by situations where not all top routes may be curated. That is, using the same flow measurements b, f, etc., we now estimate a route flow vector $\bar{x} \in \mathbb{R}^{|\overline{\mathcal{R}}|}$, where $\overline{\mathcal{R}} \subseteq \mathcal{R}$ denotes the curated routes; then, we compute the validation metrics as before, taking the corresponding entries of x^{true}. We call *modeling error* the route flow that is not modeled by the curated subset of routes. Figure 37 (bottom subfigures) shows an experiment where we consider the performance of our method where, among the top 50 routes (per OD), we are only able to curate the top 3-50 routes, and we evaluate our method in the presence of this modeling error. We observe that curating 20-50 routes (per OD) is sufficient for achieving a low ($< 15\%$) route flow error and also sufficient for performing well on the GEH metric on all links. Our preliminary results show promise for estimating route and link flows with our approach despite the challenge of selecting all routes that agents may take.

7.6 CHAPTER SUMMARY

Our work demonstrates a data-driven method that is capable of estimating route-level flow accurately on a large scale network and is versatile to different vehicle behaviors. We address the traditionally highly underdetermined problem by introducing the concept of *cellpaths* for formalizing cellular network data as n-point network flows. This chapter introduces a projected gradient algorithm suitable for the route flow estimation problem, as well as the traffic assignment problem. We validate our approach on several networks of varying sizes and underlying traffic assignment models, showing that the incorporation of cellular network data dramatically improves estimates over the use of traditional data sources by providing flow information on (coarse) routes.

Our methodology is highly compatible with past and present work in the transportation community. As route flows contain strictly more information than link flows and OD flows, which underlie many transportation methods, the potential for accurate route

flow estimates in transportation applications is vast. Additionally, our solution method is shown to be compatible with related transportation problems, which may be combined for improved estimation.

Whereas work on traffic assignment, which models rather than estimates route flows, is critical for long-term land-use planning, strong model assumptions limit their application to short time-horizon applications. Taking a data-driven approach, our method enables new short time-horizon applications for the prediction and control of transportation such as route guidance, re-routing (e.g. minimizing effects of road closures, disasters, large events, etc.), demand prediction, and anomaly detection and analysis. Our framework aims to be widely deployable (wherever there is wide-spread cellular network coverage) and extendable, thereby providing a baseline estimator of the state of our current traffic networks, against which new controls and designs for intelligent transportation systems can compare.

The directions for future work are driven by our plans for integration with the decision support system for the I-210 corridor in California, US. We plan to analyze and improve the robustness of our model and methods in the presence of measurement error. Real loop sensors are notoriously noisy and a sizable fraction of them are offline at any given point. Since cellpath flow is not measured directly, but rather is inferred from cellular network data, it is prone to error from any inference procedure used. Fine-grained control applications will require even richer state estimates of the road network, for which we plan to extend our work to the dynamic setting. The full pipeline (summarized in Figure 30) will be implemented to perform large-scale route flow estimation using cellular network traces from AT&T and actual cell tower locations for the I-210 corridor in California, US.

Part III

SYSTEM DESIGN

8

HUMAN MOBILITY PREFERENCES

> How can you govern a country which has two hundred and forty-six varieties of cheese?
>
> Charles de Gaulle,
> Les mots du général, 1962

Part iii of this thesis looks beyond the internal dynamics of mixed autonomy systems, and the problems of control and estimation thereof. Mixed autonomy systems may be viewed as being embedded within another dynamical system, one which dictates the progression of the integration, adoption, and use of automation. This process, external to the mixed autonomy system, may induce substantial effects upon the system, both positive and negative. Thus, this part of the thesis explores the design of the system itself, to mitigate the potential effects of the external process on the system.

Mobility is embedded in an overall socioeconomic system, and one major anticipated long-term impact of automated vehicles is induced demand, in which more people travel in response to the newly available roadway capacity (enabled in Part i of the thesis). This additional demand on the mobility system may compromise the benefits in road velocity and throughput by resulting in elevated energy consumption. We start in Chapter 8 by empirically studying the dynamics of the overall socioeconomic system and in particular, its couplings with the mobility system. To this end, in collaboration with Microsoft, we investigate human mobility preferences based on a user study of employees at a major technology corporation. We identify ridesharing as a promising approach and give it treatment as a design paradigm for the mobility system itself, with the goal of mitigating the effects of induced demand by dramatically improving the throughput (supply) of the mobility system. We propose that, with lightly modified existing infrastructure, ridesharing has the potential to dramatically improve (nearly triple) the throughput of

the mobility system. In Chapter 9, we propose a number of algorithms to solve the allocation problem within the framework of combinatorial optimization. The structure of the ridesharing problem motivates the adaptation of clustering algorithms from machine learning for set partitioning in the combinatorial optimization framework, which is discussed in Chapter 10. The contributions of this chapter have implications for mobility system design, urban planning, and public policy, as well as scalable combinatorial optimization.

8.1 OVERVIEW

Human decision making is notoriously complex and challenging to model, and is the impetus for entire fields and subfields, including game theory, behavioral economics, psychology, sociology, and human-robot interaction (HRI). In the mobility context, we therefore focus on human decision making or preferences in the commuting context, as a representative travel demand activity, where we may expect both more data and regularity. We first introduce induced demand as a potential long-term impact of modifying the mobility system. We then present the findings of a user study on human mobility preferences, conducted at Microsoft with 232 participants. Finally, we provide recommendations to the design of a mobility system based on the mobility preferences to increase the system performance.

8.2 INDUCED DEMAND

"Today we are well underway to a solution of the traffic problem." This claim, made by Robert Moses in 1948, known as the "master builder" of mid-20th century New York City area, is as true today as it was then. Which is to say, not at all. In the middle of the last century, the preferred solution to "the traffic problem" was more cement: new highways, bridges, and lanes. Today, the sensible solution includes more sensors and better computers: highly automated vehicles that use existing roadways and roadway networks much more efficiently. This automation, we are told, will make vehicular congestion a "thing of the past." As in the past, however, this prediction presumes that more capacity necessarily means less congestion. Today's transportation planners recognize that the relationship between these two concepts is much more complex.

By reducing the time cost of driving or the capacity or throughput of roadways, automated vehicles may encourage greater travel and increase total vehicle miles traveled (VMT), which could lead to more congestion. This induced demand, or latent demand,

is an example of demand elasticity (Lee Jr et al., 1999) and can be explained by basic supply and demand theory from microeconomics. Travel demand may be determined by a combination of exogenous and endogenous factors (Lee Jr et al., 1999). Exogenous factors include land use, population, employment, and income; that is, factors external to the mobility system. Endogenous factors, on the other hand, are internal to the mobility system and may include factors such as the capacity and level of service of the transportation infrastructure and the price points of different modes of travel. Overall, then, travel demand is the result of a combination of both exogenous factors that determine the location of the demand curve, and endogenous factors that determine the price-volume point along the demand curve (where it meets the supply curve).

The effect of endogenous factors on the overall mobility system corresponds to the emerging claim that additional capacity stimulates corresponding increases in demand, hence induced demand; the inverse effect is called reduced demand. This concept embodies the "build it and they will come" idea, or a belief in the existence of "latent demand," which suggests that there are willing buyers who will express their demand for travel once the service is offered. In growing urban areas, the evidence from recent decades seemed to support this interpretation. For planning projections, the U.S. DOT Highway Economic Requirements System (HERS) estimates vehicle-demand price elasticity in the most likely scenarios to fall by -0.7 to -0.8 in the short run, and to fall about twice that in the long run, with a range of -1.0 to -2.0 (Lee Jr et al., 1999; Litman, 2017). This implies that as travel costs (time and expenses) reduce by 10%, travel is expected to increase: by 7-8% in the short run (time period over which exogenous demand factors remain fixed, estimated to be about one year) and by an additional 2-12% in the long run (time for exogenous characteristics to change, frequently assumed at 5-20 years) (Anderson et al., 2016).

In this part of the thesis, we focus on the endogenous factors. The extent to which they affect the system is jointly dictated by design or policy choices within the mobility system, that is whether to permit travel to grow or to suppress it, and its complex interface with human decision making. Thus, we study ridesharing as a design paradigm for the mobility system, with the goal of shaping the effects of endogenous factors by considering the effect of design choices on human decision making in mobility.

8.3 METHODOLOGY AND DATA COLLECTION

Human decision making is notoriously complex and challenging to model, and is the impetus for entire fields and subfields, including game theory, behavioral economics,

psychology, sociology, and human-robot interaction (HRI). In the mobility context, we therefore focus on human decision making or preferences in the commuting context, as a representative travel demand activity, where we may expect both more data and regularity. In fact, commuting more than doubles the population of the city of Redmond, Washington each weekday, which has a night-time population of 52K and a day-time population of 110K (Balk, 2013).

Study methodology. We designed a survey consisting of 35 detailed questions concerning mobility preferences, particularly in commuting and ridesharing contexts, including commuting modes and typical travel patterns (see full list of survey questions in Chapter 8.7). We conducted the user study on SurveyGizmo, and the estimated survey length was 18 minutes. The overall survey was divided into seven pages.

Participant demographics. In July 2015, we selected 1000 employees at the Microsoft Corporation uniformly at random, among those located in Redmond, Washington in the Seattle area. Overall, there were 232 respondents to the online study, which excludes a sizeable fraction of the original 1000 who were on vacation at the time or had since relocated to another office location. Another 66 started but did not complete the survey.

The user study of 232 participants consisted of 70% male and 29% female; 86% were full-time employees, 13% were contingent staff (including contractors). The largest represented age group is 41-45 years of age (17.4%), followed by 51 or older (16.5%), 31-35 (15.2%), and 36-40 (15.2%). No respondents were below 21, and the smallest represented age range is 21-25 (10.4%). Most respondents own a vehicle (90%). The mean commute time is 40 minutes, with a standard deviation of 23 minutes.

8.4 FINDINGS

The main findings of the study are summarized below:

Most drive alone. 68.3% of respondents drive alone to or from work. 24.6% of respondents were unwilling to drive others to work, even under "sufficient incentive." A small fraction of respondents rideshare to or from work (11.8%) and in all cases the carpool was with a family member. Other transportation options include a company shuttle, public transit, walking, biking, vanpool, ferry, motorcycle, running, scooter, and working from home. See Figure 38 for summary of typical commute modes.

Comfort from intra-organization and larger rideshare. Respondents felt more sharing a ride with someone else working at Microsoft *and* while with other people, as compared to by themselves or with someone not at Microsoft. While sharing a ride with someone

Tell us about your typical commute. Please check all that apply.

	I drive (alone)	I drive others (carpool)	I am a passenger	I commute with family members	Connector	Public transit	Walk	Bike	Vanpool	Other, e.g. ferry, jetpack
Morning	150 65.5%	13 5.7%	14 6.1%	27 11.8%	41 17.9%	32 14.0%	7 3.1%	17 7.4%	8 3.5%	12 5.2%
Evening	155 68.3%	13 5.7%	12 5.3%	20 8.8%	44 19.4%	32 14.1%	8 3.5%	20 8.8%	7 3.1%	13 5.7%

Figure 38: Typical commute modes. A majority of user study participants drive alone.

else working at Microsoft seems to increase respondent comfort levels, sharing a ride with multiple people seems to primarily decrease discomfort, rather than result in a comfortable experience. See Figure 39 for the levels of comfort anticipated by respondents in the different situations.

Backup option, time savings, and data protection most important features of a ridesharing system. Two features were observed as most important: 1) having a backup option, e.g. the system provides the user with an alternative transportation option like a taxi if a planned rideshare trip falls through (79.2%), 2) time savings, as compared to one's current commute (70.9%), and 3) protection of personal data (67.9%). See Figure 40 for a five-point Likert scale results on the preferences for ridesharing system features.

HOV and distance convenience most important to ridesharing drivers. The factors most important to the driver when ridesharing to/from work include: 1) the passengers live along their route to work (78.8%), 2) the passengers live close to them (69.7%), and 3) the number of passengers permit them to use the HOV lane (59.3%). These factors are significantly more important than other factors, including: 1) there being only one stop to pick up passengers, 2) there being a central and easy meeting location in the morning, or 3) that the passengers meet the driver at their home or office. See Figure 41 for a five-point Likert scale results on the pick-up and drop-off preferences of drivers when ridesharing.

It's not all about time savings. Ridesharing has a wide variety of benefits that are recognized by a sizeable fraction of the study participants. In particular, 23.8% of respondents would even consider carpooling even if the resulting trip takes longer than their current commute, when factoring potential social, economical, environmental, and other benefits.

How comfortable do you feel about traveling by car with someone you don't know, in the following scenarios?

	Very uncomfortable	Moderately uncomfortable	Slightly uncomfortable	Neutral	Slightly comfortable	Moderately comfortable	Very comfortable
Someone at Microsoft, and you are by yourself	9 4.4%	12 5.9%	33 16.3%	31 15.3%	25 12.3%	47 23.2%	46 22.7%
Someone at Microsoft, with other people	7 3.5%	8 4.0%	20 9.9%	41 20.3%	26 12.9%	48 23.8%	52 25.7%
Someone not at Microsoft, and you are by yourself	16 7.9%	32 15.8%	46 22.7%	33 16.3%	25 12.3%	27 13.3%	24 11.8%
Someone not at Microsoft, with other people	13 6.4%	22 10.9%	37 18.3%	37 18.3%	32 15.8%	35 17.3%	26 12.9%

Figure 39: Impact of intra-organization and rideshare size on comfort. On a seven-point Likert scale, respondents compared their comfort level for ridesharing on two factors: 1) riding with someone working at Microsoft (intra-organization rideshare) versus not, and 2) riding with someone else alone versus with other people. Respondents greatly preferred intra-organizational carpools. Riding with other people shifted the discomfort levels but did not noticeably improve the comfort levels.

21.5% of respondents would consider carpooling if their trip times are more consistent (but not necessarily longer or shorter), and 35% of participants would consider carpooling if they save time on average over multiple trips (but not with each trip).

Bimodal splits on preferences. A number of factors observed bimodal splits, including the preference to spend one's commute time socializing (32% for, 37% against), driving (23% for, 44% against), getting to know people better (26.7% for, 42.2% against), or working (42.3% for, 31.7% against). Note that a "neutral" category was excluded from these

Please state the importance of the following characteristics / features of a ridesharing system to you.

	Very unimportant	Unimportant	Neutral	Important	Very important
Time savings, as compared to your current commute	1 0.5%	9 4.2%	31 14.4%	92 42.8%	82 38.1%
Money savings, as compared to your current commute	10 4.7%	19 8.9%	66 30.8%	86 40.2%	33 15.4%
Ease of expressing preferences and constraints	3 1.4%	11 5.2%	65 30.7%	88 41.5%	45 21.2%
Automated matching of ridemates based on personal preferences	8 3.8%	18 8.5%	93 43.9%	80 37.7%	13 6.1%
Clear pricing structure for fees	5 2.4%	7 3.3%	54 25.5%	106 50.0%	40 18.9%
Financial incentives (e.g. lottery, direct payment, gas money, coupons, toll money)	9 4.2%	13 6.1%	64 30.2%	87 41.0%	39 18.4%
Fairness (compensation for drivers, taking turns driving, etc.)	8 3.8%	7 3.3%	75 35.4%	95 44.8%	27 12.7%
Protection of personal data	7 3.3%	11 5.2%	50 23.6%	80 37.7%	64 30.2%
Reserved parking (for carpoolers)	9 4.2%	11 5.2%	64 30.2%	96 45.3%	32 15.1%
Backup option (in case a planned trip falls through, the system will provide an alternative transportation option, e.g. taxi)	5 2.4%	5 2.4%	34 16.1%	78 37.0%	89 42.2%
Real-time rideshare (ability to request a ride whenever or "snooze" an upcoming ride)	5 2.4%	7 3.3%	48 22.9%	96 45.7%	54 25.7%

Figure 40: Importance of features of ridesharing systems. Respondents rated as most important the features of providing a backup option, time savings, and protection of personal data.

calculations, so the percentages may not add up to 100%. These bimodal splits indicate multiple distinct and complex subpopulations within the overall commuting population, each of which has distinct preferences, needs, and likely requirements for a mobility

When carpooling to/from work, please qualify the importance of these factors when you are the driver.

	Very unimportant	Unimportant	Neutral	Important	Very important	Not applicable
The passengers live close to me.	3 1.4%	6 2.7%	24 10.9%	55 24.9%	99 44.8%	34 15.4%
The passengers live along my route to work.	2 0.9%	0 0.0%	9 4.1%	66 29.9%	107 48.4%	37 16.7%
There is only one stop to pick up passengers.	2 0.9%	9 4.1%	44 20.1%	79 36.1%	48 21.9%	37 16.9%
There is a central and easy meeting location in the morning, e.g. a Park & Ride.	1 0.5%	10 4.5%	34 15.5%	83 37.7%	54 24.5%	38 17.3%
The passengers meet me at my home in the morning / travel from my home in the evening.	12 5.5%	42 19.2%	64 29.2%	36 16.4%	26 11.9%	39 17.8%
The passengers can be dropped off at my office building in the morning (as opposed to theirs).	1 0.5%	12 5.5%	55 25.0%	79 35.9%	37 16.8%	36 16.4%
The passengers meet me at my office building in the evening.	1 0.5%	13 5.9%	51 23.3%	76 34.7%	42 19.2%	36 16.4%
The number of passengers permit me to use the HOV lane.	4 1.9%	9 4.2%	32 14.8%	46 21.3%	82 38.0%	43 19.9%

Figure 41: Importance of pick-up and drop-off considerations for rideshare drivers. On a five-point Likert scale, respondents rated as "very important" that as a rideshare driver, the factors of the passengers living close to them, the passengers living along their route to work, and the number of passengers permitting them to use the HOV lane.

system to suit their needs. See Figure 42 for a complete list of activities and preferences for how to spend one's commute time.

Morning and evenings trips are different. Morning and evening trips vary dramatically in terms of how respondents choose when to commute. Morning trips are largely governed by the time of the first meeting (30.7%) and the preference to keep a regular schedule (25.5%). Evening trips are largely dictated by traffic considerations (31.0%) and external constraints such as child pick-ups and social outings (26.6%). See Figure 43

I would like to spend my commute time...

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree	Not applicable
Driving	56 25.2%	42 18.9%	67 30.2%	37 16.7%	14 6.3%	6 2.7%
Relaxing	9 4.0%	5 2.2%	31 13.7%	106 46.7%	72 31.7%	4 1.8%
Reading / thinking	8 3.5%	8 3.5%	34 15.0%	85 37.6%	87 38.5%	4 1.8%
Working	22 9.7%	50 22.0%	55 24.2%	65 28.6%	31 13.7%	4 1.8%
Socializing	31 13.8%	53 23.6%	63 28.0%	59 26.2%	14 6.2%	5 2.2%
Meeting new people	45 19.9%	65 28.8%	69 30.5%	32 14.2%	11 4.9%	4 1.8%
Getting to know people better	37 16.4%	58 25.8%	66 29.3%	49 21.8%	11 4.9%	4 1.8%
Taking care of errands	20 8.8%	26 11.5%	60 26.5%	93 41.2%	24 10.6%	3 1.3%

Figure 42: Commute time activity preferences. A few activities are strongly preferred by most respondents, including relaxing and reading or thinking. A number of activities, however, have bimodal preferences, including driving, working, socializing, and getting to know people better. Respondents largely did not prefer to spend their commute time meeting new people.

These findings on human mobility preferences form the basis for considering ridesharing as a design paradigm for the mobility system, as detailed next.

8.5 Recommendations for Ridesharing Systems

Time is more important than money. Although many ridesharing services feature lottery, gas coupons, and other monetary incentives, time savings are significantly more important to users (see Figure 40). Time savings can come in the form of reduced variance or reduced mean travel times, such as through parking benefits.

What's the most important factor for when you arrive/depart at work?

	Time of first/last meeting	I prefer to keep a regular schedule	Traffic considerations	External constraints (e.g. kid's schedule, social outings)	No regular pattern	Other
Arrive	71 30.7%	59 25.5%	42 18.2%	38 16.5%	13 5.6%	8 3.5%
Depart	35 15.3%	30 13.1%	71 31.0%	61 26.6%	21 9.2%	11 4.8%

Figure 43: Factors for when to commute. Factors important to respondents vary greatly, between morning and evening commutes. Morning trips are largely governed by the time of the first meeting and the preference to keep a regular schedule. Evening trips are largely dictated by traffic considerations and external constraints such as child pick-ups and social outings.

Utilize high-occupany vehicle (HOV) lanes. HOV lanes are compelling because 1) they have the potential to counter social barriers against carpooling, such as comfort (see Figure 39), trust and personal commute preferences, through permitting the use of lower latency lanes (Parsons Transportation Group, Inc., 2002; U.S. Dept. of Transportation, Federal Highway Administration, 1977), 2) they provide time savings and lower variance commutes, which are a critically needed incentive for ridesharing (see Figure 40), and 3) they provide a relatively easy to implement scheme, through the re-purposing of existing infrastructure. Moreover, the discomfort level can be adjusted by converting a general purpose lane or by restricting 2+ HOV lanes to 3+ (see Figure 39). In several dense urban areas, such as the Puget Sound area in Washington and the San Francisco Bay Area, these 3+ HOV lanes in fact already exist and are in use. Chapter 9 will further develop, formalize, analyze, and propose computational solutions to this aspect of the mobility system. Chapter 10 will then relax the resulting NP-complete problem and propose a clustering-based algorithm, which converges in expectation to within a bound of the optimal solution.

Reduce trip uncertainty. Driving alone corresponds to the lowest trip uncertainty among commonly available modes of transportation. Respondents also identified having a backup option as the most important feature for a ridesharing system, for situations where the planned trip falls through.

Reduce cognitive load. A vast majority of respondents (94.8%) indicated that the most important factor in choosing their mode of commute is convenience, in terms of ease,

flexibility, and "not needing to think about it." Convenience was more important than traffic considerations (82.3%), family constraints (47.6%), and monetary savings (46.2%). A ridesharing system should limit its cognitive load on users and try to retain ease of use and flexibility.

Focus on a subpopulation. This user study demonstrated that there are distinct subpopulations within the overall commuting population, each with different needs and preferences. While 24.6% of the respondents are unwilling to rideshare even under ideal circumstances, 35.1% believe that a well-designed ridesharing program could improve their current commute. A ridesharing system should focus its efforts on a promising subpopulation, for instance a population which is more elastic on their mode of commute, based on time savings, money savings, social benefits, etc.

8.6 CHAPTER SUMMARY

This chapter considered human decision making in the commuting context, as a way to understand both soft human factors, such as comfort and inconvenience, and hard factors, such as time and monetary cost. It is crucial to understand the rich spectrum of human preferences, as evidenced by decades of lackluster attempts at large-scale ridesharing. These nuanced human preferences provide several recommendations for the design of ridesharing systems moving forward, including focusing on time, trip uncertainty, cognitive load, and subpopulations. Future work should include a more rigorous study and identification of the major subpopulations of users within a mobility system. Additionally, morning and evening commutes vary in significant ways, and thus further study can investigate the design of one-way rideshare systems that would encourage users in both settings. It is also important to recognize that mobility preferences may vary in different contexts, such as at companies in different sectors or in universities.

8.7 MOBILITY PREFERENCE SURVEY QUESTIONS

Following are the survey introductory text and questions.

Survey introductory text.

> Welcome to the MSR Commuting Survey!
>
> As part of a study conducted by Microsoft Research to see how much better we can make your everyday life, we are interested in your commuting

patterns, the factors which influence your commute, and the opportunities to improve.

As part of this study, we will be asking you for personal information (e.g. email address, age). Your personal data will not be shared with anyone outside of the research team. Aggregate information may be shared with university collaborators outside of Microsoft and for publication. Please contact xxxxxx@microsoft.com if you have any questions about this research project.

Estimated survey length: 18 minutes. (You can save and continue later at the top of the next page.)

Now, let's start. Please tell us a little about you and your life at Microsoft.

Survey questions.

1. Team / organization (optional)
2. Age
3. Gender
4. In which building do you primarily work (e.g. 88, Studio A)?
5. What is your employment status?
6. Do you use the Outlook app on your phone?
7. What is your vehicle capacity (including the driver)? (Enter 0 if you don't have a car.)
8. What's the maximum number of people you are willing to drive to/from work, with sufficient incentive (e.g. splitting tolls, gas reimbursement, lottery, social benefits)?
9. For a typical day, when do you arrive at work at the beginning of your day?
10. For a typical day, when do you depart from work at the end of your day?
11. How long is your commute on average (minutes), departing from work?
12. During a typical week, how variable are your...
    - Arrival times at the start of your day?
    - Departure times at the end of your day?
13. When your arrival/departure time does vary, how much does it usually vary?
    - Arrival time
    - Departure time
14. Tell us about your typical commute. Please check all that apply.
    - I drive (alone)
    - I drive others (carpool)

    - I am a passenger
    - I commute with family members
    - Connector
    - Public transit
    - Walk
    - Bike
    - Vanpool
    - Other, e.g. ferry, jetpack
15. If "other", please specify:
16. Please qualify the following statements.
    - I am happy with my current commute.
    - I enjoy driving.
17. What's the most important factor for when you arrive/depart at work?
    - Time of first/last meeting
    - I prefer to keep a regular schedule
    - Traffic considerations
    - External constraints (e.g. kid's schedule, social outings)
    - No regular pattern
    - Other
18. If "other," please specify:
19. How often do you make other stops along the way to/from work (e.g. pickup/-dropoff someone, social outings, errands)?
20. How often do you carpool to/from work?
21. In choosing your mode of commute, how important are the following factors to you?
    - Convenience (e.g. easiest, flexibility, don't need to think about it)
    - Traffic
    - Environmental reasons (e.g. greenhouse gases, polar bears)
    - Monetary savings (e.g. tolls, gas, vehicle maintenance, parking)
    - External incentives (e.g. HOV lanes, lottery, special parking)
    - Family constraints (e.g. pickup/dropoff family member)
    - Social reasons (e.g. family, friends, significant other)
    - Health reasons (e.g. walk, run, bike, to not sit all the time)
22. Anything else we should know about your commute? Other factors for your choice of mode of commute, anything out of the ordinary about your commute, e.g. last week not representative, high irregularity, high variance in commute times, asym-

metry in your arrival/departure commutes? (optional)

23. I would like to spend my commute time...
    - Driving
    - Relaxing
    - Reading / thinking
    - Working
    - Socializing
    - Meeting new people
    - Getting to know people better
    - Taking care of errands
24. Please state your preferences.
    - I can imagine an efficient and well-designed carpool program to be better than my current commute.
    - If carpooling, I would be willing to drive.
    - If carpooling, I would be willing to be a passenger.
25. Would you consider carpooling, even if the resulting trip may take longer than your current commute, considering potential social, economical, environmental, etc. benefits? Please check all that resonate with you.
26. When sharing a ride to/from work, I would be willing to share a ride with...
    - People who work at Microsoft
    - People who don't work at Microsoft
    - People in my organization
    - People on my team
    - My management chain (e.g. manager, skip, skip skip)
    - People of the opposite gender
    - People of a similar age
    - People of a different age group
    - People who drive safely and carefully
    - Friends
    - Friends of friends
    - Family
27. How comfortable do you feel about traveling by car with someone you don't know, in the following scenarios?
    - Someone at Microsoft, and you are by yourself
    - Someone at Microsoft, with other people
    - Someone not at Microsoft, and you are by yourself

- Someone not at Microsoft, with other people

28. On a typical day, by how much would you be willing (or able) to modify your usual work arrival/departure times? Please check all that apply.
29. When carpooling to/from work, please qualify the importance of these factors when you are the driver...
    - The passengers live close to me.
    - The passengers live along my route to work.
    - There is only one stop to pick up passengers.
    - There is a central and easy meeting location in the morning, e.g. a Park & Ride.
    - The passengers meet me at my home in the morning / travel from my home in the evening.
    - The passengers can be dropped off at my office building in the morning (as opposed to theirs).
    - The passengers meet me at my office building in the evening.
    - The number of passengers permit me to use the HOV lane.
30. As the driver, how much time would you be willing to spend in gathering the passengers before/after work? This is the time you are willing to deviate from your usual route, wait for people to come out of their homes/offices, or wait at the PR, etc.
31. How regularly would you like your ridemates to change?
32. If "other," please specify:
33. Please state the importance of the following characteristics / features of a rideshar-ing system to you.
    - Time savings, as compared to your current commute
    - Money savings, as compared to your current commute
    - Ease of expressing preferences and constraints
    - Automated matching of ridemates based on personal preferences
    - Clear pricing structure for fees
    - Financial incentives (e.g. lottery, direct payment, gas money, coupons, toll money)
    - Fairness (compensation for drivers, taking turns driving, etc.)
    - Protection of personal data
    - Reserved parking (for carpoolers)
    - Backup option (in case a planned trip falls through, the system will provide an alternative transportation option, e.g. taxi)

- Real-time rideshare (ability to request a ride whenever or "snooze" an upcoming ride)

34. If you do not carpool regularly, why don't you? What are your concerns about ridesharing? Have you tried carpooling in the past? What would encourage you to try carpooling (again)? (optional) If you do carpool, what aspects of carpooling make it worthwhile to you? (optional)
35. What are some other features / characteristics that you think would be important for an effective ridesharing system? (optional)

9

OPTIMIZING THE DIAMOND LANE: COMPLEXITY AND ALGORITHMS FOR RIDESHARING

Move fast with stable infrastructure.

Mark Zuckerberg,
Founder and CEO, Facebook, 2014

Ridesharing (or carpooling) has been long deemed a promising approach to improved transportation infrastructure operation. However, there are several reasons why ridesharing is still not the preferred mode of commute in the United States: first, complex human factors, including trust, compatibility, and not having the right incentive structures, discourage the sharing of rides; second, algorithmic and technical barriers inhibit the development of online services for matching riders. In Chapter 8, we substantiated the human factors concerning ridesharing and subsequently proposed that high-occupancy vehicle (HOV) lanes which permit vehicles that hold three or more people (HOV3+) have the potential to simultaneously decrease trust concerns and dramatically reduce travel times, thereby providing a promising avenue for addressing both types of issues.

The goal of this chapter is to present algorithms which provide a basis for optimizing the use of HOV3+ lanes. We formalize the HOV3 Carpool problem, show that it is NP-Complete, and provide a brief survey of related complexity results in ridesharing. We then formally pose the HOV3- Carpool problem, relaxing the strict capacity constraint of size three. Unlike the previous problem, this new problem is amenable to a wide range of common exact and heuristic methods for solving the problem of finding globally optimal carpool groups (in terms of total vehicle distance) that may utilize these HOV lanes. We present local search, integer programming, and dynamic programming methods; our local search methods include sampling-based (hill-climbing and simulated annealing), classical neighborhood search, and a hybrid random neighborhood search. This chapter assesses the methods numerically in terms of scalability and convergence to a naive

"perfect carpooling" lower bound. We additionally assess the impact of domain-specific warm-starting strategies to the convergence of the iterative methods. Our numerical experiments study synthetic agents set in both Euclidean plane and San Francisco Bay Area settings and highlight that the hill climbing local search method scales up to 100K agents, thereby improving upon related previous work (which studies up to 1000 agents), and numerically converge to 1.1 of the lower bound. All other benchmark methods were shown to face scaling or convergence limitations.

One primary aim of this chapter is to introduce methods which are practical for implementation in real-world settings; as such, this chapter successfully studies the carpool problem in the context of true road networks and large populations. Importantly, our focus on the HOV3+ setting is specifically to counter known social and implementation barriers to improving transportation efficiency. Our work additionally shows that despite the theoretical challenges of the problem, heuristic algorithms can still efficiently and effectively solve the problem. The hill climbing method studied in this chapter is not only easy to implement, computationally efficient, and converges quickly, it is also flexible to rich classes of costs not studied in this chapter, including travel time, inconvenience, and social preferences. Additional constraints, such as hard preferences and required stops, may also be encoded in this problem formulation. Some types of constraints, however, such as requiring exactly three passengers per vehicle, may require further method design.

9.1 OVERVIEW

General Background. Transportation-related costs of air pollution, greenhouse gas emissions, noise, delay from traffic congestion, and losses and injury from collisions are estimated to be approaching $1.1 trillion annually in the US alone (Blincoe et al., 2015; Grant et al., 2000; Schrank et al., 2015). With the introduction of the sharing economy, there is renewed interest and energy in improving transportation infrastructure utilization and operation through the design of automation (Hoshino et al., 2007), optimization (Miao et al., 2016), and logistics systems (Venkatadri et al., 2016). Mobility-on-demand services and public mass transit are promising but cannot address a large segment of commuting. Today, 50% of Americans live in suburban regions, where current public transit services and mobility-on-demand services are not economically feasible. These regions are major contributors to the cost of transportation. For example, it has been estimated that nearly 20% of suburban household GHG emissions in the US are linked to transportation (Jones and Kammen, 2014). We investigate ridesharing (Chan and S. A. Shaheen, 2012; Furuhata

et al., 2013) as a promising path forward for reducing the multiple costs associated with transportation. Ridesharing has the potential to significantly reduce congestion and air pollution and to make cities more social and pleasant places (Selker and Saphir, 2010).

Goals and motivation. The main goal of the present chapter is studying the carpooling incentive of high-occupancy-vehicle (HOV) lanes, and how such incentives can be algorithmically incorporated into ridesharing systems to achieve higher transportation system efficiency, thereby permitting more reliable transportation estimates and operation. As seen in Chapter 8, HOV lanes are compelling because 1) they have the potential to counter social barriers against carpooling, such as comfort, trust and personal commute preferences, through permitting the use of lower latency lanes, 2) they provide time savings and lower variance commutes, which are a critically needed incentive for ridesharing, and 3) they provide a relatively easy to implement scheme, through the re-purposing of existing infrastructure.

Commonly, HOV lanes may have restrictions which specify how many people must be in a vehicle in order to use the lane. In most locations, HOV lanes require at least two occupants. Increasingly, in bottleneck areas such as metropolitan areas, including the San Francisco Bay Area and Seattle, WA, HOV lanes increasingly require three passengers in order to maintain high throughput (Department of Transportation Division of Traffic Operations, 2016; Washington State Department of Transportation, 2017). Optimizing the use of the existing road network is a highly complex problem, but it starts with optimizing the use of the HOV lanes, which is the topic of this chapter, and we will pose this as the HOV3 or HOVk carpool problem. The problem of having exactly two occupants in each vehicle reduces to bipartite graph matching, for which efficient algorithms exist (Garey and Johnson, 1979). Remarkably, the problem of having exactly three occupants in each vehicle is NP-Complete, so it is very unlikely that an efficient algorithm can solve the problem exactly. In this chapter, we prove this complexity result and present algorithms to solve the problem approximately. This crossing from a tractable (HOV2) to intractable (HOV3, formally defined in Section 9.3) problem, if not addressed, would greatly impede the scalability of any practical rideshare matchup system.

We cast the ridesharing problem with HOV lanes into a general combinatorial optimization framework (Papadimitriou and Steiglitz, 1998; Wolsey and Nemhauser, 2014), which enables us to devise and invoke several types of solution methods and assess their accuracy and scalability. In particular, we formulate our problem as a set partitioning problem, an integer program, and as a dynamic programming recurrence. We study local search, integer programming, and dynamic programming methods to the problem.

We call the problem of finding optimal size-three groups for carpooling the *HOV3*

Carpool problem. We show that this problem is NP-Complete and, in addition, is difficult to approximate or solve iteratively. Thus, we formulate a relaxed problem, which we call the *HOV3- Carpool problem* (note the minus sign). This problem relaxes the size-three carpool groups to allow *up to* size three carpools. The relaxed form is amenable to iterative methods such as local search. We refer to this relaxed ridesharing problem as the carpool problem throughout the chapter. We validate our methods numerically through workloads of agents (simulated participants) generated from census data in the San Francisco Bay Area in California.

Contributions. The main contributions of this chapter are:

- We study the challenges of optimizing for high-occupancy vehicle (HOV) lane usage, then find and present a relaxation of the problem with properties amenable to highly scalable and powerful heuristic algorithms, thereby transforming a challenging combinatorial problem into a practical solution for ridesharing optimization.
- We formulate and present the *HOV3 Carpool* and *HOV3- Carpool* problems, which specifically integrate the carpooling incentive of 3+ HOV lanes.
- We prove that the *HOV3 Carpool* problem is NP-Complete by reduction from Exact Cover by 3-Set. We then demonstrate that the relaxed *HOV3- Carpool* problem is amenable to iterative methods such as local search, due to the existence of a polynomial algorithm for finding feasible solutions. The complexity results of this HOV carpool problems are summarized in Table 7 in Section 9.3.3.
- We study four local search methods for the relaxed *HOV3- Carpool problem*: hill climbing, simulated annealing, classical local search, and local search with random neighborhood. The first two are sampling-based local search methods; the third is a classical local search approach; the fourth combines sampling with the classical local search approach.
- To demonstrate the limitations of classical and general methods for the carpool problem, we present for comparison an integer program (IP) formulation and a dynamic programming formulation for the *HOV3- Carpool problem*. Our IP formulation utilizes the problem structure to yield a concise representation.
- We empirically compare the above six methods on Euclidean world and Bay Area synthetic workloads (simulated participants) of up to 100K agents. We show that the hill climbing method outperforms by far the rest in terms of scale, computational runtime, and convergence to a naive lower bound. For the largest problem size of 100K, the hill climbing method converges to a ratio close to 1 to the lower bound. Interestingly, we additionally find that a greedy initialization performs just as well as other heuristic warm-starts, which may incorporate more domain knowl-

edge.

Related work. Many variants of the carpooling problem have been studied, for instance focusing on maximizing the number of participants (Armant et al., 2015) or minimizing the total distance or time (Herbawi and Weber, 2012); for more variants, we refer the reader to excellent surveys on ridesharing by Agatz et al. (2012) and Furuhata et al. (2013). Although there are numerous works studying the HOV or high-occupancy toll (HOT) lanes from the perspective of pricing (Yang and H.-J. Huang, 1999; Konishi and Mun, 2010), we are not aware of any work that explicitly optimizes for its utilization.

Our work temporarily side-steps the pricing aspect of the carpooling problem to focus on the optimization aspect; however, since we optimize for the global cost, our work is compatible with the incentive-compatible pricing mechanism introduced in Kamar and Horvitz (2009), based on the Vickery auction (Vickrey, 1961).

General optimization techniques from operations research are very natural approaches to this problem. A closely related work by Armant and Brown (2014) studies MILP and CP methods for a variant of the global optimization problem where passengers travel to the pickup locations that are along the path of the driver. Our work also compares against some of these underlying techniques, but we show that sampling methods can do much better in our setting. This pickup restriction tremendously simplifies the problem but may provide a less seamless experience for passengers. We provide a formulation which permits driver deviations for passenger pickups. Perhaps surprisingly, however, the restriction to fixed routes results in no easier of a problem in terms of theoretical complexity. Even slight variants in the carpooling problem can lead to large differences in the theoretical and numerical results. As such, we compare our work against several common underlying techniques of related work, instead of comparing against related problem formulations. An early version of our work is present in Wu et al. (2016b).

The remainder of this chapter is organized as follows: In Section 9.2, we survey known complexity results of ridesharing problems. We then formally pose the carpooling problem in Section 9.3, prove NP-Completeness, and present a relaxation. In Section 9.4, we present three ways to formulate the problem; Section 9.5 details the methods we study for each formulation. In Section 9.6, different warm-starts for the carpooling problem are proposed. Section 9.7 poses the numerical implementation and setup, and Section 9.8 summarizes the numerical findings. We conclude with a summary, additional challenges, and next steps in Section 9.9.

9.2 SURVEY OF COMPLEXITY RESULTS IN RIDESHARING

Specific ridesharing problems take on many forms, for example having varying problem setups in terms of objectives, costs, and constraints. For instance, some studies investigate maxmizing participation whereas other studies investigate minimizing total travel time or distance (see Agatz et al. (2012) for a review and overview on the space of ridesharing problems and optimization approaches). However, almost all ridesharing problems studied to date are NP-Complete (decision versions) or NP-Hard (optimization versions), with a few exceptions.

Several works which address the computational complexity of ridesharing, study a setting where the drivers take a fixed route; that is, the riders accommodate the driver, either by meeting up at the driver's origin or meeting along the driver's route (Simonin and O'Sullivan, 2014; Kutiel, 2016; Ma and Wolfson, 2013). In contrast, in our work the driver's route is determined by the optimal route considering the members of its carpool. The general forms of these problems, which consider capacity or both capacity and delay constraints (Ma and Wolfson, 2013) or desire a balance of users among vehicles (Simonin and O'Sullivan, 2014), are NP-Complete. Some restricted settings have been found to be polynomial-time solvable. Ma and Wolfson (2013) show that its non-capacitated or capacity-two versions are polynomial-time solvable. Kutiel (2016) shows that when the set of drivers and riders is given in advance, it is polynomial-time solvable. Simonin and O'Sullivan (2014) show that in the case of optimizing for the maximal number of participating ridesharing users (irrespective of cost) in a fixed capacity setting, a polynomial time algorithm is admissible. We do not consider restrictions such as pre-assigned roles, size-two carpools, or unlimited capacity carpools and focus on the harder problem.

Due to the computational complexity of ridesharing, numerical studies have largely focused on small problem sizes (up to about 1000 agents) (Armant et al., 2015) or these special polynomial-solvable cases (Hartman et al., 2014), which are strong technical limitations for providing carpooling at scale as a suitable transportation alternative. While it is unrealistic to provide scalable methods for the general problem, one of the primary goals of our work is to provide highly scalable methods for a problem that is realistic; thus, we focus on devising scalable methods for our specific HOV carpool problem.

9.3 THE CARPOOL PROBLEM

A natural way to optimize for HOV3 lane use is to optimize for the best configuration of carpools of size three. However, we prove in this section by reduction that this problem

is NP-Hard. Subsequently, we study a relaxed version where we optimize for carpools of size up to (and including) three. The relaxed problem lends itself more easily to approximation methods such as local search. The complexity results of these HOV carpool problems are summarized in Table 7.

9.3.1 *Optimizing usage of the HOV3+ lane*

We study the carpool incentive of permitting usage of specific lower latency lanes (often on the freeway) which require at least three occupants in a vehicle. Under such a policy, there is little incentive for vehicles to have more than three occupants, since each additional passenger adds additional divergence costs to the carpool group. Consequently, we reason that the best reasonably achievable solution would optimize for three occupants per vehicle.

Now we specialize to the *HOV3 carpool problem*, in which the goal is to find size-three ridesharing groups that minimize the overall cost of the system. Additionally, we consider time constraints.

Definition 1 (HOV3 carpool problem)**.** *Let $\mathcal{U}$ be the set of agents. Each agent is endowed with a home and a work location $(p_h^u, p_w^u), \forall u \in \mathcal{U}$ as well as time constraints $T^u = (t_0^u, t_1^u, \dots), \forall u \in \mathcal{U}$ with $t_i^u = [t_{i,s}^u, t_{i,e}^u]$ for each time window (interval). The subscript s denotes the start time of a time window; the subscript e denotes the end time. The collection of feasible subsets of the universe (rideshare groups) $\mathcal{S}$ consists of groups of size 3 such that the agents have a non-empty common time window. That is,*

$$\mathcal{S} := \left\{ S : S \in 2^{\mathcal{U}}, |S| = 3, \bigcap_{u \in S} T^u \neq \emptyset \right\}. \tag{114}$$

The overall objective of the HOV3 carpool problem is to find the optimal partition with size-three groups of agents; that is, to find a subset $\mathcal{R} \subseteq \mathcal{S}$ such that

- *$\mathcal{R}$ minimizes some given cost function $C : 2^{\mathcal{S}} \to \mathbb{R}$.*
- *$\mathcal{R}$ forms a partition of $\mathcal{U}$, as no agent may participate simultaneously in multiple ridesharing groups.*

The specific cost function we study is given in Equation (118). The *search* version (as opposed to the optimization version) of this problem, which excludes costs, is already NP-complete, as we will prove next in Section 9.3.2. Note that, in general, the time constraints can be used to encode other types of constraints as well, such as age restrictions or arbitrary agent preferences, which are also important for a practical rideshare

Problem	Version	Complexity
HOV3 carpool problem	Optimization	NP-Hard
HOV3 carpool problem	Search	NP-complete
HOV3 carpool problem	Feasible solution	NP-complete
HOV3- carpool problem	Optimization	N/A
HOV3- carpool problem	Search	N/A
HOV3- carpool problem	**Feasible solution**	**P**
HOVk carpool problem	Optimization	NP-Hard

Table 7: This table summarizes the complexity analysis of the HOV carpool problems. The original HOV3 carpool problem is NP-Hard. One of the key contributions of this chapter is finding that its relaxation, the HOV3- carpool problem, although perhaps still NP-Hard, offers a polynomial-time solution for finding a feasible solution, which lends itself to the rich and powerful family of local search methods.

matchup system. For clarity and concreteness, we represent rideshare group constraints as time constraints.

For completeness, we define the search version of the problem, which excludes costs, but is otherwise identical to Definition 1.

Definition 2 (HOV3 carpool problem (search version))**.** *Let $\mathcal{U}$ be the set of agents. Each agent is endowed with a home and a work location $(p_h^u, p_w^u), \forall u \in \mathcal{U}$ as well as a set of time windows $T^u = (t_0^u, t_1^u, \dots), \forall u \in \mathcal{U}$ with the ith time window defined as $t_i^u := [t_{i,s}^u, t_{i,e}^u]$ for each time window (interval). The subscript s denotes the start time of a time window; the subscript e denotes the end time. The collection of feasible subsets of the universe (rideshare groups) $\mathcal{S}$ consists of groups of size three such that the agents have a non-empty common time window. That is,*

$$\mathcal{S} := \left\{ S : S \in 2^{\mathcal{U}}, |S| = 3, \bigcap_{u \in S} T^u \neq \emptyset \right\}. \tag{115}$$

Find a subset $\mathcal{R} \subseteq \mathcal{S}$ such that

- *$\mathcal{R}$ forms a partition of $\mathcal{U}$, as no agent may participate simultaneously in multiple ridesharing groups.*

9.3.2 *Reduction from Exact Cover by 3-Set*

The *Exact Cover by 3-Set problem* is a known NP-Complete problem (by generalization of Tripartite Matching, which is NP-Complete (Dasgupta et al., 2006)). The problem is as follows: $F = \{S_1, \ldots, S_m\}$ of subsets of a finite set U with $|S_i| = 3$ and $|U| = 3m$ for some m. Find m sets in F that are disjoint and have U as their union.

We observe that Exact Cover by 3-Set problem is closely related to the search version of the HOV3 carpool problem, allowing us to show the following complexity result:

Theorem 3. *HOV3 carpool search is NP-complete.*

Proof. First, the problem is trivially in NP. Given a set of rideshare groups, it can be checked in polynomial time that groups indeed form a partition. Naively this can be done in $O(n^2)$. Furthermore, even if given a solution that may contain arbitrary subsets of $\mathcal{U}$ (not necessarily in $\mathcal{S}$), it can be checked in linear time if the groups satisfy time and size constraints (and thus are feasible subsets in $\mathcal{S}$).

Now, we show by reduction from Exact Cover by 3-Set that the HOV3 carpool problem is NP-Hard.

$(\Rightarrow)$: An instance of Exact Cover by 3-Set can be mapped in polynomial time to an instance of the HOV3 carpool problem as follows: Let $\mathcal{U} := U$. For each subset $S_i \in F$, select a new time window t which has no overlap with any other time window used so far, and let $t \in T^u, \forall u \in S_i$. This assignment ensures that $S_i \in F \implies S_i \in \mathcal{S}$, and the uniqueness of the time window ensures that $S_i \in \mathcal{S} \implies S_i \in F$. Note that the size-three constraint is satisfied by definition of the Exact Cover by 3-Set problem.

$(\Leftarrow)$: A solution, i.e. a partition, $\mathcal{R}$ to the HOV3 carpool instance can then be mapped directly to a solution to the Exact Cover by 3-Set instance, as all sets $S \in \mathcal{R}$ are size three, are subsets of $\mathcal{U} = U$, and are by construction in $\mathcal{S} = F$.

Then, Exact Cover by 3-Set reduces to HOV3 carpool search problem, and thus we conclude that the latter is NP-Complete. □

Corollary 1. *The HOV3 carpool problem (optimization version) (see Definition 1), is NP-Hard.*

In addition to the complexity result, the HOV3 Carpool problem has several additional pain points. First, simply finding a feasible solution to the HOV3 problem is difficult.

Lemma 2. *Finding a feasible solution to the HOV3 problem (optimization version) is NP-Complete.*

Proof. Finding a feasible solution is the same as the search version of the problem, which disregards the cost. Then, the result follows from Theorem 3. □

Second, due to the partition constraint, it is difficult to use standard approximation techniques such as linear programming (LP) relaxation. Once the partition constraint is violated, such as through an LP rounding scheme, it is difficult to recover a partition; recovering it may be seen as another combinatorial optimization problem.

Finally, the above results generalize to additional HOV restrictions, such as 4+, 5+, etc. We refer to the more general problem as the HOVk carpool problem, where k denotes the minimum number of occupants in a vehicle required for use of the HOVk lane.

Corollary 2. *The HOVk carpool problem is NP-Hard.*

9.3.3 *Relaxation of HOV3 carpool problem*

Finally, we present the *HOV3- carpool problem*, which we study for the remainder of the chapter, referred to simply as the *carpool problem*. The only difference from the previous problem is that feasible rideshare groups may now have size less than or equal to three ($|S| \leqslant 3$).

Definition 3 (HOV3- carpool problem)**.** *Let $\mathcal{U}$ be the set of agents. Each agent is endowed with a home and a work location $(p_h^u, p_w^u), \forall u \in \mathcal{U}$ as well as time windows $T^u = (t_0^u, t_1^u, \dots), \forall u \in \mathcal{U}$ with $t_i^u = [t_{i,s}^u, t_{i,e}^u]$. The collection of feasible subsets of the universe (rideshare groups) $\mathcal{S}$ consists of groups of size three such that the agents have a non-empty common time window. That is,*

$$\mathcal{S} := \left\{ S : S \in 2^{\mathcal{U}}, |S| \leqslant 3, \bigcap_{u \in S} T^u \neq \emptyset \right\}. \tag{116}$$

The overall objective of the HOV3- carpool problem is to find a subset $\mathcal{R} \subseteq \mathcal{S}$ such that

- *$\mathcal{R}$ minimizes some given cost function $C : 2^{\mathcal{S}} \to \mathbb{R}$.*
- *$\mathcal{R}$ forms a partition of $\mathcal{U}$, as no agent may participate simultaneously in multiple ridesharing groups.*

9.3.4 *Problem setting*

Global optimum. In this chapter, we study computing the globally optimal rideshare groupings with the minimal physical (vehicle) distance. In particular, any agent may be a driver or a passenger. The driver picks up all the passengers. The cost of a rideshare group is the distance traveled by the driver. The overall cost of a carpool solution is thus the sum of the cost of each rideshare group in the solution. That is,

$$c_S = \sum_{(u,v)\in \mathrm{TSP}(S)} d_{uv} \tag{117}$$

$$C = \sum_{S\in\mathcal{R}} c_S \tag{118}$$

where $\mathrm{TSP}(S)$ denotes the solution to the Traveling Salesman Problem. Given a set of agents S, $\mathrm{TSP}(S)$ computes the minimum cost tour (given as pairs of locations, i.e. $(x_1, x_2), (x_2, x_3), ...$). We study the global optimum in the sense that this objective minimizes the total vehicle distance traveled, and thus has implications for fuel consumption and greenhouse gas (GHG) emissions.

Assumptions.

- *Capacity.* We assume that all agents have vehicles, that the capacity is three, that the vehicle capacity is uniform.
- *Single destination, single time window.* We specialize to the setting where all agents have the same destination and all agents are allowed to specify one time window. This is representative of many situations common in the transportation world, for example casual carpool in SF or carpool to campus services (national labs, Microsoft, etc.) and is complementary to real-time dispatch systems such as studied in Miao et al. (2016).
- *Static assignment.* We solve the static version of the problem, where travel is instantaneous; there is no time overhead for pickups.

9.4 PROBLEM FORMULATION

9.4.1 *Carpool as a set partition problem*

We give a natural 3-set partition formulation of the carpool problem.

$$\min_{X=(x_S)_{S\in\mathcal{S}}} \sum_{S\in\mathcal{S}} c_S x_S \tag{119}$$

$$\text{s.t.} \sum_{S:u\in S} x_S = 1, \quad \forall u \in \mathcal{U} \tag{120}$$

$$x_S \in \{0,1\}, \quad \forall S \in \mathcal{S} \tag{121}$$

where $\mathcal{S}$ is the feasible subsets defined in Equation (116) and c_S is the group cost defined in Equation (117).

This formulation, unlike the dynamic programming and integer programming formulations (given in Sections 9.4.2 and 9.4.3, respectively), not only decouples the cost computation c_S from the rest of the problem, it allows representing all the constraints concisely as $\mathcal{S}$. Additionally, by representing the solution vector X as a binary vector of size $|\mathcal{S}|$, this formulation enables the easy design and expression of neighborhoods with respect to the solution vector.

9.4.1.1 *Neighborhoods*

We now describe the studied neighborhoods. We consider two neighborhoods in our subsequent methods: the swap neighborhood and the move neighborhood. The *swap neighborhood* consists of a single pairwise swap between two groups. The *move neighborhood* consists of a single move of an agent from one group to another. Here we define them explicitly. For simplicity, although X denotes a binary vector, we abuse the notation here to represent the set of rideshare groups indicated by the binary vector.

SWAP NEIGHBORHOOD The swap neighborhood consists of a single pairwise swap between two groups. The neighborhood is denoted $\mathcal{N}_S$ and can be explicitly defined as

follows:

$$\mathcal{N}_S(X) := \Big\{ X' : \quad \exists A', B' \in X', \text{s.t. } A', B' \notin X, \tag{122}$$

$$\exists A, B \in X, \text{s.t. } A, B \notin X', \tag{123}$$

$$\exists u \in A' \text{ s.t. } u \in B, u \notin A, \tag{124}$$

$$u' \in A' \iff u' \in A, \text{ for } u' \neq u$$

$$\exists v \in B' \text{ s.t. } v \in A, v \notin B, \tag{125}$$

$$v' \in B' \iff v' \in B, \text{ for } v' \neq v$$

$$S \in X \iff S \in X' \tag{126}$$

$$\text{if } S \notin \{A, B, A', B'\} \Big\}$$

Equations (122) and (123) mean that a solution in the neighborhood of X differs by four entries (i.e. four rideshare groups), since two groups are modified. In particular, there are two groups, denoted A', B', that exist in X' but not X (and vice versa). Now, within these four groups, there exists agents (u, v) which are swapped between them, whereas the rest of the agents in those groups are kept the same (see Equations (124) through (125)). Finally, Equation (126) states that the rest of the solution vectors X', X are the same.

MOVE NEIGHBORHOOD The move neighborhood consists of a single move of an agent from one group to another. The neighborhood is denoted $\mathcal{N}_M$ and can be explicitly defined as follows:

$$\mathcal{N}_M(X) := \Big\{ X' : \quad \exists A', B' \in X', \text{s.t. } A', B' \notin X, \tag{127}$$

$$\exists A, B \in X, \text{s.t. } A, B \notin X', \tag{128}$$

$$\exists u \in A' \text{ s.t. } u \in B, u \notin A, \tag{129}$$

$$u' \in A' \iff u' \in A, \text{ for } u' \neq u,$$

$$v' \in B' \iff v' \in B, \text{ for } v' \neq u,$$

$$S \in X \iff S \in X' \tag{130}$$

$$\text{if } S \notin \{A, B, A', B'\} \Big\}$$

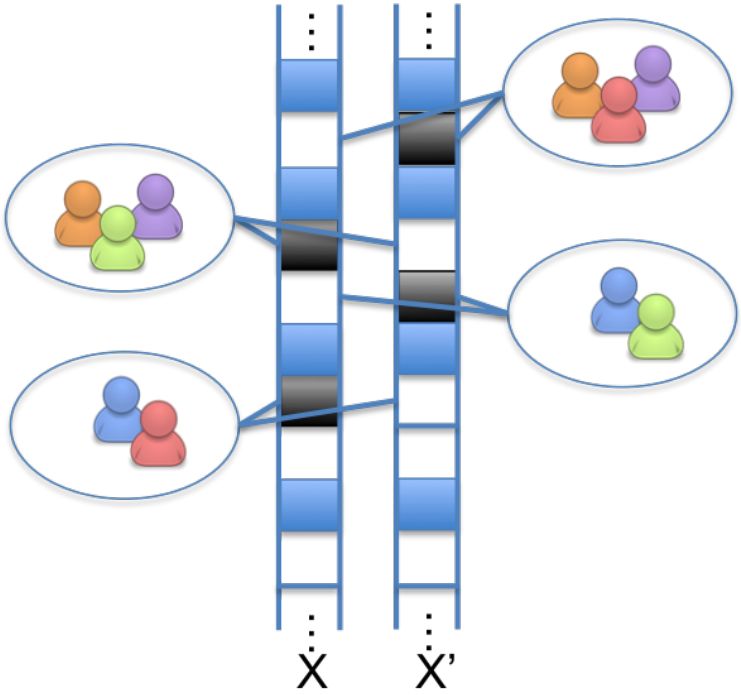


Figure 44: An example of a solution X' in the swap neighborhood of X is given in this figure. Each vector is an indicator vector of the rideshare groups in the assignment; each entry represents one rideshare group. The white entries indicate unselected rideshare groups in the assignment, whereas the blue and black entries indicate selected rideshare groups. As described in Equations (122)-(126), the two vectors differ by exactly four entries (denoted by the black entries), consisting of a swap of a two agents (the green and red agents) between two groups. Best viewed in color.

Similarly to the swap neighborhood, Equations (127) and (128) mean that a solution in the neighborhood of X differs by four rideshare groups, since two groups are modified. In particular, there are two groups that exist in X' but not X (and vice versa). Now, within these four groups, there exists a single agent u which was moved from B into A, and the rest of these groups are unmodified (see Equation (129)). Finally, Equation (130) states that the rest of the solution vectors X', X are the same.

9.4.2 *Carpool as a dynamic programming problem*

For completeness of the exposition, we derive a dynamic programming solution. In particular, we present a formulation of the carpool problem which lends itself to a dynamic

programming (DP) scheme for exhaustively searching the solution space for the optimal solution. This approach provides a naive but exact baseline for comparing our methods, as this is a highly expensive method but one that is guaranteed to return the optimal answer (in exponential time).

Let $I = \{1, 2, \ldots\}$ denote the index set numbering the rideshare groups in the solution. For convenience, let index $j = 0$ represent an unassigned agent, and let $j = 1, 2, \ldots$ be the rideshare group number (i.e., the first group, the second group, etc.). Let $M \in (I \cup \{0\})^n$ denote a (partial) assignment vector, where agent i is assigned to rideshare group number $M[i]$ (which may be 0, i.e. unassigned). This vector keeps track of subproblems in our DP formulation.

The following is notation used in the dynamic programming problem. Let M_0 denote the indices of the zero entries of M, and we let $m := \min M_0$, the index of the first unassigned agent. We denote $\mathbf{1}_i \in \mathbb{Z}^n$ as the indicator vector, with a one in the ith position and zero everywhere else. We use the following binomial coefficient as shorthand for the set of all groups of size two among unassigned agents M_0, excluding agents with indices M':

$$g \in \binom{M_0 \setminus M'}{2} \iff \tag{131}$$

$$g = \{a_1, a_2\} : \{a_1, a_2\} \in M_0 \setminus M' \times M_0 \setminus M', a_1 \neq a_2. \tag{132}$$

Now, we define the following subproblem: $G(M, j)$ assigns the first unassigned agent of M (denoted as m, as above) to rideshare group number j and gives an optimal assignment for the unassigned agents in M:

$$G(M, j) = \min \Bigg\{ c_{\{i\}} + G(M + j\mathbf{1}_i, j + 1), \tag{133}$$

$$\min_{g \in M_0 \setminus \{i\}} c_{\{i,g\}} + G(M + j\mathbf{1}_g + j\mathbf{1}_i, j + 1), \tag{134}$$

$$\min_{g \in \binom{M_0 \setminus \{i\}}{2}} c_{\{i\} \cup g} + G(M + j\mathbf{1}_g + j\mathbf{1}_i, j + 1) \Bigg\} \tag{135}$$

$$G(M, j) = 0 \quad \forall j \quad \text{if } M_0 = \emptyset \quad \text{(base case)} \tag{136}$$

The three terms within the minimization refers to placing i in the best carpool group of size one, two, and three, respectively, and then labeling that as the jth rideshare group. The subsequent subproblem finds the assignment for the $j + 1$th rideshare group and excludes the agents just assigned to group number j. Notice that there is no explicit

collection of feasible subsets, so let $c_S = $ when S is infeasible (e.g. incompatible time windows). The base case (Equation (136)) is invoked when all agents have been assigned, i.e. $M_0 = \emptyset$.

To solve the overall problem, we invoke $G(\mathbf{0}, 1)$, where $\mathbf{0}$ denotes the all zeros vector. That is, at first, all agents are unassigned, and without loss of generality we can place the first agent in rideshare group number 1.

Using the recursion of Equations (133)-(136), we can examine the combinatorial explosion of possible rideshare group assignments. A problem of 10, 20, and 100 agents requires checking 6×10^4, 5×10^{12}, and 1×10^{103} assignments, respectively. With 20 agents, if each possibility can be checked in 1nsec, then the computation would take 1.4 hours. The size 10 problem, on the other hand, computes in 0.00006 seconds. Given these prohibitive costs, the only valid use of this algorithm for our purposes is checking for optimality in small benchmark cases.

9.4.3 *Carpool as an integer program*

For completeness, we also present an integer programming solution to the carpool problem.

Variables. We first define some additional needed notation.

INPUTS
- oo_{uv} = distance between the origin of agent u and origin of agent v. Note that $oo_{uu} = 0$.
- od_u = distance between the origin and destination of agent u
- $T^u = ([t^u_{0,s}, t^u_{0,e}])$ = the time window of valid departure times for agent i. Note that in this formulation, we restrict to setting with only one time window per agent.

OUTPUT
- $y_{uvw} \in \{0, 1\} = 1$ if driver u is carpooling with passengers v and w, 0 otherwise. In addition, if a driver is picking up only one passenger, $y_{uvv} = 1$ and if no passengers, $y_{uuu} = 1$.

IP Formulation. The key insight of our integer programming (IP) formulation, which permits a clean and concise representation, is to encode fixed-size groups of size three, and pad smaller groups as needed. Thus, a lone driver $u \in \mathcal{U}$ would be represented by a group (u, u, u); observe that the distance traveled is $oo_{uu} + oo_{uu} + od_u = 0 + 0 + od_u = od_u$. The IP formulation is as follows:

$$\min_y \sum_{i \in \mathcal{U}} y_{uvw} \cdot [oo_{uv} + oo_{vw} + od_w]$$

subject to

$$\sum_{v,w\in\mathcal{U}} y_{uvw} + \sum_{v,w\in\mathcal{U}} y_{vuw} + \sum_{v,w\in\mathcal{U}} y_{vwu} \tag{137}$$
$$-\sum_{v\in\mathcal{U}} y_{uuv} - \sum_{v\in\mathcal{U}} y_{uvu} - \sum_{v\in\mathcal{U}} y_{vuu}$$
$$+ y_{uuu} = 1, \forall u$$
$$y_{uuv} = 0, \forall u, v, u \neq v \tag{138}$$
$$y_{uvw} \implies [t^u_{0,s}, t^u_{0,e}] \cap [t^v_{0,s}, t^v_{0,e}] \cap [t^w_{0,s} t^w_{0,e}] \neq \emptyset, \forall u, v, w \tag{139}$$
$$(1 - y_{uvw}) \vee (\max_{x\in\{u,v,w\}} (t^x_{0,s}) \leqslant \min_{x\in\{u,v,w\}} (t^x_{0,e})), \forall u, v, w$$

The overall objective is the total distance traveled by the drivers. Equation (137) ensures that each agent be in exactly one group. Since each agent may be a driver y_{u**}, first passenger y_{*u*}, or the second passenger y_{**u}, we need to delete double- and triple-counted duplicates.

Equation (138) addresses the fact that we can represent a two person group with (u, v) as either y_{uuv} or y_{uvv}, but not both. We have chosen arbitrarily y_{uvv} for this case, so we need to ensure that y_{uuv} is never set. Finally, Equation (139) enforces that for each group of agents, there is some overlap in their time windows.

9.4.4 *Carpooling lower bound*

Due to the NP-Completeness of HOV3 carpool problem, the optimal solution cannot be determined efficiently, even for small problem instances; however, by comparing our numerical results to a lower bound on the optimal solution, the effectiveness of the proposed methods can be evaluated.

We denote the *perfect carpooling* lower bound on the carpool problem by

$$\text{obj}_{\text{LB}} = \frac{1}{3} \sum_{u\in\mathcal{U}} c_{\{u\}} \tag{140}$$

which encodes the situation in which every agent is in a three-person carpool and incurs no additional pickup costs. This lower bound is achieved by a problem instance in which there are $3m$ agents and exactly three agents live in each location (and thus incur the same singleton travel costs); that is, for such a problem instance, the lower bound is

indeed the optimal solution.

9.5 METHODS FOR SOLVING THE CARPOOL PROBLEM

By relaxing the original HOV3 carpool problem to the HOV3- carpool problem, we have presented a formulation which is amenable to highly scalable and powerful heuristic algorithms, thereby transforming a challenging combinatorial problem into a practical solution for ridesharing optimization. In this section, we thus describe the algorithms for solving the carpool problem, given in Definition (3), presenting methods for each of the problem formulations given in Section 9.4. For the set partitioning problem, we pose four local search methods; for dynamic programming, we use a memoized recursive approach; for integer programming, we invoked an open-source MILP solver. The methods are summarized in Table 8.

9.5.1 *Local search: hill climbing*

Hill climbing is the simplest and most efficient of our local search methods in terms of per-iteration computation cost, given in Algorithm 5. In each iteration, a random coin flip determines from which neighborhood to sample. Then, a swap or move is sampled accordingly. Since not all solution vectors in the neighborhood are equally close, the sampling distribution is determined by a heuristic weighting (based on the relative distance of the agents involved in the swap or move). Importantly, all neighbors have nonzero probability of being sampled. Then, if the action improves the overall objective value, then the action is accepted (X is updated).

Sampling a neighbor takes $O(n)$ and computing the cost difference takes $O(1)$, so each iteration takes $O(n)$ time.

9.5.2 *Local search: simulated annealing*

A generalization of hill climbing is simulated annealing (see Algorithm 6), which has a few key aspects: 1) it allows multiple actions to occur in a single iteration, 2) it allows actions which worsen the objective to be accepted, and 3) it allows tuning the number of actions and probability of accepting a worse solution, even as the algorithm progresses.

At each iteration, this method consists of sampling a number of actions (moves or swaps) according to the current temperature T. All actions are performed and then eval-

Method	Optimal	HOV3 compatible	Warm-start compatible	Stochastic	Runtime	Practical Limit
Hill climbing	✗	✗	✓	✓	$O(ni)$	100,000+ agents
Simulated annealing	✗	✗	✓	✓	$O(ni)$	100,000+ agents
Swap and move neighborhood	✗	✗	✓	✗	$O(n^2 i)$	1,000 agents
Random neighborhood	✗	✗	✓	✗	$O(n^2 i)$	1,000 agents
Integer programming	✓	✓	✗	✗	$O(\frac{n!}{2^n})$	20 agents
Dynamic programming	✓	✓	✗	✗	$O(\frac{n!}{2^n})$	10 agents

Table 8: This table summarizes the methods studied in this chapter for the HOV3- carpool problem. The attributes examined are as follows: guaranteed optimality of the solution, compatibility with the original HOV3 problem, whether the method is compatible with warm-starts, whether the method is stochastic, the method's runtime, and the practical limit of the method, in terms of the number of agents for which the method can compute a good solution. The i denotes the number of iterations, for the local search methods. The practical limit is determined based on our numerical experiments where the time limit for n agents is set to be $\log n$ hours. The integer programming runtime is approximated as the same as the dynamic programming runtime, which is computed based on the recursive formulation.

Algorithm 5 Hill climbing

Require: feasible initial solution X
Require: number of iterations $k \geqslant 0$
for i in range(k) **do**
  $r \sim \text{Bernoulli}(0.5)$
  if $r == 0$ **then**
    $X' \sim \mathcal{N}_S(X)$
  else
    $X' \sim \mathcal{N}_M(X)$
  end if
  if $c_{X'} \leqslant c_X$ **then**
    $X \leftarrow X'$
  end if
end for
return X

uated relative to the current iterate X. If the objective is better, it is accepted as before. If the objective is worse, the actions can be accepted with probability according to the magnitude of change and a temperature parameter. At the end of each iteration, the temperature parameter is decayed at rate β. The full algorithm is given in Algorithm 6.

Note that in the limit as $k \to$, simulated annealing converges to hill climbing. That is, the method then samples a single action at a time and with high probability accepts the action only if it improves the objective.

The iteration complexity of this method is again $O(n)$. However, the constant factor is larger and determined by the temperature parameters (T, β).

9.5.3 *Local search: swap and move neighborhood*

In the classical local search method (see Algorithm 7), at each iteration, a full search of the swap and move neighborhoods is performed and the single best action is selected, if it yields an improvement.

9.5.4 *Local search: random neighborhood*

Our last local search method combines some of the deterministic aspects of the classical local search method with the stochastic nature of hill climbing. At each iteration, a coin

Algorithm 6 Simulated annealing

Require: feasible initial solution X
Require: number of iterations $k \geqslant 0$
Require: initial temperature $T \geqslant 0$
Require: temperature decay rate $\beta \geqslant 0$ (default= 0.99)

for i in range(k) **do**
  nActions $\leftarrow \lceil \exp(T) \rceil$
  $X' \leftarrow X$
  for j in range(nActions) **do**
    $r \sim$ Bernoulli(0.5)
    if $r == 0$ **then**
      $X' \sim \mathcal{N}_S(X')$
    else
      $X' \sim \mathcal{N}_M(X')$
    end if
  end for
  $r \sim$ Unif$(0, 1)$
  if $c_{X'} \leqslant c_X$ or $r \leqslant \frac{1}{1+\exp((c_{X'}-c_X)/T)}$ **then**
    $X \leftarrow X'$
  end if
  $T \leftarrow \beta T$
end for
return X

Algorithm 7 Classical local search

Require: feasible initial solution X
Require: number of iterations $k \geqslant 0$

for i in range(k) **do**
  $c_{X'}, X' \leftarrow \min(\min_{X' \in \mathcal{N}_S(X)} C_{X'}, \min_{X' \in \mathcal{N}_M(X)} C_{X'})$
  if $c_{X'} \leqslant c_X$ **then**
    $X \leftarrow X'$
  end if
end for
return X

flip chooses a random neighborhood (move or swap). Then, the best action within that neighborhood is selected (see Algorithm 8).

Algorithm 8 Local search with random neighborhood

Require: feasible initial solution X
Require: number of iterations $k \geqslant 0$
for i in range(k) **do**
  $r \sim$ Bernoulli(0.5)
  if $r == 0$ **then**
    $c_{X'}, X' \leftarrow \min_{X' \in \mathcal{N}_S(X)} C_{X'}$
  else
    $c_{X'}, X' \leftarrow \min_{X' \in \mathcal{N}_M(X)} C_{X'}$
  end if
  if $c_{X'} \leqslant c_X$ **then**
    $X \leftarrow X'$
  end if
end for
return X

9.5.5 *Exact search: dynamic programming*

We use a memoized recursive DP scheme to solve the subproblems given by Equations (133)-(136) in Section 9.4.2.

9.5.6 *Exact search: Integer programming*

We implement our IP formulation in Python-based Numberjack (Hebrard et al., 2010), an open-source framework that interfaces with MILP and constraint programming (CP) solvers. Specifically, we use the SCIP solver (Achterberg, 2009).

9.6 WARM-STARTING THE LOCAL SEARCH METHODS

In this section, we study the techniques for using a warm start to affect the convergence of our iterative local search methods for the carpool problem. We describe several initialization schemes and study whether they help speed up the computation, important for practical ridesharing optimization solutions. We refer to them as *raw initializations*

because they may not satisfy all the constraints of our problem; for instance, time constraints may be violated. However, we developed a procedure to transform the raw initializations into a feasible solution with which to start the local search methods (given in Section 9.11). Each of these initializations is no worse (and almost certainly better) than the trivial initial solution, which consists of all singleton sets as the rideshare groups.

Since local search methods may converge slowly or to a poor local optimum given an arbitrary starting point, in this section, we utilize the structure of the problem to construct raw initialization points to warm start the local search method, instead of using a naive initialization. An example naive initial solution is to group agents into sets of three arbitrarily (and then transform that assignment into a feasible solution). We now give several heuristic alternatives, which we compare numerically in Section 9.8.4.

Greedy initialization. The greedy initialization is an $O(n^2)$ scheme, which forms groups by greedily selecting an unassigned agent and the two unassigned agents who are closest (in origin); the scheme terminates when all agents are assigned (see Algorithm 9 in Section 9.11 for more details).

Distance-based clustering initialization. This initialization computes a distance-based clustering, with a measure on agent origins, and is similar to the approach in Chapter 10, which uses centroid-based clustering to solve a set partitioning problem, such as ridesharing. The algorithm essentially computes a clustering using k-means, and then splits up the clusters into subsets that satisfy the capacity constraints of agents. Note that both clustering-based initializations implicitly assume an Euclidean space through use of k-means clustering. Thus, these heuristic initializations are likely to perform better for the Euclidean settings and Euclidean-like network configurations than the general road network settings. For example, networks divided by rivers or other geography may benefit less from such a heuristic. The initialization is given in more details in Algorithm 10 in Section 9.11.

Angle-based clustering initialization. Similarly, this initialization computes an angle-based clustering on the angle of the agents origins, relative to the destination. The algorithm for computing an angle-based clustering is given in Algorithm 11 in Section 9.11.

Extending the greedy initialization to clustering-based initialization (either distance-based or angle-based) has the potential to boost the performance significantly by starting the objective closer to an optimal value. However, since the initializations promote better starting objectives without explicitly considering the time constraints (the greedy initialization has the same fault), much of the objective gained could be lost as soon as the time constraints are enforced.

9.7 NUMERICAL IMPLEMENTATION

For our experiments, we generate problem instances with agent sizes that are exponentially increasing (from 10 to 100K) and attempt to solve them using multiple methods, both exact and approximate.

9.7.1 *Workloads*

We experiment with problem instances drawn from two different settings.

Euclidean setting. In the Euclidean setting, we generate agent home locations based on a Gaussian distribution over the $\mathbb{R}^2$ space. We assume that the distances in the space are Euclidean. Work locations are fixed at the origin, and home locations are distributed according to a single Gaussian distribution with standard deviation 0.1 and centered at the work location. Time windows are determined by sampling two numbers uniformly in the range [6,9] (i.e. morning commute hours). Then, the smaller of the numbers is the start time of the time window; the larger is the end time.

San Francisco (SF) Bay Area setting. In this setting, we assign agent home locations by sampling with replacement from a dataset consisting of 400K agent plans for the San Francisco Bay Area (Pozdnoukhov et al., 2016) based on the California Metropolitan Transportation Commission (MTC) travel model.

The dataset was generated using MATSIM, an agent-based transportation simulator (Horni and (eds.), 2016), which invokes a co-evolutionary scheme to generate agent plans in a large-scale multi-agent urban environment. The distances in this space are clearly not Euclidean, due to the bay in middle, and had to be computed using the real road network. Work locations are fixed at Soda Hall at UC Berkeley. The agent plans include the time that the agent arrives at her work location, so we used that as the end time of the time window. We uniformly fix a one-hour time window for each agent.

Problem sizes. In the prior work, the largest instances have been 1000 and typically much smaller (Armant et al., 2015). However, if we want to support carpooling at a scale that can make a significant social impact, we need to work with much larger problem sizes. For example, the population of the nine-county Bay Area is more than seven million (Cynthia Kroll et al., 2016). So supporting even 1-2% of the population will require supporting agent counts that are one-two orders of magnitude larger (70K - 140K). Therefore, we studied instances of {10, 100, 1K, 10K, 100K} agents.

9.7.2 *Initialization*

We now briefly describe our stages of computing an initial solution, which is important for the local search methods. Using one of the initialization schemes in Section 9.6, we compute a raw initialization to the problem; we refer to this as a raw initialization (denoted the `raw_init` stage in Figures (49) and (50)) because this initialization could violate the time constraints. Then to satisfy the time constraints, we arbitrarily break up groups which are not compatible in time. We refer to this as the `raw_init_time_ranges` stage. Finally, we solve the small traveling salesman problem to optimize the ordering of the agents in each group and thereby compute c_S; this is the `raw_init_opt_pickup` stage.

9.7.3 *Computational challenges of routing distance*

In this section we discuss the non-trivial computational challenges of using distances on networks and how we overcame them. In summary, pre-computation is desirable but may be expensive; we must consider both storage and time costs of pre-computation.

In the San Francisco Bay Area setting, the act of querying (i) distances from origin to destination of agents, and (ii) pairwise distances between agent origins, underlies the operation of every method. Although computing Euclidean distances takes roughly 0.5 msec, querying a routing service incurs I/O cost, thus increasing the time cost to the order of hundreds of milliseconds. Thus network settings (such as the SF Bay Area network) appears to immediately incur a penalty of 100x, regardless of how fast the algorithm is. To overcome this, we pre-compute the $O(n^2)$ required distances. For 100K agents, the space requirement for pre-computation is on the order of 20GB; this is easily supportable even with a general purpose `m4.2xlarge` AWS instance, which has 32GB of RAM.

Unfortunately, due to the $O(n^2)$ nature of the pairwise distance computations, even the pre-computation time does not scale well beyond 10k agents. As we can see from Table 9, even using the `roadsindb` distance oracle (Samet and Sankaranarayanan, 2014), which uses a pre-computed representation of driving distances between all locations in a particular region, scaling from problem size 1k to size 10k increases the time required from 209 secs to 3 hours. This quadratic increase implies that the pre-computing distances for 100k agents would require 300 hours = 12.5 days. Thus in the 100K agents instances, we instead query distances on the fly as needed from `roadsindb`, and this is successfully used in our largest hill climbing and simulated annealing experiments, which require fewer distance queries per iteration than the other local search methods.

Agents	Euclidian dists	SF Bay Area dists
10	0.009	0.043
100	0.721	2.010
1000	71.760	209.356
10K	412.742	12.3×10^3

Table 9: Time scaling for pre-computing pairwise agent distances for different problem sizes, for both Euclidean plane and SF Bay Area settings; the latter is calculated using queries to the cached `roadsindb` data. All times are in secs.

9.7.4 *Experimental Setup*

Our experimental setup consists of a combination of a medium-sized AWS instance and a large shared server. The system configurations of the two servers: 1) an AWS instance with 8 CPUs and 30GB RAM, used to generate instances, pre-compute distance matrices, and execute the 100k-agent local search experiments, and 2) a shared lab server with 24 cores and 256GB RAM, used to run instances of 10-10K agents using pre-computed distance matrices.

The experimental timeouts are set logarithmically; the timeouts are $\{1, 2, 3, 4, 5\}$ hours for $\{10, 100, 1000, 10k, 100k\}$ agents, respectively.

9.7.5 *Implementation*

The local search algorithms and dynamic programming scheme are implemented entirely in Python and could be further optimized to achieve a speedup of several magnitudes. Thus, the implementations are proof of concept prototypes. On the other hand, the MILP solver is already highly tuned software. Additionally, we used simple and fixed parameters (such as a simple temperature schedule for the simulated annealing method) and did very little tuning of hyperparameters, as that is not the focus of this chapter.

9.8.1 *Scalability*

In line with findings from Section 9.7.3, algorithms with linear or greater running times did not scale well. While the exponential exhaustive search algorithms was not expected to scale, we were surprised to find that it did not finish solving even the 20 agent problem in over a day.

When we analyzed the computation time of four local search methods (hill climbing, simulated annealing, classical local search (called `best_step`), and random neighborhood local search (called `best_random`) with increasing workloads, we found that the `best_step` and `best_random` methods did not finish even one 10k iteration before timing out after 4 hours. Figure 45 summarizes the time per iteration cost of each method for varying problem sizes.

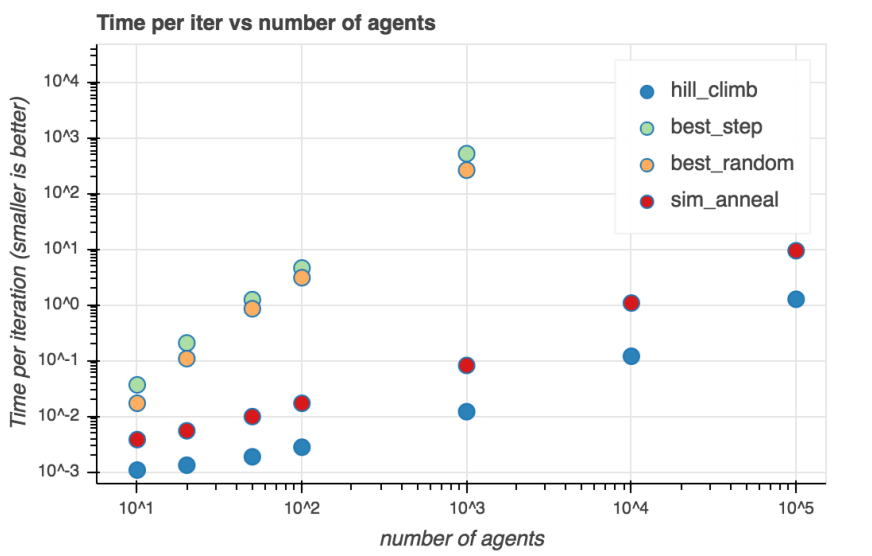


Figure 45: Time per iteration for the local search methods for varying numbers of agents. Note that the scale is log-log. The `best_step` and `best_random` methods were not able to compute problem sizes 10k and 100K within the time alloted for the respective problem sizes (see Section 9.7.1).

We additionally break the computation time down into the various initialization stages (see Section 9.7.2) and the per iteration computation. The results are summarized in Figure 49. Hill climbing and simulated annealing scale significantly better than the local search methods that perform a full neighborhood search. For these methods, the per

iteration computation is significantly less than the initialization time.

This is an expected assessment because the local search methods that perform a full neighborhood search are much more thorough in their evaluation of which action to perform at each iteration (thus taking $O(n^2)$ time to choose). The next natural question is whether it is a necessary expense.

9.8.2 *Exact solvers*

We were unable to compute exact solutions for problem sizes greater than 50 for IP and greater than 10 for dynamic programming (see Table 10). Even in the case of 10 agents, hill climbing out performs IP.

size	IP obj	IP time	HC obj	HC time
10 agents	0.898	2.137	0.898	1.11
20 agents	1.578	44.360	1.577	2.77
50 agents	3.413	12225.513	3.513	8.35

Table 10: Comparison between the objective and the run-time (sec) for the IP and hill climbing (HC) methods. This data is from a single random instance from the Euclidean setting.

9.8.3 *Convergence*

We now turn to assessing the convergence of each of the methods to the naive lower bound. Relating the objective value to the naive lower bound conveniently allows us to assess different problem sizes in the same figure by normalizing the value. Figure 46 summarizes the convergence of the different methods (minus simulated annealing) for different problem sizes. We find that hill climbing by far converges the fastest. However, the classical local search method converges to a lower ratio in some instances. These findings suggest combining the two methods to achieve the best of both worlds: first using hill climbing to quickly converge close to a local optimum, then using classical local search to converge to an exact local optimum.

Simulated annealing is excluded from the previous figure due to its chaotic nature in convergence. To demonstrate this we have included a figure comparing the convergence of simulated annealing to hill climbing (see Figure 47). We note that our simulated an-

nealing implementation is not highly tuned for our problem. With tuning such as specialized temperature schedules and restarts, simulated annealing has been demonstrated to perform very well in many practical combinatorial optimization settings. We do not claim that it would not work here as well, but that we have not discovered optimal hyperparameters.

9.8.4 *Warm-start*

All three proposed warm-starts give a solution that is between two- and three- times the lower bound objective (see Figure 48), indicating that all the initial solutions are far from optimal. In fact, three times the lower bound is the upper bound and corresponds to the solution where each agent rides alone (which is the trivial solution of singleton groups).

Surprisingly, hill climbing with the greedy initialization warm-start performs at least as well as the other proposed warm-starts (see Figure 48). This is surprising because we would expect that the distance- and angle-based warm-starts are closer to the final solution than a greedy initialization. In particular, the angle-based warm start is a proxy for the arcs emanating out of the single destination, which we expect drivers to follow to best optimize. On the contrary, the angle-based initialization often performs the worst, and remains much farther from the optimal solution than the other methods. Further investigation is required into the causes of this deviation. For the SF Bay Area setting, this may make sense because the geography of the region (the bay, highways, etc.) does not often permit straight-line routes to the destination. Nonetheless, all the warm-starts begin far from the optimal solution, and these results imply that the optimization method is effective enough to compensate for any differences in the warm-start strategies. For practitioners, this implies that a greedy initialization strategy may be good enough.

9.9 CHAPTER SUMMARY

In this chapter, we cast the *HOV3- Carpool problem* into the framework of combinatorial optimization and shown that it is NP-Hard and difficult to solve iteratively. Next, we show that a relaxed version of the problem, the *HOV3- Carpool problem*, is amenable to many classes of solution methods for combinatorial optimization: local search, integer programming, and dynamic programming. In particular, the new relaxed formulation permits local search methods, and we experimentally show that a sampling-based local search method (hill climbing) scales up to 100K agents and converges to a ratio of within 1.1 of the lower bound in five hours, with a prototype implementation. The other

methods either do not complete in the time allotted or do not converge as well.

This work is a first step towards incorporating various, complex human considerations into carpool optimization. There are many considerations yet to be addressed for a real-world ride sharing system to be viable. For instance, our work does not consider the delay caused by waiting for participants, travel time, routing complications, or congestion. Additionally, participants' preferences with respect to when to travel or arrive may be non-uniform and may be correlated with spatial-temporal features. There are also many complex human factors that must be considered including fairness, social norms, and cultural compatibility. One of the major benefits of local search methods such as the ones studied in this chapter is that they are extremely flexible with respect to different cost terms in the objective; however, further investigation is needed for an empirical confirmation.

On the other hand, local search methods are highly sensitive to the constraints of the optimization problem, and therefore new methods will be needed for new or modified constraints, such as allowing for more flexible carpool sizes or permitting participants to specify intermediate stops. However, problems with such constraints can potentially be decomposed into independent problems with simplified constraints.

In practice, we expect that some of these complex cost factors and constraints may actually contribute to cleaner or faster methods, introduce new technical problems, and improve the feasibility of carpooling in the real-world. We especially encourage the research community to embrace the complexity and take advantage of the structure within these complex human factors.

There are several extensions to pursue next. We would like to understand the algorithmic complexity of the HOV3- problem, and we are interested in convergence guarantees for the hill climbing method, perhaps by casting the method into a MCMC framework and studying Markov chain summaries. We are also interested in studying more complex agent constraints and costs; for instance, agents may have limited tolerance for deviating from a baseline travel plan. We are interested in conducting more extensive experiments after improving implementations, combining methods, or trying alternative warm-starting schemes.

9.10 SCALABILITY: TIME BREAKDOWN FOR LOCAL SEARCH

See Figures 49 and 50 for breakdowns on computational time in the various stages for the local search methods.

See Algorithms 9, 10, and 11 for details of the three warm-start strategies. The procedure for enforcing feasibility of solutions is described in Algorithm 12 and essentially removes agents from groups until feasibility is achieved. Note that the `group.agents.pop()` function in line 3 is arbitrary – it can remove the first agent in the list, the most troublesome agent, etc. Groups of singleton agents form trivially feasible groups. In our experiments, we refer to this feasibility enforcement by the `raw_init_time_ranges` stage, since capacity constraints are already taken care of in the raw initialization, so the only constraint to enforce is the time constraint.

Algorithm 9 Greedy initialization of groups $\mathcal{R}_0$

Require: agents.size $\geqslant 0$
1: $\mathcal{R} = \{\}$
2: free_agents $\leftarrow$ agents
3: **while** free_agents is not empty **do**
4: a = free_agents.pop()
5: agents = a.get_nearest_two_agents()
6: free_agents.remove(agents)
7: $\mathcal{R} \leftarrow \mathcal{R} \cup$ group([a, agents])
8: **end while**
9: **return** $\mathcal{R}$

Algorithm 10 Distance cluster initialization of groups $\mathcal{R}_0$

Require: number of clusters $k > 0$
1: $X = (x_i, y_i)_{i \in [n]}$
2: $\mathcal{R} = \{\}$
3: lists_of_agents $\leftarrow$ k-means(X, k)
4: **for** list_of_agents in lists_of_agents **do**
5: **while** list_of_agents is not $\emptyset$ **do**
6: agents $\leftarrow$ list_of_agents.pop_three()
7: list_of_agents.remove(agents)
8: $\mathcal{R} \leftarrow \mathcal{R} \cup$ group(agents)
9: **end while**
10: **end for**
11: **return** $\mathcal{R}$

Algorithm 11 Angle cluster initialization of groups $\mathcal{R}_0$

Require: number of clusters $k > 0$

$X = (x_i - w_x, y_i - w_y)_{i \in [n]}$
$Y = \arctan((y_i - w_y)/(x_i - w_x))$ {Compute the angle relative to the destination}
$\mathcal{R} = \{\}$
lists_of_agents $\leftarrow$ k-means(Y, k)
for list_of_agents in lists_of_agents **do**
  while list_of_agents is not $\emptyset$ **do**
    agents $\leftarrow$ list_of_agents.pop_three()
    list_of_agents.remove(agents)
    $\mathcal{R} \leftarrow \mathcal{R} \cup$ group(agents)
  end while
end for
return $\mathcal{R}$

Algorithm 12 Ensure feasibility of $\mathcal{R}$

for group in $\mathcal{R}$ **do**
  while group is not feasible **do**
    a = group.agents.pop()
    $\mathcal{R} \leftarrow \mathcal{R} \cup$ new_group([a])
  end while
end for
return $\mathcal{R}$

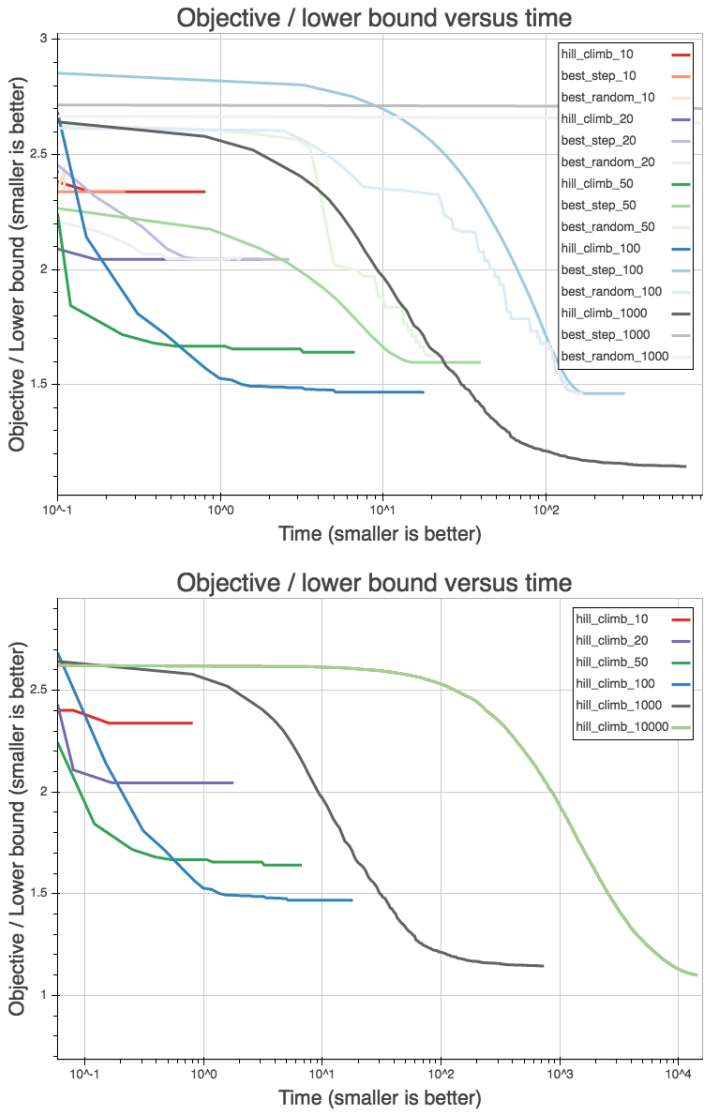


Figure 46: This figure demonstrates the convergence of the methods in the SF Bay Area setting. **Top:** For the smaller problem instances: $\{10, 20\}$, the three methods displayed converge to the same ratio to the lower bound, but hill climbing converges much faster and, for the larger problem instances of $\{50, 100, 1000\}$, to a lower ratio. The classical local search method gives a steady but slow convergence, whereas the random neighborhood search gives a more jagged convergence. In the case of size 50 and 100, the classical local search method converges to a slightly lower ratio than hill climbing, implying that hill climbing may have reached a different local optimum (or not reached one at all). In the case of problem size 1000, the `best_step` and `best_random` (the gray near-horizontal lines) are further from the lower bound and were cut short due to the timeout. **Bottom:** We display convergence only for the hill climbing method and problem instances $\leqslant$ 100K. In all instances the problem converges to between 1.1 to 2.4 times the lower bound. Notably, in the largest problem size, hill climbing achieves a ratio close to 1, implying that the solution is near optimal. Note that the x-axis is on a log scale.

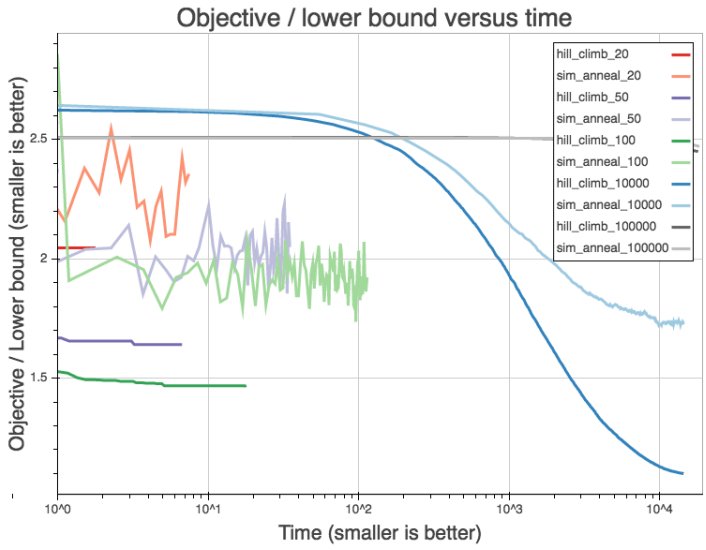


Figure 47: Convergence of the simulated annealing method (lighter shades) is much more chaotic than hill climbing (darker shades), and simulated annealing does not converge to as low a ratio to the lower bound as the hill climbing method. The experiments shown here are from the SF Bay Area setting. Note that the x-axis is on a log scale.

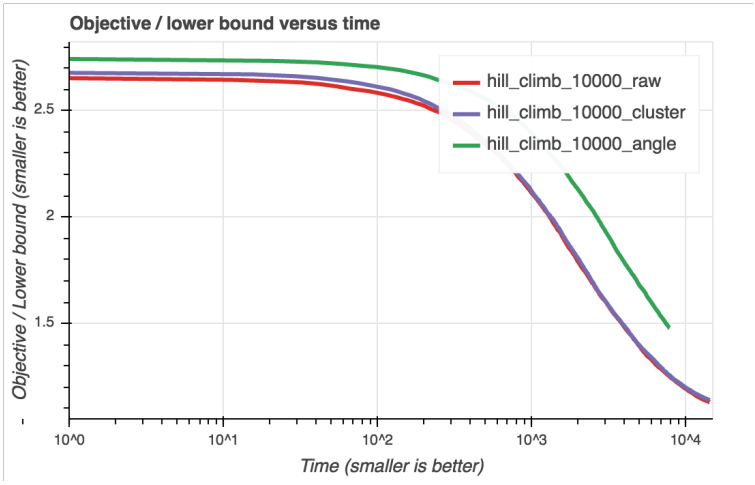


Figure 48: Three different warm-start strategies (shown for the SF Bay Area setting) and their subsequent performance with the hill climbing method. The three warm-starts are: a greedy grouping, a clustering of the initial positions based on relative distance, and a clustering of the initial positions based on relative angle to the destination. They appear to perform similarly numerically, with the angular clustering performing the worst. Here is displayed a sample experiment configuration, using the hill climbing method, 10K agents, and averaged across nine runs. Note that the x-axis is on a log scale.

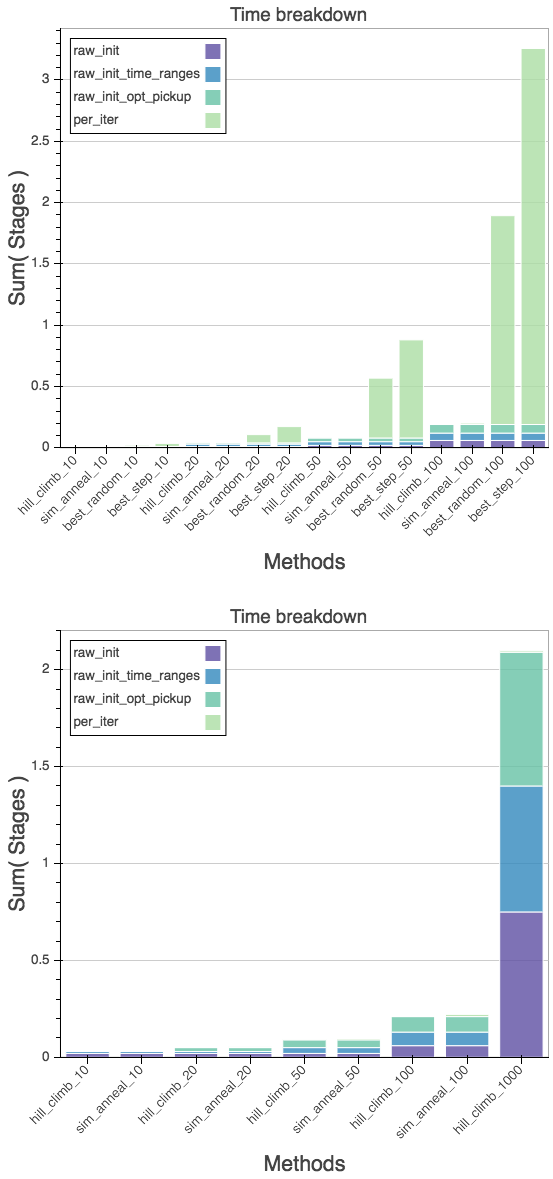


Figure 49: This figure demonstrates the scalability of the methods in the SF Bay Area setting, using the greedy warm-start. **Top:** With 20 agents or more, the per iteration computation dominates the local search methods which perform a full neighborhood search. Beyond 100 agents, the relative timing becomes indistinguishable because the per iteration time dominates, so we have excluded the larger values. **Bottom:** We focus on the sampling-based local search methods, which scale much better in per iteration computation time. At 1000 agents, the per iteration computation is only a small fraction of the total initialization time (the topmost stacked bar (light green), barely visible), although for larger sizes, the initialization far dominates the per iteration computation because the initialization time is $O(n^2)$, whereas the per iteration computation is $O(n)$. Best viewed in color.

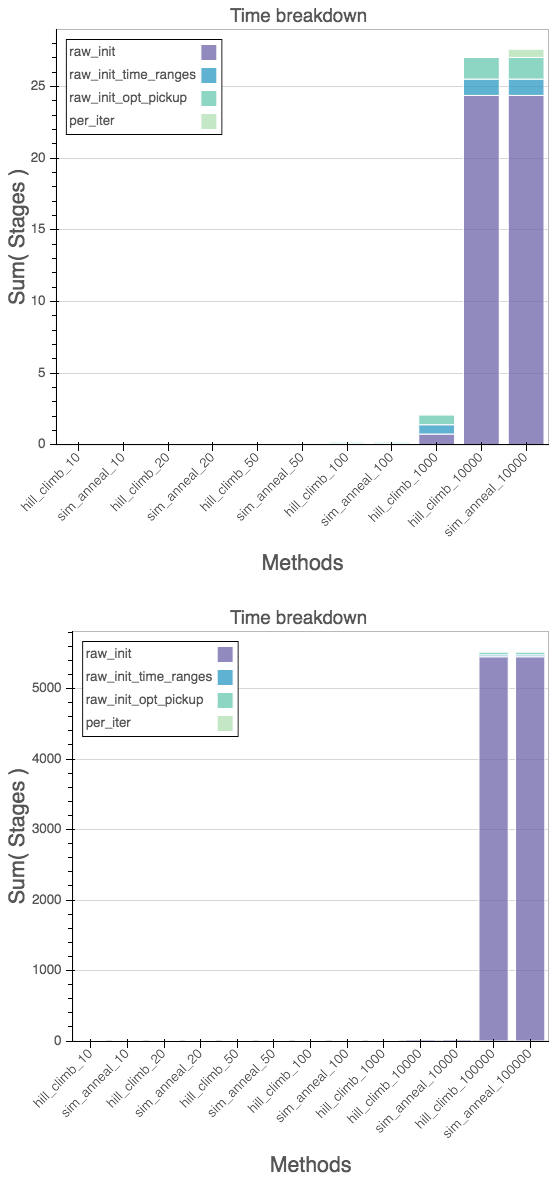


Figure 50: For the larger problem instances (10k and 100K), the initialization time far dominates the per iteration computation time. This is a result of the $O(n^2)$ nature of the initialization process, which computes pairwise distances between unassigned agents and searches for the minimum. The results shown here are for the SF Bay Area setting. **Top**: Up to problem size 10k is included, and we can see that the smaller problem instances are already dwarfed by it. **Bottom**: Including problem size 100K, we can no longer even see the runtime of any of the smaller experiments, and for size 100K, the other steps in the computation are dwarfed by the init time. Best viewed in color.

10

CLUSTERING FOR SET PARTITIONING WITH A CASE STUDY IN RIDESHARING

We are all connected;
To each other, biologically
To the earth, chemically
To the rest of the universe atomically.

Neil deGrasse Tyson,
We Are All Connected,
Symphony of Science, 2009

By exploring alternative approaches to combinatorial optimization, we propose the first known formal connection between clustering and set partitioning, with the goal of identifying a subclass of set partitioning problems that can be solved efficiently and with optimality guarantees through a clustering approach. We prove the equivalence between classical centroid clustering problems and a special case of set partitioning called metric k-set partitioning. We discuss the implications for k-means and regularized geometric k-medians, and we give several future extensions and applications. Finally, we present a case study in combinatorial optimization for ridesharing, in which we use an efficient Expectation Maximization (EM) style algorithm to achieve a 69% reduction in total vehicle distance, as compared with no ridesharing.

10.1 OVERVIEW AND COMBINATORIAL OPTIMIZATION PROBLEMS

Set partitioning is the optimization form of exact cover, one of Karp's 21 NP-complete problems, and it arises commonly in planning applications where it is used to identify the best partition of a set of objects. Examples include scheduling airline flight crews (Hoffman and M. Padberg, 1993; Chu et al., 1997), sharing rides (Santi et al., 2013), and

kidney swapping programs (Biro et al., 2009; Vemuganti, 1999). Optimizing for the best set partitioning solution is NP-hard, and classical combinatorial optimization methods are cumbersome (e.g. integer programming (M. W. Padberg, 1973), branch-and-cut (Hoffman and M. Padberg, 1993)), restricted (e.g. triangle-packing approximation (Hassin and Rubinstein, 2006; J. Wang and Feng, 2008), m-sets (Arkin and Hassin, 1998; Chandra and Halldorsson, 2001)), or do not provide guarantees (e.g. simulated annealing (Dowsland, 1993), genetic algorithms (Gandibleux et al., 2004), metaheuristics (Delorme et al., 2004)). At the same time, clustering methods also solve a partitioning problem, and although the underlying optimization problem is again NP-hard, methods in common use provide efficient iterative procedures that offer probabilistic guarantees and that are amenable to parallelization. On the other hand, while classical combinatorial optimization approaches are suitable for extremely general and expressive problems, clustering methods are typically restricted to specific models and objectives.

We are interested in the following questions:

- What optimization problems can be expressed at the intersection of the two approaches?
- How can these problems benefit from the formalism of one and the methods of the other?

We formalize the connection between set partitioning and clustering, with the goal of identifying a subclass of set partitioning problems that can be solved efficiently and with optimality guarantees through a clustering approach. We offer a demonstrative example of the connection: we demonstrate the correspondence between a restricted setting of set partitioning and classical centroid-based unsupervised clustering methods, for instance the k-means algorithm (Hartigan and Wong, 1979).

10.2 SET PARTITIONING

First, consider the problem of *set partitioning* (see Figure 51), where the goal is to find the best disjoint cover within some collection $\mathcal{S} \subseteq 2^{\mathcal{U}}$, where $\mathcal{U}$ is the universe of elements. This is a restricted setting of set packing, where feasible solutions must cover the whole universe, rather than a subset. Best is defined as minimizing the overall cost in terms of weights assigned to each subset. We denote the cost of each subset $S \in \mathcal{S}$ by c_S. We denote the solution vector $x = (x_S)_{S \in \mathcal{S}}$, where $x_S = 1$ indicates that S is in our set partitioning solution. Then, the problem is defined as follows:

$$\text{opt} = \min_{x} \sum_{S \in \mathcal{S}} c_S x_S \tag{141}$$

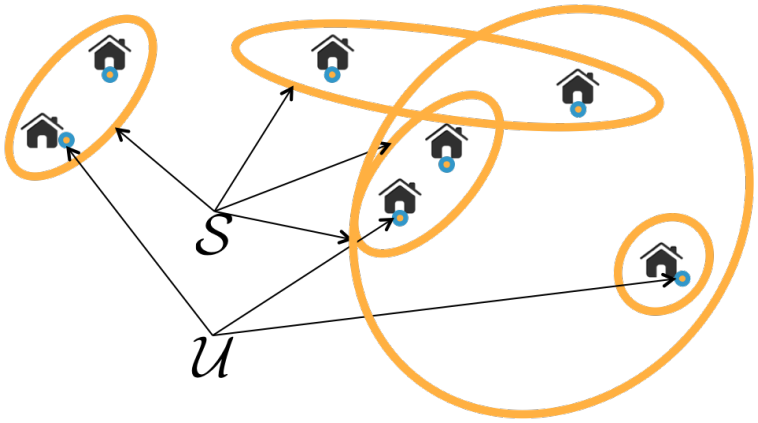


Figure 51: Set partitioning problem.

subject to

$$\sum_{S:u\in S} x_S = 1, \quad u \in \mathcal{U} \tag{142}$$

$$x_S \in \{0,1\} \quad S \in \mathcal{S} \tag{143}$$

10.2.1 *Metric k-set partitioning and centroid-based clustering*

Now, we specialize to the special case of metric spaces and k-covers (covers of exactly size k). We let $\mathcal{S} = 2^{\mathcal{U}}$, and we endow universe $\mathcal{U}$ with a metric space $(\mathcal{X}, d)$ such that $\mathcal{U} \subseteq \mathcal{X}$ and define the cost of a subset to be minimal relative to a centroid point, that is $c_S := \min_{x\in\mathcal{X}} \sum_{s\in S} d(s,x)$. Additionally including a k-set constraint, we then have the following optimization problem, which we call the *metric k-set partitioning problem*:

$$\min_x \sum_{S\in\mathcal{S}} \min_{x'\in\mathcal{X}} \sum_{s\in S} d(s,x')x_S = \min_x \sum_{S\in\mathcal{S}} \sum_{s\in S} d(s,\mu_S)x_S \tag{144}$$

subject to

$$\sum_{S:u\in S} x_S = 1, \quad u \in \mathcal{U} \tag{145}$$

$$\sum_{S\in\mathcal{S}} x_S = k \tag{146}$$

$$x_S \in \{0,1\} \quad S \in \mathcal{S} \tag{147}$$

where we define the subset centroids $\mu_S := \arg\min_{x \in \mathcal{X}} \sum_{s \in S} d(s, x)$. Note that under a metric, without the size k constraint, the optimal solution would be all the singleton sets. This is true for clustering as well.

Constraint (145) implies that each element is measured with respect to exactly one centroid (and in fact, the closest one, due to the objective) and Constraint (146) encodes that there are exactly k centroids (and in fact, they minimize the distance with respect to the elements assigned to it, due to the objective). With these observations, the previous optimization problem then collapses neatly into the following well-studied optimization problem for *centroid-based clustering*:

$$\min_{(T_1, T_2, \ldots, T_k)} \sum_{j=1}^{k} \sum_{x \in T_j} d(x, \mu_j) \tag{148}$$

subject to

$$\cup_{j \in [k]} T_j = \mathcal{U} \tag{149}$$

$$T_i \cap T_j = \emptyset, \quad \forall i \neq j \tag{150}$$

$$\mu_j = \arg\min_{x \in \mathcal{X}} \sum_{s \in T_j} d(s, x) \tag{151}$$

When the metric $d(\cdot, \cdot)$ is restricted to the sum of squares loss, also knowns as k-means clustering, the resulting optimization problem may be solved efficiently and with probabilistic guarantees with `k-means++` initialization (Arthur and Vassilvitskii, 2007) and the k-means algorithm. Thus, the *metric k-set partitioning problem*, an instance of set partitioning, benefits tremendously from a clustering approach. We now prove the result.

10.3 A FORMAL CONNECTION BETWEEN CLUSTERING AND SET PARTITIONING

Theorem 4 (Equivalence). *Metric k-set partitioning and centroid-based clustering are equivalent. That is, Problem (144)-(147) and Problem (148)-(151) are equivalent.*

Before proving the theorem, we first make the following definition and lemma:

Definition 4 (k-partition). *If $\mathcal{P}$ is a k-partition constraint of $\mathcal{U}$, then $\mathcal{P} = (P_1, P_2, \ldots, P_k) \subseteq \mathcal{S}$*

such that:

$$\begin{aligned} P_i \subseteq \mathcal{U}, \quad & \forall i \\ \bigcup_i P_i = \mathcal{U} & \\ P_i \cap P_j = \emptyset, \quad & \forall i \neq j \end{aligned}$$

Lemma 3 (k-partition constraint)**.** *The constraints of Problem (144)-(147) is a* k*-partition constraint.*

Proof of claim. $(\rightarrow)$: We first show that we may construct such a partition $\mathcal{P}$ from a solution $x = (x_S)_{S \in \mathcal{S}}$ satisfying the constraints of the problem defined by Equations (144)-(147). Define $\mathcal{P} := \{S : x_S = 1, S \in \mathcal{S}\}$. By construction, $\forall P \in \mathcal{P}, P \in \mathcal{S} \implies P \subseteq \mathcal{U}$. Constraint (146) ($\sum_{S \in \mathcal{S}} x_S = k$) implies that $|\mathcal{P}| = k$. Constraint (145) ($\sum_{S:u \in S} x_S = 1, \quad u \in \mathcal{U}$) implies both coverage and mutually disjoint conditions. Thus, x may be represented as a k-partition $\mathcal{P}$.

$(\leftarrow)$: Now we show that a partition $\mathcal{P}$ may be used to construct a solution $x = (x_S)_{S \in \mathcal{S}}$ satisfying the constraints of Problem (144)-(147). We define $x \in \{0,1\}^{|\mathcal{S}|}$ such that

$$x_S := \begin{cases} 1 & \text{if } S \in \mathcal{P} \\ 0 & \text{otherwise} \end{cases}$$

Then

$$\begin{aligned} \bigcup_i P_i = \mathcal{U} &\implies \sum_{S:u \in S} x_S \geqslant 1, \quad u \in \mathcal{U} \\ P_i \cap P_j = \emptyset, \quad \forall i \neq j &\implies \sum_{S:u \in S} x_S \leqslant 1, \quad u \in \mathcal{U} \end{aligned}$$

Together, this implies Constraint (145) ($\sum_{S:u \in S} x_S = 1, u \in \mathcal{U}$). Constraint (146) is trivially satisfied by construction. Thus, $\mathcal{P}$ may be represented as a length-$|\mathcal{S}|$ binary vector satisfying the constraints of Problem (144)-(147). Finally, we conclude that the constraints of Problem (144)-(147) are equivalent to that of a k-partition. □

Proof of theorem. We perform a change of variables in Problem (144)-(147) from $x = (x_S)_{S \in \mathcal{S}}$ to $\mathcal{P}$ (established in Lemma 3), taking $\mathcal{P} := \{S : x_S = 1, S \in \mathcal{S}\}$.

$$\begin{aligned}\min_{x} \sum_{S \in \mathcal{S}} \min_{x' \in \mathcal{X}} \sum_{s \in S} d(s, x') x_S &= \min_{\mathcal{P}} \sum_{S \in \mathcal{P}} \min_{x' \in \mathcal{X}} \sum_{s \in S} d(s, x') 1 \\ &\quad + \sum_{S \in \mathcal{S} \setminus \mathcal{P}} \min_{x' \in \mathcal{X}} \sum_{s \in S} d(s, x') 0 \\ &= \min_{\mathcal{P}} \sum_{S \in \mathcal{P}} \min_{x' \in \mathcal{X}} \sum_{s \in S} d(s, x')\end{aligned}$$

subject to the k-partition constraint. We observe that our sum is reduced from $|\mathcal{S}|$ terms to k terms. To make the size of $\mathcal{P}$ more explicit, we may write equivalently,

$$\min_{\mathcal{P} = (P_1, P_2, \ldots, P_k)} \sum_{i=1}^{k} \min_{x' \in \mathcal{X}} \sum_{s \in P_i} d(s, x')$$

Finally, we define $\mu_i := \arg\min_{x' \in \mathcal{X}} \sum_{s \in P_i} d(s, x')$, and we arrive at a common form of the centroid-based clustering problem (as in Problem (148)-(151)):

$$\min_{\mathcal{P} = (P_1, P_2, \ldots, P_k)} \sum_{i=1}^{k} \sum_{s \in P_i} d(s, \mu_i)$$

subject to

$$\begin{aligned} P_i &\subseteq \mathcal{U}, \quad \forall i \\ \bigcup_i P_i &= \mathcal{U} \\ P_i \cap P_j &= \emptyset, \quad \forall i \neq j \end{aligned}$$

Since all operations are reversible (equivalences), we have established equivalence of the two problems. □

Some examples of centroid-based clustering problems include the k-means problem, the k-median problem, and the geometric k-median problem. Under equivalence given by Theorem 4, the algorithms for one problem may apply also to the other. We now present several special cases and extensions, and then we present an application that

makes use of these settings.

10.3.1 *Special case:* k-*means*

A special case of the equivalence is the k-means objective (Steinhaus, 1956), where $d(s, x) = \min_{x \in X} \sum_{s \in S} \|s - x\|_2^2$.

Corollary 3 (k-means++ for set partitioning)**.** *When restricted to sum of squares loss,* `k-means++` *attains a* $O(\log k)$*-approximate solution in expectation to the metric* k*-set partitioning problem* `k-means++`*.*

By examining the operations of the k-means algorithm, we may gain intuition on how the steps translate back into context of the set partitioning problem.

Atomic operations of Lloyd's algorithm: Lloyd's algorithm for k-means clustering is an iterative method with two main steps (Lloyd, 1982):

1. Assignment step:

$$P_i^{(t)} = \left\{ s : \left\| s - \mu_i^{(t)} \right\| \leqslant \left\| s - \mu_j^{(t)} \right\|, \forall j \in [k] \right\} \tag{152}$$

   where each $s \in \mathcal{U}$ is assigned to exactly one $P_i^{(t)}$.
2. Update step:

$$\mu_i^{(t+1)} = \frac{1}{\left| P_i^{(t)} \right|} \sum_{s_j \in P_i^{(t)}} s_j \tag{153}$$

In the context of set partitioning, the two steps have the following interpretation:

1. Assignment step: Select a (better) feasible binary solution vector $x = (x_S)_{S \in \mathcal{S}}$. Recall that a feasible x implies that x satisfies a k-partition constraint.
2. Update step: Update the objective to reflect the cost of the selected subsets, i.e. update $\sum_{S \in \mathcal{S}} c_S x_S$ by computing c_S for each selected subset S.

The algorithm terminates when no change is made during the assignment step; that is, a better feasible binary solution vector cannot be found by the algorithm. We observe that, after each iteration of Lloyd's algorithm (after the update step), the algorithm maintains a feasible solution to the corresponding set partitioning problem. This observation also motivates related methods for solving set partitioning problems, such as local search methods, which maintain feasible solutions at each iteration.

10.3.2 *Extension: regularized centroid-based clustering*

We can now discuss our first extension beyond the classical centroid-based clustering to the regularized setting.

Definition 5 (Relative cost)**.** *We call* $c_{S,r}$ *a relative cost if it takes the form*

$$c_{S,r} := \min_{x \in \mathcal{X}} \sum_{s \in S} d(s, x) + r(x)$$

where $r : \mathcal{X} \to \mathbb{R}$ *denotes a relative term.*

Corollary 4 (Regularization is relative-cost)**.** *Denote regularizer* $r : \mathcal{X} \mapsto \mathbb{R}$*. Then adding* $\sum_{j=1}^{k} r(\mu_j)$ *to the objective of centroid-based clustering (Problem (148)-(151)) and the corresponding definition of* μ_j *is equivalent to the* k*-set partitioning problem (144)-(147) with the relative cost* $c_{S,r} := \min_{x \in \mathcal{X}} \sum_{s \in S} d(s, x) + r(x)$*. That is, the regularized centroid-based clustering problem is equivalent to the relative-cost metric* k*-set partitioning problem.*

10.3.3 *Extension:* k*-median on graphs*

Instead of embedding our universe $\mathcal{U}$ in a metric space, we can embed it in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Then, set partitioning has an equivalence to clustering according to graph distance instead of a metric. We define an analogous cost called *graph median*.

Definition 6 (Graph median)**.** *Given a graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$*,* $v \in \mathcal{V}$ *is the graph median of a subset* $S \subseteq \mathcal{V}$ *if*

$$v = \arg\min_{u \in \mathcal{V}} \sum_{s \in S} \tilde{d}(s, u)$$

where $\tilde{d}(s, u)$ *denotes the shortest path distance between nodes* s *and* u*.*

Lemma 4 (Median on graphs)**.** *The graph median for a subset* S *of cardinality* m *can be computed in* $O\big(\log(m(|V| + |E|) \log |V|)\big)$*.*

10.3.4 *Embeddable cost substructures*

In the more general setting of metric set packing, where the solution need not cover the entire universe, we may draw a connection to how clustering methods handle outliers. Additionally, by making use of recent advances in semi-supervised clustering and spectral clustering, we propose to further identify a subclass of set partitioning problems that may be embedded in a cluster learning framework for efficient computation and probabilistically optimal set partitioning. When we characterize set partitioning problems in terms of their underlying cost structure and additional constraints, we conjecture a formal parallel between set partitioning and clustering with the following *embeddable substructures*: metric spaces, relative cost, pairwise affinity, m-sets, and singleton sets. For each embeddable substructure, we plan to prove the equivalence to their clustering counterpart and give an example efficient and often parallelizable algorithm. We also discuss when the substructures and algorithms may be combined and study clustering methods beyond centroid-based methods, and we hope to contribute a unifying theory of semi-supervised clustering in terms of the underlying optimization problems.

10.4 CASE STUDY: RIDESHARING MEETUP PROBLEM

We have been studying the cluster learning approach to solving a large-scale ridesharing problem. The ridesharing problem is classically formulated as a set partitioning or set cover problem (Kamar and Horvitz, 2009), with complex costs dependent on a road network and the constraints of the participants. When these costs are decomposed into embeddable substructures, we hope to demonstrate the ability of fast clustering methods to solve classically combinatorial problems.

We first study a simplified setting of sharing rides when commuting to work in the morning. We restrict to a single destination area. Consider the setting where users in a ride share group agree to meet up at a location, the users travel there individually, and then they share a single vehicle from the meetup point to work. We formally define the ridesharing meetup problem (see Figure 52):

Definition 7 (Ridesharing meetup problem)**.** *Let $\mathcal{U}$ denote the universe of users $u \in \mathcal{U}$, who live in $\mathcal{X} = \mathbb{R}^2$ and work at the origin $(0,0) \in \mathcal{X}$. Let $\mathcal{S}$ be the collection of possible ride shares. We wish to select the ride share groups and their respective meetup points that minimize the total vehicle distance.*

This is very naturally a set partitioning problem. The cost of each subset $S \in \mathcal{S}$ can be written as $c_S = \min_{x \in \mathcal{X}} \sum_{s \in S} \|s - x\|_2 + \|x\|_2$. The first term is the total distance traveled

by each member of the ride share to the meetup point, and the second term is the distance traveled by one shared vehicle. Assume (for now) that vehicles have infinite (or large) capacity. By Corollary 4, we observe that we have a relative cost that is equivalent to an ℓ_2-regularized loss in the clustering optimization.

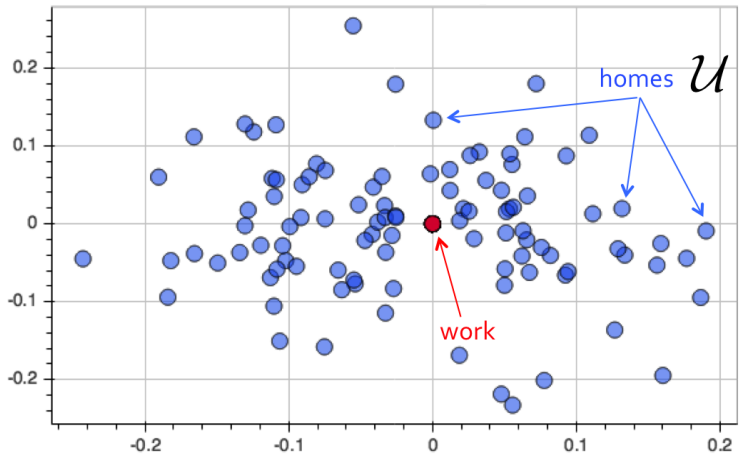


Figure 52: Ridesharing meetup problem. 100 normally distributed users, with a single destination at the origin $(0, 0)$.

Then, by Theorem 4 and Corollary 4, we formulate this problem as a regularized geometric k-median problem, which takes the following objective

$$\min_{(T_1,T_2,\ldots,T_k)} \sum_{j=1}^{k} \sum_{x \in T_j} \|x - \mu_j\|_2 + \lambda \|\mu_j\|_2 \tag{154}$$

When $\lambda = 1$, this formulation exactly minimizes the total vehicle distance. Thus, we can solve the ridesharing problem with an Expectation Maximization (EM) style method. We present preliminary results in Figure 53, resulting in 69% less vehicle distance (compared against singleton sets, i.e. no ridesharing). We may further extend the ride share meetup problem to networks (instead of Euclidean space) by Lemma 4.

10.5 CHAPTER SUMMARY

We have presented the first formal connection between set partitioning and clustering, with the goal of identifying a subclass of set partitioning problems that can benefit from algorithms and theoretical guarantees for clustering problems. We prove that classical

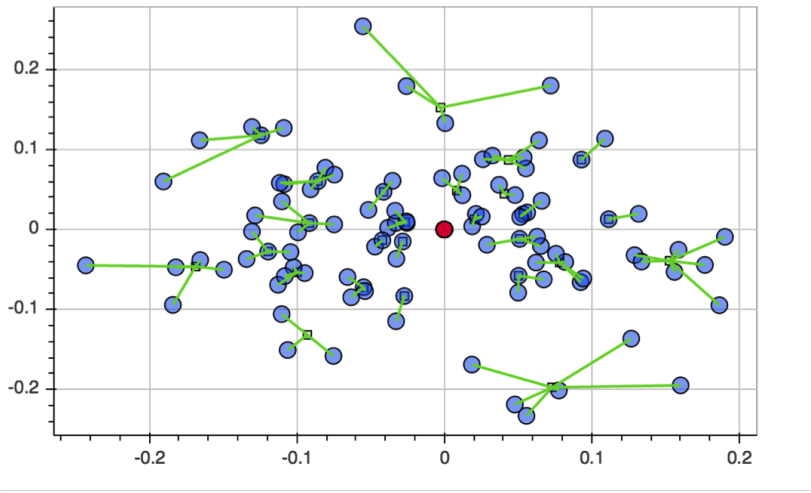


Figure 53: Regularized geometric k-median for the ridesharing meetup problem results in 26 ride share groups from 100 users.

centroid clustering problems are equivalent to metric k-set partitioning, and we present a simplified ridesharing problem that uses this formalism to achieve fast algorithms for a classical combinatorial optimization problem.

In reality, the constraints and costs of a large-scale ridesharing problem will be significantly more complicated than the setting we studied in this chapter. As suggested above, distances will be non-Euclidean, there will be multiple destinations and time windows, and participants will have different preferences about time, money, and social factors, as well as different types of constraints. A real-world ridesharing system may have extrinsic properties such as monetary incentives by means of payments between participants or time incentives by means of high-occupancy vehicles (HOV) lanes. In some settings, a linear pickup order may make more sense than a participant meetup scheme.

Moving forward, one promising approach is to decompose the complex overall ridesharing problem into simple problems such as those studied in this chapter, as a way to both compose principled building blocks for complex system design and achieve theoretical guarantees. Further investigation is needed to determine how solutions of the decomposed problems might be merged together meaningfully to solve the overall problem. In another direction of work, we seek to understand how multiple complex cost structures and embeddable substructures can be supported within the combinatorial optimization framework such that efficient algorithms are still admissible.

Part IV

FINAL REMARKS

11

THE ROAD AHEAD

> Today we are well underway to a solution of the traffic problem.
>
> Robert Moses, master builder of the New York City area, 1948

Automation and AI have profound impacts on society. As such, mixed autonomy is a challenging, broad, and increasingly important problem domain for society moving forward. This thesis is only a first step in enabling a science and engineering of mixed autonomy systems. We introduced the broad definition and challenges of mixed autonomy systems, developed scalable optimization algorithms and systems, and addressed concrete challenges–in control, state estimation, and system design–in particular in mobility. Historically, these problems have largely been studied in isolation by different research communities and subcommunities using vastly different tools. This thesis sought establish through a concrete example from the open world–in mobility–the complex interplay between all the moving parts in mixed autonomy: from the theoretical and algorithmic development to the computer and simulation systems to the ultimate application. We close now with a discussion of the road ahead. We present challenges and opportunities for research in mixed autonomy.

11.1 CHALLENGES IN MIXED AUTONOMY

The continued and accelerating introduction of AI and automation into society will generate wide-spread yet poorly understood externalities; this calls for a new science and engineering of mixed autonomy, with the goal of enabling the design, management, and regulation of such mixed autonomy systems. The challenges presented in this section aims to be the start rather than the end of a dialogue. Mixed autonomy inherits the

challenges of automation science, including those for control systems, machine learning systems, and engineering systems. These challenges include safety, security, efficiency, stability, robustness, resilience, and backwards compatibility. Following are a few additional important challenges specific to mixed autonomy.

Degrees of autonomy. Mixed autonomy is most intuitive in settings which can be posed as a fraction of a system being automated, such as the fraction of vehicles or workers being automated, as studied in this thesis. However, there are different ways of viewing the degree of automation of a task, such as driving. Different degrees, levels, or combinations of partial automation, such as steering, lane keeping, route guidance, or highway driving, have different implications on critical human factors such as trust and attention. These, in turn, have consequences for safety and efficiency. How can the degrees of automation of tasks be better understood? What are the external effects of such partial automation on other participating agents, such as other roadway users, pedestrians, and traffic lights? How can the effects of combinations of partial automation be modeled or predicted?

Learning in the absence of human behavior models. The introduction of automation into the open world, may alter the behavior of the human actors with whom it interacts. As discussed in Chapter 1, this phenomenon is also called behavioral drift, and current design paradigms for automation largely do not take this into account. The challenge lies in the fact that we simply do not have models for human behavior in situations they have never experienced, such as with the introduction of new (or a new degree of) automation. For instance, how can we anticipate the very first interaction of a human driver with an automated vehicle at an intersection, if humans have never experienced that situation before? Or how can we predict the "steady state" intersection interaction, once humans have acclimated to the automation? In what situations can we predict behavioral drift? How can we design automation that is resilient to behavioral drift?

Mixed mixed autonomy. As automation is increasingly adopted by society, we can imagine situations in which the automated components will not only have complex effects when interacting with human agents, but also with one another. For example, with enough adoption of electric automated vehicles, these vehicles can directly interact with and influence energy markets (e.g. negotiating with refrigerators), in addition to the mobility system. Such integration of multiple mixed autonomy systems can certainly yield a more desirable outcome than the separate systems, but is likely also to exhibit unanticipated effects. What methods and tools would enable the study of these mixed mixed autonomy systems?

Science of abstraction. Due to the increasing introduction of automation and AI, our previous working assumption (or abstraction) of different system components as being independent no longer holds in many systems of societal relevance. For instance, in mobility systems, we demonstrated that even a very small fraction of automated vehicles can have a profound impact on the overall mobility system through subtle effects on the human actors in the system. As such, we can no longer study these different system components in isolation. That is not to say that we must study all components jointly, as that is likely to be intractable. However, this observation calls for a re-evaluation of what types of abstractions (or assumptions) are "reasonable" to permit desirable outcomes in mixed autonomy systems.

Unanticipated effects. The interaction of automated and human actors can generate both anticipated and unanticipated effects. Both types of effects are important. An effect being anticipated does not mean it is easy to address, of course, but it can be viewed as a first step. In this thesis, we studied anticipated effects, such as induced demand, and unanticipated effects, such as the emergent behaviors in traffic settings. How can unanticipated effects be discovered? Can unanticipated effects be addressed directly?

11.2 OPPORTUNITIES IN MIXED AUTONOMY

In light of these challenges, we see great opportunities to leverage modern advances in computing, both in terms of algorithms and systems, for the study of mixed autonomy in a variety of application domains. As such, we describe specific directions of promising research in terms of three broad areas, which we have seen throughout this thesis:

- **Optimization, control, and learning**: Mixed autonomy systems are hugely complex systems which in different situations employ different optimization frameworks, including convex optimization, reinforcement learning, and combinatorial optimization.
- **High performance computing**: Mixed autonomy benefits tremendously from high performance computing, for large-scale optimization, processing massive datasets, as well as for rich and complex simulation.
- **Domain modeling and engagement**: Mixed autonomy tightly interfaces with domains with existing systems, and it is the domain which gives the mixed autonomy system both the rich problem structure and the eventual impact on people.

These broad areas overlap in interesting and exciting ways that point towards promising

research directions. Below, as we discuss research directions, we indicate the corresponding area(s) using the above colored dots.

Structured reinforcement learning algorithms. (●) Reinforcement learning algorithms can benefit substantially from design that utilizes the structure of a given task. Model-based reinforcement learning or optimal control is the most dramatic instantiation of this, which assumes full knowledge of the problem structure. Model-based reinforcement learning methods seek to speed up learning by fitting a dynamics model and using it for planning or speeding up learning (Deisenroth and Rasmussen, 2011). However, in most situations, the algorithm may only have access to incomplete or crude information about the problem. Exciting research in this direction utilizes the inherent hierarchical structure of tasks, i.e hierarchical RL (Dayan and Hinton, 1993; Vezhnevets et al., 2017), multi-agent problem structure, i.e. multi-agent deep RL (Busoniu et al., 2008; Lowe et al., 2017), reward structures, e.g. reward shaping and implicit rewards (Ng et al., 1999; Chentanez et al., 2005), and learned models (Levine et al., 2016; Nagabandi et al., 2017). Also of interest is the automatic detection of certain structures within a given task and utilizing this information to leverage the strength of a variety of methods, such as in switching or sliding mode control (S. V. Drakunov and Utkin, 1992; Young et al., 1999).

Task complexity. (●, ●) A closely related and very important question in reinforcement learning is one of sample complexity. The question of sample complexity can be viewed as: how much data must we collect in order to achieve "learning" for a large class of tasks, such as discrete MDPs? We refer the reader to Kakade et al. (2003) for an overview of sample complexity in reinforcement learning. Relatedly, the question of *task complexity* can be viewed as: how much data must we collect in order to solve a particular task? This view is motivated by the fact that we can consider an algorithm to be successful if it can solve problems that we care about, rather than all problems, many of which may never manifest. Conversely, with the amount of time that we have to collect data, what kinds of tasks can we solve? This question is implicit in nearly all empirical reinforcement learning research–from determining the number of expert demonstration needed for an imitation learning task to reward shaping to selecting critical hyperparameters (e.g. discount factor, batchsize). Task complexity is also the key question for the design for reinforcement learning benchmarks (Todorov et al., 2012; Bellemare et al., 2013; Brockman et al., 2016; Synnaeve et al., 2016; Beattie et al., 2016; Côté et al., 2018). Task complexity would benefit from dedicated empirical and theoretical exploration, with an eye to problem structure drawn from important domains. For instance, techniques from control theory can shed light on when a task can or cannot be solved with a linear policy (Callier and Desoer, 2012; Khalil, 1996).

Compute-aware optimization. (●, ●) The development of optimization algorithms and systems has largely been conducted in isolation. One result is the observation that only linear and sublinear algorithms are often tractable in practice, even though a much larger class of algorithms are tractable in theory. Similarly, workloads are increasingly complex for modern machine learning and optimization; beyond the algorithm itself, there is the computational cost to loading, sampling, or simulating data, the time cost of communicating between processes and nodes, and the complexity introduced by the use of function approximation. Important directions moving forward include the development of optimization methods and system which can intelligently trade off between the costs of different computational resources (e.g. bandwidth, CPU time, space), as well as further development of finite sample analysis of learning algorithms.

High performance simulation. (●, ●) Closely integrated with high performance computing systems, simulations of molecular dynamics have the capability of simulating multi-trillion particles for a variety of scientific and engineering domains, including molecular biology, solid-state physics, material science (Heinecke et al., 2015). Societal domains, such as mobility, energy, and social networks, on the other hand, have historically heavily emphasized simplistic models for its component actors. While this has the benefit of permitting analysis tools such as game theory and mean field analysis, it has the downside of often failing to capture pertinent attributes of the actors. There is great potential moving forward in building high performing simulation for large-scale societal domains, which closely integrate domain-specific modeling and high performance computing systems.

Closing the loop. (●, ●) Closing the loop on mixed autonomy systems means working closely with domain experts to carefully evaluate the systems. Depending on the domain, these integrated studies can take the form of user studies, field experiments, system design recommendations, public policy recommendations, and/or decision support systems. These studies will likely involve highly performant real-time systems, with which users, policy makers, engineers alike may engage and provide feedback.

BIBLIOGRAPHY

Abrahamsson, T. (1998). "Estimation of origin-destination matrices using traffic counts - a literature survey." In: *Interim Report IR-98-021, International Institute for Applied Systems Analysis, Laxenburg, Austria* (cit. on p. 117).

Achterberg, T. (2009). "SCIP: Solving constraint integer programs." In: *Mathematical Programming Computation* 1.1. http://mpc.zib.de/index.php/MPC/article/view/4, pp. 1–41 (cit. on p. 186).

Agatz, N., A. Erera, M. Savelsbergh, and X. Wang (2012). "Optimization for dynamic ride-sharing: A review." In: *European Journal of Operational Research* 223.2, pp. 295–303 (cit. on pp. 169, 170).

Anderson, J. M., N. Kalra, K. D. Stanley, P. Sorensen, C. Samaras, and O. A. Oluwatola (2016). *Autonomous vehicle technology: a guide for policymakers*. Santa Monica, CA: Rand Corporation (cit. on p. 151).

Arkin, E. M. and R. Hassin (1998). "On local search for weighted k-set packing." In: *Mathematics of Operations Research* 23.3, pp. 640–648 (cit. on p. 202).

Armant, V. and K. N. Brown (2014). "Minimizing the driving distance in ride sharing systems." In: *Tools with Artificial Intelligence (ICTAI), 2014 IEEE 26th International Conference on*. IEEE, pp. 568–575 (cit. on p. 169).

Armant, V., N. Mahbub, and K. N. Brown (2015). "Maximising the number of participants in a ride-sharing scheme: MIP versus CP formulations." In: *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*. IEEE, pp. 836–843 (cit. on pp. 169, 170, 188).

Arthur, D. and S. Vassilvitskii (2007). "k-means++: The advantages of careful seeding." In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, pp. 1027–1035 (cit. on p. 204).

Au, T.-C., S. Zhang, and P. Stone (2014). "Semi-autonomous intersection management." In: *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, pp. 1451–1452 (cit. on p. 54).

Auld, J., M. Hope, H. Ley, V. Sokolov, B. Xu, and K. Zhang (2016). "POLARIS: Agent-based modeling framework development and implementation for integrated travel demand and network and operations simulations." In: *Transportation Research Part C:*

Emerging Technologies 64, pp. 101–116. URL: https://polaris.es.anl.gov/ (cit. on p. 101).

Baert, A.-E. and D. Seme (2004). "Voronoi mobile cellular networks: topological properties." In: *In Third International Symposium on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks*, pp. 29–35 (cit. on p. 116).

Balk, G. (2013). "Census: Redmond has largest daytime population surge in U.S." In: *The Seattle Times*. URL: http://blogs.seattletimes.com/fyi-guy/2013/06/03/census-redmond-has-largest-daytime-population-surge-in-u-s/ (cit. on p. 152).

Ban, X. (, P. Hao, and Z. Sun (2011). "Real time queue length estimation for signalized intersections using travel times from mobile sensors." In: *Transportation Research Part C: Emerging Technologies* 19.6, pp. 1133–1156. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2011.01.002. URL: http://www.sciencedirect.com/science/article/pii/S0968090X11000143 (cit. on p. 115).

Bando, M., K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama (1994). "Structure stability of congestion in traffic dynamics." In: *Japan Journal of Industrial and Applied Mathematics* 11.2, pp. 203–223 (cit. on pp. 39, 104).

Bando, M., K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama (1995). "Dynamical model of traffic congestion and numerical simulation." In: *Physical review E* 51.2, p. 1035 (cit. on pp. 39, 104).

Bar-Gera (2002). "Origin-based Algorithm for the Traffic Assignment Problem." In: *Transportation Science* (cit. on pp. 126, 133).

Barth, M. and K. Boriboonsomsin (2008). "Real-world carbon dioxide impacts of traffic congestion." In: *Transportation Research Record: Journal of the Transportation Research Board* 2058, pp. 163–171 (cit. on p. 37).

Barth, M. and K. Boriboonsomsin (2009). "Energy and emissions impacts of a freeway-based dynamic eco-driving system." In: *Transportation Research Part D: Transport and Environment* 14.6. The interaction of environmental and traffic safety policies, pp. 400–410. ISSN: 1361-9209. DOI: https://doi.org/10.1016/j.trd.2009.01.004. URL: http://www.sciencedirect.com/science/article/pii/S1361920909000121 (cit. on pp. 37, 51).

Baskar, L. D. (2009). *Traffic management and control in intelligent vehicle highway systems*. TU Delft, Delft Univ. of Technology (cit. on p. 54).

Batista, M. and E. Twrdy (2010). "Optimal velocity functions for car-following models." In: *Journal of Zhejiang University-SCIENCE A* 11.7, pp. 520–529 (cit. on p. 105).

Bayen, A. M., I. M. Mitchell, M. K. Osihi, and C. J. Tomlin (2007). "Aircraft autolander safety analysis through optimal control-based reach set computation." In: *Journal of Guidance, Control, and Dynamics* 30.1, pp. 68–77 (cit. on p. 26).

Beattie, C., J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, et al. (2016). "DeepMind Lab." In: *arXiv preprint arXiv: 1612.03801* (cit. on pp. 83, 216).

Beckmann, M., C. B. McGuire, and C. B. Winsten (1956). *Studies in the Economics of Transportation*. Ed. by N. H. Yale Univ. Press. Cowles Commission Monograph (cit. on pp. 126, 137).

Behrisch, M., L. Bieker, J. Erdmann, and D. Krajzewicz (2011). "SUMO–simulation of urban mobility: an overview." In: *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind (cit. on pp. 42, 85).

Bell, M. G. H. and Y. Iida (1997). *Transportation Network Analysis*. Wiley, West Sussex, United Kingdom (cit. on pp. 115, 117, 126).

Bell, M., C. Shield, F. Busch, and G. Kruse (1997). "A stochastic user equilibrium path flow estimator." In: *Transportation Research Part C: Emerging Technologies* 5.34, pp. 197–210. ISSN: 0968-090X. DOI: http://dx.doi.org/10.1016/S0968-090X(97)00009-0. URL: http://www.sciencedirect.com/science/article/pii/S0968090X97000090 (cit. on p. 115).

Bellemare, M. G., Y. Naddaf, J. Veness, and M. Bowling (2013). "The Arcade Learning Environment: An evaluation platform for general agents." In: *Journal of Artificial Intelligence Research (JAIR)* 47, pp. 253–279 (cit. on pp. 83, 216).

Belletti, F., D. Haziza, G. Gomes, and A. M. Bayen (2018). "Expert level control of ramp metering based on multi-task deep reinforcement learning." In: *IEEE Transactions on Intelligent Transportation Systems* 19.4, pp. 1198–1207 (cit. on pp. 53, 82, 101).

Bellman, R. (1957). *A Markovian decision process*. Tech. rep. DTIC Document (cit. on p. 86).

Best, M. J. and N. Chakravarti (1990). "Active set algorithms for isotonic regression; a unifying framework." In: *Math. Programming* 47, pp. 425–439 (cit. on pp. 131, 132).

Biro, P., D. F. Manlove, and R. Rizzi (2009). "Maximum weight cycle packing in directed graphs, with application to kidney exchange programs." In: *Discrete Mathematics, Algorithms and Applications* 1.04, pp. 499–517 (cit. on p. 202).

Blandin, S., L. E. Ghaoui, and A. Bayen (2009). "Kernel regression for travel time estimation via convex optimization." In: *IEEE Conference on Decision and Control* (cit. on p. 117).

Blincoe, L., T. R. Miller, E. Zaloshnja, and B. A. Lawrence (2015). *The economic and societal impact of motor vehicle crashes, 2010 (Revised)*. Tech. rep. (cit. on p. 166).

Bojarski, M., D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. (2016). "End to end learning for self-driving cars." In: *arXiv preprint arXiv:1604.07316* (cit. on p. 53).

Bopardikar, S. D., B. Englot, and A. Speranzon (2015). "Multiobjective Path Planning: Localization Constraints and Collision Probability." In: *IEEE Transactions on Robotics* 31.3, pp. 562–577. ISSN: 1552-3098. DOI: 10.1109/TRO.2015.2411371 (cit. on p. 82).

Bose, A. and P. A. Ioannou (2003). "Analysis of traffic flow with mixed manual and semi-automated vehicles." In: *IEEE Trans. on Intelligent Transportation Systems* 4.4, pp. 173–188 (cit. on p. 100).

Boyd, S., N. Parikh, E. Chu, B. Peleato, and J. Eckstein (2011). "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers." In: *Foundations and Trends in Machine Learning* 3, pp. 1–122 (cit. on p. 14).

Boyd, S. and L. Vandenberghe (2004). *Convex Optimization*. Cambridge University Press. URL: http://www.stanford.edu/~boyd/cvxbook/ (cit. on pp. 13, 118, 128, 130).

Brackstone, M. and M. McDonald (1999). "Car-following: a historical review." In: *Transportation Research Part F: Traffic Psychology and Behaviour* 2.4, pp. 181–196 (cit. on pp. 27, 37, 84).

Brébisson, A. de, É. Simon, A. Auvolat, P. Vincent, and Y. Bengio (2015). "Artificial neural networks applied to taxi destination prediction." In: *arXiv preprint arXiv:1508.00021* (cit. on p. 53).

Brockman, G., V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba (2016). "OpenAI gym." In: *arXiv preprint arXiv:1606.01540* (cit. on pp. 83, 93, 216).

Bubeck, S. et al. (2015). "Convex optimization: Algorithms and complexity." In: *Foundations and Trends® in Machine Learning* 8.3-4, pp. 231–357 (cit. on p. 13).

Buehler, M., K. Iagnemma, and S. Singh (2009). *The DARPA urban challenge: autonomous vehicles in city traffic*. Vol. 56. springer (cit. on p. 26).

Busoniu, L., R. Babuska, and B. De Schutter (2008). "A comprehensive survey of multiagent reinforcement learning." In: *IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews* 38.2, p. 156 (cit. on pp. 52, 216).

Caceres, N., J. P. Wideberg, and F. G. Benitez (2007). "Deriving origin-destination data from a mobile phone network." In: *IET Intell. Transp. Syst* 1, pp. 15–26 (cit. on p. 116).

Calabrese, F., G. D. Lorenzo, L. Liang, and C. Ratti (2011). "Estimating Origin-Destination Flows Using Mobile Phone Location Data." In: *IEEE Pervasive Computing*, pp. 36–44 (cit. on p. 116).

Callier, F. M. and C. A. Desoer (2012). *Linear system theory*. Springer Science & Business Media (cit. on p. 216).

Cameron, G. D. and G. I. Duncan (1996). "PARAMICS–Parallel microscopic simulation of road traffic." In: *The Journal of Supercomputing* 10.1, pp. 25–53 (cit. on p. 101).

Candia, J., M. González, P. Wang, T. Schoenharl, G. Madey, and A.-L. Barabási (2008). "Uncovering individual and collective human dynamics from mobile phone records." In: *Journal of Physics A: Mathematical and Theoretical* 41 (cit. on p. 116).

Casas, J., J. L. Ferrer, D. Garcia, J. Perarnau, and A. Torday (2010). "Traffic Simulation with Aimsun." In: *Fundamentals of traffic simulation*. Springer, pp. 173–232 (cit. on p. 101).

Castillo, E., I. Gallego, J. M. Menendez, and A. Rivas (2010). "Optimal Use of Plate-Scanning Resources for Route Flow Estimation in Traffic Networks." In: *IEEE Trans. on Intelligent Transportation Systems* 11, pp. 380–391 (cit. on p. 115).

Castillo, E., J. M. Menendez, and P. Jimenez (2008). "Trip matrix and path flow reconstruction and estimation based on plate scanning and link observations." In: *Transportation Research Part B: Methodological* 42, pp. 455–481 (cit. on p. 115).

Chan, N. D. and S. A. Shaheen (2012). "Ridesharing in north america: Past, present, and future." In: *Transport Reviews* 32.1, pp. 93–112 (cit. on p. 166).

Chandra, B. and M. M. Halldorsson (2001). "Greedy local improvement and weighted set packing approximation." In: *Journal of Algorithms* 39.2, pp. 223–240 (cit. on p. 202).

Chen, B. and H. H. Cheng (2010). "A review of the applications of agent technology in traffic and transportation systems." In: *IEEE Transactions on Intelligent Transportation Systems* 11.2, pp. 485–497 (cit. on p. 101).

Chen, Y. F., M. Liu, M. Everett, and J. P. How (2017). "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning." In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, pp. 285–292 (cit. on p. 53).

Chentanez, N., A. G. Barto, and S. P. Singh (2005). "Intrinsically motivated reinforcement learning." In: *Advances in neural information processing systems*, pp. 1281–1288 (cit. on p. 216).

Choe, T., A. Skabardonis, and P. Varaiya (2002). "Freeway performance measurement system (PeMS): an operation tool." In: *81st Annual Meeting Transportation Research Board, Washington, DC* (cit. on p. 134).

Choi, E.-H., F. Zhang, E. Y. Noh, S. Singh, and C.-L. Chen (2008). *Sampling Design Used in the National Motor Vehicle Crash Causation Survey*. Tech. rep. DOT HS 810 930, 2008. Washington, D.C.: National Highway Traffic Safety Administrations National Center

for Statistics and Analysis. URL: http://purl.access.gpo.gov/GPO/LPS98147 (cit. on p. 24).
Chu, H. D., E. Gelman, and E. L. Johnson (1997). "Solving large scale crew scheduling problems." In: *Interfaces in Computer Science and Operations Research*. Springer, pp. 183–194 (cit. on p. 201).
Chung, J., C. Gulcehre, K. Cho, and Y. Bengio (2015). "Gated feedback recurrent neural networks." In: *International Conference on Machine Learning*, pp. 2067–2075 (cit. on p. 86).
CIECA (2007). *Internal project on Eco-driving in category B driver training & the driving test* (cit. on pp. 37, 51).
Corporation, B. (1934). "Four Wheels On Jacks Park Car." In: *Popular Science Monthly*. URL: https://books.google.com/?id=HCgDAAAAMBAJ&pg=PA58&dq=Popular+Science+1931+plane#v=onepage&q&f=true (cit. on p. 82).
Côté, M.-A., Á. Kádár, X. Yuan, B. Kybartas, T. Barnes, E. Fine, J. Moore, M. Hausknecht, L. E. Asri, M. Adada, et al. (2018). "TextWorld: A Learning Environment for Text-based Games." In: *arXiv preprint arXiv:1806.11532* (cit. on p. 216).
Cui, S., B. Seibold, R. Stern, and D. B. Work (2017). "Stabilizing Traffic Flow via a Single Autonomous Vehicle: Possibilities and Limitations." In: *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, pp. 447–453 (cit. on pp. 48, 54).
Cui, Y. and S. S. Ge (2003). "Autonomous vehicle positioning with GPS in urban canyon environments." In: *IEEE Transactions on Robotics and Automation* 19.1, pp. 15–25. ISSN: 1042-296X. DOI: 10.1109/TRA.2002.807557 (cit. on p. 82).
Cynthia Kroll, Shija Lu, Aksel Olsen, and Hing Wong (2016). *Regional Forecast for Plan Bay Area 2040*. Tech. rep. Association of Bay Area Governments. (Visited on 06/07/2016) (cit. on p. 188).
Daganzo, C. F. and Y. Sheffi (1977). "On stochastic models of traffic assignment." In: *Transportation Science* 11, pp. 253–274 (cit. on p. 115).
Daganzo, C. F. (2002). "A behavioral theory of multi-lane traffic flow. Part I: Long homogeneous freeway sections." In: *Transportation Research Part B: Methodological* 36.2, pp. 131–158 (cit. on p. 91).
Dai, H., E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song (2017). "Learning Combinatorial Optimization Algorithms over Graphs." In: *arXiv preprint arXiv:1704.01665* (cit. on p. 103).
Dasgupta, S., C. H. Papadimitriou, and U. Vazirani (2006). *Algorithms*. McGraw-Hill, Inc. (cit. on p. 173).

Daumé, H., J. Langford, and D. Marcu (2009). "Search-based structured prediction." In: *Machine learning* 75.3, pp. 297–325 (cit. on p. 16).
Dayan, P. and G. E. Hinton (1993). "Feudal reinforcement learning." In: *Advances in neural information processing systems*, pp. 271–278 (cit. on p. 216).
Deisenroth, M. and C. E. Rasmussen (2011). "PILCO: A model-based and data-efficient approach to policy search." In: *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472 (cit. on p. 216).
Delorme, X., X. Gandibleux, and J. Rodriguez (2004). "GRASP for set packing problems." In: *European Journal of Operational Research* 153.3, pp. 564–580 (cit. on p. 202).
Department of Transportation Division of Traffic Operations (2016). *High-Occupancy Vehicle Guidelines for Planning, Design, and Operations*. State of California Business, Transportation, and Housing Agency. URL: http://www.dot.ca.gov/trafficops/tm/docs/HOV-Guidelines-English-Edition-Nov2016.pdf (cit. on p. 167).
Dissanayake, M. W. M. G., P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba (2001). "A solution to the simultaneous localization and map building (SLAM) problem." In: *IEEE Transactions on Robotics and Automation* 17.3, pp. 229–241. ISSN: 1042-296X. DOI: 10.1109/70.938381 (cit. on p. 82).
DOT, U. (2016). "National transportation statistics." In: *Bureau of Transportation Statistics, Washington, DC* (cit. on p. 82).
Dowling, R., A. Skabardonis, and V. Alexiadis (2004). *Traffic analysis toolbox volume III: guidelines for applying traffic microsimulation modeling software*. Tech. rep. (cit. on p. 141).
Dowsland, K. A. (1993). "Some experiments with simulated annealing techniques for packing problems." In: *European Journal of Operational Research* 68.3, pp. 389–399 (cit. on p. 202).
Drakunov, S., U. Ozguner, P. Dix, and B. Ashrafi (1995). "ABS control using optimum search via sliding modes." In: *IEEE Transactions on Control Systems Technology* 3.1, pp. 79–85. ISSN: 1063-6536. DOI: 10.1109/87.370698 (cit. on p. 82).
Drakunov, S. V. and V. I. Utkin (1992). "Sliding mode control in dynamic systems." In: *International Journal of Control* 55.4, pp. 1029–1037 (cit. on p. 216).
Dresner, K. and P. Stone (2008). "A multiagent approach to autonomous intersection management." In: *Journal of artificial intelligence research* 31, pp. 591–656 (cit. on p. 54).
Duan, Y., X. Chen, R. Houthooft, J. Schulman, and P. Abbeel (2016). "Benchmarking deep reinforcement learning for continuous control." In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)* (cit. on pp. 55, 63–65, 68, 85, 86, 88).

Duchi, J., S. Gould, and D. Koller (2008a). "Projected subgradient methods for learning sparse gaussians." In: *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence* (cit. on p. 133).

Duchi, J., S. Shalev-Shwartz, Y. Singer, and T. Chandra (2008b). "Efficient Projections onto the l1-Ball for Learning in High Dimensions." In: *Proceedings of the 25th International Conference on Machine Learning* (cit. on p. 118).

Erdmann, J. (2015). "SUMO's Lane-Changing Model." In: *Modeling Mobility with Open Data*. Springer, pp. 105–123 (cit. on p. 106).

Erdmann, J. (2016). *Simulation of Urban MObility - Wiki: Car-Following-Models*. URL: http://sumo.dlr.de/wiki/Car-Following-Models#tau (visited on 11/14/2016) (cit. on p. 102).

Falcone, P., F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat (2007). "Predictive active steering control for autonomous vehicle systems." In: *IEEE Transactions on control systems technology* 15.3, pp. 566–580 (cit. on p. 54).

Falcone, P., H. Eric Tseng, F. Borrelli, J. Asgari, and D. Hrovat (2008). "MPC-based yaw and lateral stabilisation via active front steering and braking." In: *Vehicle System Dynamics* 46.S1, pp. 611–628 (cit. on p. 54).

Farrell, J. and M. Barth (1999). *The global positioning system and inertial navigation*. Vol. 61. Mcgraw-hill New York (cit. on p. 116).

Fellendorf, M. (1994). "VISSIM: A microscopic simulation tool to evaluate actuated signal control including bus priority." In: *64th Institute of Transportation Engineers Annual Meeting*. Springer, pp. 1–9 (cit. on p. 101).

Fellendorf, M. and P. Vortisch (2010). "Microscopic traffic flow simulator VISSIM." In: *Fundamentals of traffic simulation*. Springer, pp. 63–93 (cit. on p. 101).

Fisk, C. (1980). "Some developments in equilibrium traffic assignment." In: *Transportation Research Part B* 14, pp. 243–255 (cit. on p. 115).

Foerster, J., G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson (2017). "Counterfactual Multi-Agent Policy Gradients." In: *arXiv preprint arXiv:1705.08926* (cit. on pp. 58, 70, 77).

Ford, L. R. and D. R. Fulkerson (1962). *Flows in Networks*. Princeton Univ. Press, Princeton, NJ (cit. on pp. 115, 137).

Furuhata, M., M. Dessouky, F. Ordóñez, M.-E. Brunet, X. Wang, and S. Koenig (2013). "Ridesharing: The state-of-the-art and future directions." In: *Transportation Research Part B: Methodological* 57, pp. 28–46 (cit. on pp. 166, 169).

Gandibleux, X., X. Delorme, and V. T?Kindt (2004). "An ant colony optimisation algorithm for the set packing problem." In: *Ant Colony Optimization and Swarm Intelligence*. Springer, pp. 49–60 (cit. on p. 202).

Garavello, M. and B. Piccolli (2006). *Traffic flow on networks: Conservation Laws Models*. Springfield, MO: American Institute of Mathematical Sciences (cit. on p. 82).

Gardner, M. (1970). "Mathematical games: The fantastic combinations of John Conway's new solitaire game "life"." In: *Scientific American* 223.4, pp. 120–123 (cit. on p. 36).

Garey, M. R. and D. S. Johnson (1979). "A Guide to the Theory of NP-Completeness." In: *WH Freemann, New York* 70 (cit. on p. 167).

Gense, N. (2000). *Driving style, fuel consumption and emissions–final report*. Tech. rep. TNO Automotive Technical Report Number 00. OR. VM. 021.1/NG, TNO Automotive (cit. on p. 4).

Gillula, J. H., G. M. Hoffmann, H. Huang, M. P. Vitus, and C. J. Tomlin (2011). "Applications of hybrid reachability analysis to robotic aerial vehicles." In: *The International Journal of Robotics Research* 30.3, pp. 335–354. DOI: 10.1177/0278364910387173. eprint: https://doi.org/10.1177/0278364910387173. URL: https://doi.org/10.1177/0278364910387173 (cit. on p. 26).

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep learning*. MIT Press (cit. on pp. 17, 53).

Gozalvez, J., M. Sepulcre, and R. Bauza (2012). "IEEE 802.11p vehicle to infrastructure communications in urban environments." In: *IEEE Communications Magazine* 50.5, pp. 176–183. ISSN: 0163-6804. DOI: 10.1109/MCOM.2012.6194400 (cit. on p. 24).

Grant, J., W. Schroeer, B. Petersen, and M. O'Neill (2000). *Our built and natural environments: A technical review of the interactions between land use, transportation, and environmental quality*. Tech. rep. EPA 231-R-00-005): US Environmental Protection Agency (cit. on p. 166).

Graves, A. et al. (2012). *Supervised sequence labelling with recurrent neural networks*. Vol. 385. Springer (cit. on p. 55).

Greensmith, E., P. L. Bartlett, and J. Baxter (2004). "Variance reduction techniques for gradient estimates in reinforcement learning." In: *Journal of Machine Learning Research* 5.Nov, pp. 1471–1530 (cit. on pp. 59, 68, 70).

Grotzinger, S. J. and C. Witzgall (1984). "Projection onto Order Simplexes." In: *Applied Mathematics and Optimization* 12, pp. 247–270 (cit. on p. 131).

Gu, S., T. Lillicrap, Z. Ghahramani, R. E. Turner, and S. Levine (2017). "Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic." In: *International Conference on Learning Representations (ICLR2017)* (cit. on pp. 64, 68).

Hartigan, J. A. and M. A. Wong (1979). "Algorithm AS 136: A k-means clustering algorithm." In: *Applied statistics*, pp. 100–108 (cit. on p. 202).

Hartman, I. B.-A., D. Keren, A. A. Dbai, E. Cohen, L. Knapen, D. Janssens, et al. (2014). "Theory and practice in large carpooling problems." In: *Procedia Computer Science* 32, pp. 339–347 (cit. on p. 170).

Hassin, R. and S. Rubinstein (2006). "An approximation algorithm for maximum triangle packing." In: *Discrete Applied Mathematics* 154.6, pp. 971–979 (cit. on p. 202).

Hatipoglu, C., U. Ozguner, and K. A. Redmill (2003). "Automated lane change controller design." In: *IEEE Transactions on Intelligent Transportation Systems* 4.1, pp. 13–22. ISSN: 1524-9050. DOI: 10.1109/TITS.2003.811644 (cit. on p. 82).

Hato, E., M. Taniguchi, Y. Sugie, M. Kuwahara, and H. Morita (1999). "Incorporating an information acquisition process into a route choice model with multiple information sources." In: *Transportation Research Part C* 7, pp. 109–129 (cit. on p. 115).

Hausknecht, M., P. Khandelwal, R. Miikkulainen, and P. Stone (2012). "HyperNEAT-GGP: A HyperNEAT-based Atari general game player." In: *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. ACM, pp. 217–224 (cit. on p. 18).

Haykin, S. (1994). *Neural networks: a comprehensive foundation*. Prentice Hall PTR (cit. on p. 86).

Hebrard, E., E. OMahony, and B. OSullivan (2010). "Constraint programming and combinatorial optimisation in numberjack." In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer, pp. 181–185. (Visited on 06/06/2016) (cit. on p. 186).

Heess, N., G. Wayne, D. Silver, T. Lillicrap, T. Erez, and Y. Tassa (2015). "Learning continuous control policies by stochastic value gradients." In: *Advances in Neural Information Processing Systems*, pp. 2944–2952 (cit. on pp. 18, 37, 83).

Heinecke, A., W. Eckhardt, M. Horsch, and H.-J. Bungartz (2015). *Supercomputing for Molecular Dynamics Simulations: Handling Multi-Trillion Particles in Nanofluidics*. Springer (cit. on p. 217).

Herbawi, W. M. and M. Weber (2012). "A genetic and insertion heuristic algorithm for solving the dynamic ridematching problem with time windows." In: *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. ACM, pp. 385–392 (cit. on p. 169).

Herrera, J.-C., D. B. Work, R. Herring, J. Ban, Q. Jacobson, and A. M. Bayen (2009). "Evaluation of traffic data obtained via GPS-enabled mobile phones: the Mobile Century experiment." In: *Transportation Research Part C* 18, pp. 568–583 (cit. on p. 115).

Hoffman, K. L. and M. Padberg (1993). "Solving airline crew scheduling problems by branch-and-cut." In: *Management Science* 39.6, pp. 657–682 (cit. on pp. 201, 202).

Horn, B. K. (2013). "Suppressing traffic flow instabilities." In: *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*. IEEE, pp. 13–20 (cit. on pp. 100, 106).

Horni A., K. N. and K. A. (eds.) (2016). *The Multi-Agent Transport Simulation MATSim*. Ubiquity, London (cit. on pp. 101, 188).

Horvitz, E., J. Apacible, R. Sarin, and L. Liao (2005). "Prediction, Expectation, and Surprise: Methods, Designs, and Study of a Deployed Traffic Forecasting Service." In: *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*. UAI'05. Edinburgh, Scotland: AUAI Press, pp. 275–283. ISBN: 0-9749039-1-4. URL: http://dl.acm.org/citation.cfm?id=3020336.3020371 (cit. on p. 115).

Hoshino, S., J. Ota, A. Shinozaki, and H. Hashimoto (2007). "Hybrid Design Methodology and Cost-Effectiveness Evaluation of AGV Transportation Systems." In: *IEEE Transactions on Automation Science and Engineering* 4.3, pp. 360–372. ISSN: 1545-5955. DOI: 10.1109/TASE.2006.887162 (cit. on p. 166).

Howard, R. A. (1964). "Dynamic programming and Markov processes." In: (cit. on p. 86).

Hunter, T., R. Herring, P. Abbeel, and A. Bayen (2009). "Path and travel time inference from GPS probe vehicle data." In: *NIPS Analyzing Networks and Learning with Graphs* (cit. on p. 115).

Illenberger J., G. F. and K. Nagel (2007). "Enhancing MATSim with capabilities of within-day re-planning." In: *IEEE Intelligent Transportation Systems Conference* (cit. on p. 139).

Ioannou, P. (2013). *Automated highway systems*. Springer Science & Business Media (cit. on p. 25).

Ioannou, P. A. and C.-C. Chien (1993). "Autonomous intelligent cruise control." In: *IEEE Trans. on Vehicular technology* 42.4, pp. 657–672 (cit. on pp. 53, 100).

Ioannou, P. A. and M. Stefanovic (2005). "Evaluation of ACC vehicles in mixed traffic: Lane change effects and sensitivity analysis." In: *IEEE Transactions on Intelligent Transportation Systems* 6.1, pp. 79–89 (cit. on p. 100).

Jaakkola, T., M. I. Jordan, and S. P. Singh (1994). "Convergence of stochastic iterative dynamic programming algorithms." In: *Advances in neural information processing systems*, pp. 703–710 (cit. on p. 18).

Janecek, A., A. A. Hummel, D. Valerio, F. Ricciato, and H. Hlavacs (2012). "Cellular data meet vehicular traffic theory: location area updates and cell transitions for travel time estimation." In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, pp. 361–370 (cit. on p. 116).

Jiang, S., G. A. Fiore, Y. Yang, J. Ferreira Jr, E. Frazzoli, and M. C. González (2013). "A review of urban computing for mobile phone traces: current methods, challenges and opportunities." In: *Proc. of the 2nd ACM SIGKDD International Workshop on Urban Computing*. ACM, p. 2 (cit. on p. 116).

Jin, I. G. and G. Orosz (2014). "Dynamics of connected vehicle systems with delayed acceleration feedback." In: *Transportation Research Part C: Emerging Technologies* 46, pp. 46–64 (cit. on pp. 54, 100, 105).

Jin, W.-L. (2010). "A kinematic wave theory of lane-changing traffic flow." In: *Transportation Research Part B: Methodological* 44.8-9, pp. 1001–1021. DOI: 10.1016/j.trb.2009.12.014 (cit. on p. 41).

Jones, C. and D. M. Kammen (2014). "Spatial distribution of US household carbon footprints reveals suburbanization undermines greenhouse gas benefits of urban population density." In: *Environmental science & technology* 48.2, pp. 895–902 (cit. on p. 166).

Kahn, G., A. Villaflor, V. Pong, P. Abbeel, and S. Levine (2017). "Uncertainty-aware reinforcement learning for collision avoidance." In: *arXiv preprint arXiv:1702.01182* (cit. on p. 53).

Kakade, S. M. (2002). "A natural policy gradient." In: *Advances in neural information processing systems*, pp. 1531–1538 (cit. on pp. 18, 57, 68).

Kakade, S. M. et al. (2003). "On the sample complexity of reinforcement learning." Doctoral dissertation. University of London London, England (cit. on p. 216).

Kamal, M. A. S., J.-i. Imura, T. Hayakawa, A. Ohata, and K. Aihara (2014). "Smart driving of a vehicle using model predictive control for improving traffic flow." In: *IEEE Transactions on Intelligent Transportation Systems* 15.2, pp. 878–888 (cit. on pp. 54, 100).

Kamar, E. and E. Horvitz (2009). "Collaboration and Shared Plans in the Open World: Studies of Ridesharing." In: *IJCAI*. Vol. 9, p. 187 (cit. on pp. 169, 209).

Kanatani, K. and K. Watanabe (1990). "Reconstruction of 3-D road geometry from images for autonomous land vehicles." In: *IEEE Transactions on Robotics and Automation* 6.1, pp. 127–132. ISSN: 1042-296X. DOI: 10.1109/70.88128 (cit. on p. 82).

Karlaftis, M. G. and E. I. Vlahogianni (2011). "Statistical methods versus neural networks in transportation research: Differences, similarities and some insights." In: *Transportation Research Part C: Emerging Technologies* 19.3, pp. 387–399 (cit. on pp. 53, 101).

Kelly, F. P. (1991). "Network routing." In: *Philosophical Trans.: Physical Sciences and Engineering* 337, pp. 343–367 (cit. on p. 138).

Kesting, A. et al. (2007). "Jam-avoiding adaptive cruise control (ACC) and its impact on traffic dynamics." In: *Traffic and Granular Flow05*. Springer, pp. 633–643 (cit. on p. 54).

Khalil, H. K. (1996). "Noninear Systems." In: *Prentice-Hall, New Jersey* 2.5, pp. 5–1 (cit. on pp. 47, 216).
Klar, A. and R. Wegener (1998). "A hierarchy of models for multilane vehicular traffic I: Modeling." In: *SIAM Journal on Applied Mathematics* 59.3, pp. 983–1001 (cit. on p. 91).
Kober, J., J. A. Bagnell, and J. Peters (2013). "Reinforcement learning in robotics: A survey." In: *The International Journal of Robotics Research* 32.11, pp. 1238–1274 (cit. on p. 16).
Koch, T., T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, et al. (2011). "MIPLIB 2010." In: *Mathematical Programming Computation* 3.2, p. 103 (cit. on p. 20).
Konda, V. R. and J. N. Tsitsiklis (2000). "Actor-critic algorithms." In: *Advances in neural information processing systems*, pp. 1008–1014 (cit. on p. 68).
Konishi, H. and S.-i. Mun (2010). "Carpooling and congestion pricing: HOV and HOT lanes." In: *Regional Science and Urban Economics* 40.4, pp. 173–186 (cit. on p. 169).
Korkmaz, G., E. Ekici, F. Özgüner, and Ü. Özgüner (2004). "Urban Multi-hop Broadcast Protocol for Inter-vehicle Communication Systems." In: *Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks*. VANET '04. Philadelphia, PA, USA: ACM, pp. 76–85. ISBN: 1-58113-922-5. DOI: 10.1145/1023875.1023887. URL: http://doi.acm.org/10.1145/1023875.1023887 (cit. on p. 24).
Kotsialos, A., M. Papageorgiou, and A. Messmer (1999). "Optimal coordinated and integrated motorway network traffic control." In: *14th International Symposium on Transportation and Traffic Theory* (cit. on p. 84).
Krajzewicz, D., J. Erdmann, M. Behrisch, and L. Bieker (2012). "Recent Development and Applications of SUMO-Simulation of Urban MObility." In: *International Journal On Advances in Systems and Measurements* 5.3&4 (cit. on pp. 85, 88, 101).
Kutiel, G. (2016). "Approximation Algorithms for the Maximum Carpool Matching Problem." In: *arXiv preprint arXiv:1604.05609* (cit. on p. 170).
Lai, M. (2015). "Giraffe: Using deep reinforcement learning to play chess." In: *arXiv preprint arXiv:1509.01549* (cit. on pp. 37, 83).
Lasky, T. A., S. M. Donecker, K. S. Yen, and B. Ravani (2004). *Intelligent ultra high speed distributed sensing system and method for sensing roadway markers for intelligent vehicle guidance and control*. US Patent 6,772,062 (cit. on p. 26).
LeBlanc, L. J., E. K. Morlok, and W. P. Pierskalla (1975). "An efficient approach to solving the road network equilibrium traffic assignment problem." In: *Transportation Research* 9, pp. 309–318 (cit. on p. 133).

LeCun, Y. (1985). "Une procedure d'apprentissage ponr reseau a seuil asymetrique (a learning scheme for asymmetric threshold networks)." In: *proceedings of Cognitiva 85*, pp. 599–604 (cit. on p. 17).
LeCun, Y., Y. Bengio, and G. Hinton (2015). "Deep learning." In: *nature* 521.7553, p. 436 (cit. on p. 17).
Lee Jr, D., L. Klein, and G. Camus (1999). "Induced traffic and induced demand." In: *Transportation Research Record: Journal of the Transportation Research Board* 1659, pp. 68–75 (cit. on p. 151).
Lee, J., M. Park, and H. Yeo (2016). "A probability model for discretionary lane changes in highways." In: *KSCE Journal of Civil Engineering* 20.7, pp. 2938–2946 (cit. on p. 83).
Lee, J., J. Choi, K. Yi, M. Shin, and B. Ko (2014). "Lane-keeping assistance control algorithm using differential braking to prevent unintended lane departures." In: *Control Engineering Practice* 23, pp. 1–13 (cit. on p. 82).
Lefevre, S., Y. Gao, D. Vasquez, H. E. Tseng, R. Bajcsy, and F. Borrelli (2014). "Lane keeping assistance with learning-based driver model and model predictive control." In: *12th International Symposium on Advanced Vehicle Control* (cit. on p. 82).
Levine, S., C. Finn, T. Darrell, and P. Abbeel (2016). "End-to-end training of deep visuomotor policies." In: *Journal of Machine Learning Research* 17.1, pp. 1334–1373 (cit. on pp. 17, 44, 57, 68, 216).
Levinson, D. M. and K. J. Krizek (2015). *The End of Traffic & the Future of Transport*. David M. Levinson (cit. on p. 30).
Li, F. and Y. Wang (2007). "Routing in vehicular ad hoc networks: A survey." In: *IEEE Vehicular technology magazine* 2.2 (cit. on p. 24).
Li, L., Y. Lv, and F. Wang (2016). "Traffic signal timing via deep reinforcement learning." In: *IEEE/CAA Journal of Automatica Sinica* 3.3, pp. 247–254. ISSN: 2329-9266. DOI: 10.1109/JAS.2016.7508798 (cit. on p. 101).
Li, Y. (2017). "Deep reinforcement learning: An overview." In: *arXiv preprint arXiv:1701.07274* (cit. on p. 17).
Liang, C.-Y. and H. Peng (1999). "Optimal adaptive cruise control with guaranteed string stability." In: *Vehicle system dynamics* 32.4-5, pp. 313–330 (cit. on p. 54).
Liang, C.-Y. and H. Peng (2000). "String stability analysis of adaptive cruise controlled vehicles." In: *JSME International Journal Series C Mechanical Systems, Machine Elements and Manufacturing* 43.3, pp. 671–677 (cit. on pp. 54, 100).
Liang, E., R. Liaw, R. Nishihara, P. Moritz, R. Fox, J. Gonzalez, K. Goldberg, and I. Stoica (2017). "Ray RLLib: A Composable and Scalable Reinforcement Learning Library." In: *arXiv preprint arXiv:1712.09381* (cit. on pp. 85, 88).

Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra (2016). "Continuous control with deep reinforcement learning." In: *International Conference on Learning Representations (ICLR2016)* (cit. on pp. 17, 18, 57, 64).

Lint, J. van, H. J. van Zuylen, and H. Tu (2008). "Travel time unreliability on freeways: Why measures based on variance tell only half the story." In: *Transportation Research Part A: Policy and Practice* 42.1, pp. 258–277. ISSN: 0965-8564. DOI: https://doi.org/10.1016/j.tra.2007.08.008. URL: http://www.sciencedirect.com/science/article/pii/S0965856407000742 (cit. on p. 55).

Litman, T. (2017). *Generated traffic and induced travel*. Victoria Transport Policy Institute (cit. on p. 151).

Lloyd, S. (1982). "Least squares quantization in PCM." In: *IEEE transactions on information theory* 28.2, pp. 129–137 (cit. on p. 207).

Lowe, R., Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch (2017). "Multi-agent actor-critic for mixed cooperative-competitive environments." In: *Advances in Neural Information Processing Systems*, pp. 6379–6390 (cit. on pp. 58, 68, 103, 216).

Lu, X., S. Shladover, and J. Hedrick (2004). "Heavy-duty truck control: Short inter-vehicle distance following." In: *American Control Conference, 2004. Proceedings of the 2004*. Vol. 5. IEEE, pp. 4722–4727 (cit. on p. 53).

Lv, Y., Y. Duan, W. Kang, Z. Li, and F.-Y. Wang (2015). "Traffic flow prediction with big data: a deep learning approach." In: *IEEE Transactions on Intelligent Transportation Systems* 16.2, pp. 865–873 (cit. on pp. 53, 101).

Ma, S. and O. Wolfson (2013). "Analysis and evaluation of the slugging form of ridesharing." In: *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, pp. 64–73 (cit. on p. 170).

Maher, M. J. and P. C. Hughes (1997). "A probit-based stochastic user equilibrium assignment model." In: *Transportation Research* 31, pp. 341–355 (cit. on p. 115).

Maniezzo, A. C. M. D. V. (1992). "Distributed optimization by ant colonies." In: *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*. MIT Press, p. 134 (cit. on p. 36).

Mardani, M. and G. B. Giannakis (2013). "Robust network traffic estimation via sparsity and low rank." In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (cit. on p. 117).

Martin, E. W., K. Boriboonsomsin, N. D. Chan, N. Williams, S. A. Shaheen, and M. Barth (2013). "Dynamic ecodriving in Northern California: A Study of survey and vehicle operations data from an ecodriving feedback device." In: *92nd Annual Meeting of the Transportation Research Board, Washington, DC, January* (cit. on p. 100).

Mashaw, J. L. and D. L. Harfst (1990). *The struggle for auto safety*. Harvard University Press Cambridge, MA (cit. on p. 31).

Mathew, J. and P. Xavier (2014). "A survey on using wireless signals for road traffic detection." In: *International Journal of Research in Engineering and Technology* 3.1 (cit. on p. 117).

Meyer, G. and S. Shaheen, eds. (2017). *Disrupting Mobility: Impacts of Sharing Economy and Innovative Transportation on Cities*. Springer (cit. on p. 83).

Miao, F., S. Han, S. Lin, J. A. Stankovic, D. Zhang, S. Munir, H. Huang, T. He, and G. J. Pappas (2016). "Taxi Dispatch With Real-Time Sensing Data in Metropolitan Areas: A Receding Horizon Control Approach." In: *IEEE Transactions on Automation Science and Engineering* 13.2, pp. 463–478. ISSN: 1545-5955. DOI: 10.1109/TASE.2016.2529580 (cit. on pp. 166, 175).

Michalopoulos, P. G., D. E. Beskos, and Y. Yamauchi (1984). "Multilane traffic flow dynamics: some macroscopic considerations." In: *Transportation Research Part B: Methodological* 18.4, pp. 377–395 (cit. on p. 91).

Miculescu, D. and S. Karaman (2014). "Polling-systems-based control of high-performance provably-safe autonomous intersections." In: *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, pp. 1417–1423 (cit. on p. 54).

Milans, V., D. F. Llorca, J. Villagr, J. Prez, C. Fernndez, I. Parra, C. Gonzlez, and M. A. Sotelo (2012). "Intelligent automatic overtaking system using vision for vehicle detection." In: *Expert Systems with Applications* 39.3, pp. 3362–3373. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2011.09.024. URL: http://www.sciencedirect.com/science/article/pii/S0957417411013339 (cit. on p. 82).

Minguez, J. and L. Montano (2009). "Extending Collision Avoidance Methods to Consider the Vehicle Shape, Kinematics, and Dynamics of a Mobile Robot." In: *IEEE Transactions on Robotics* 25.2, pp. 367–381. ISSN: 1552-3098. DOI: 10.1109/TRO.2009.2011526 (cit. on p. 82).

Mitchell, I. M., A. M. Bayen, and C. J. Tomlin (2005). "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games." In: *IEEE Transactions on automatic control* 50.7, pp. 947–957 (cit. on p. 26).

Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu (2016). "Asynchronous methods for deep reinforcement learning." In: *International Conference on Machine Learning*, pp. 1928–1937 (cit. on pp. 57, 68).

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller (2013). "Playing atari with deep reinforcement learning." In: *arXiv preprint arXiv:1312.5602* (cit. on pp. 17, 37, 83).

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. (2015). "Human-level control through deep reinforcement learning." In: *Nature* 518.7540, pp. 529–533 (cit. on pp. 17, 18, 57, 58).

Monderer, D. and L. S. Shapley (1996). "Potential Games." In: *Games and Economic Behavior* 14, pp. 124–143 (cit. on p. 126).

Montgomery, W. H. and S. Levine (2016). "Guided policy search via approximate mirror descent." In: *Advances in Neural Information Processing Systems*, pp. 4008–4016 (cit. on p. 67).

Mordatch, I. and P. Abbeel (2017). "Emergence of Grounded Compositional Language in Multi-Agent Populations." In: *arXiv preprint arXiv:1703.04908* (cit. on pp. 52, 55).

Mordatch, I., K. Lowrey, G. Andrew, Z. Popovic, and E. V. Todorov (2015). "Interactive control of diverse complex characters with neural networks." In: *Advances in Neural Information Processing Systems*, pp. 3132–3140 (cit. on p. 68).

Moritz, P., R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, W. Paul, M. I. Jordan, and I. Stoica (2017). "Ray: A Distributed Framework for Emerging AI Applications." In: *arXiv preprint arXiv:1712.05889* (cit. on p. 85).

Nagabandi, A., G. Kahn, R. S. Fearing, and S. Levine (2017). "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning." In: *arXiv preprint arXiv:1708.02596* (cit. on p. 216).

Narayanamurthy, S. M. and B. Ravindran (2008). "On the hardness of finding symmetries in Markov decision processes." In: *Proceedings of the 25th international conference on Machine learning*. ACM, pp. 688–695 (cit. on p. 43).

Naus, G. J., R. P. Vugts, J. Ploeg, M. J. van de Molengraft, and M. Steinbuch (2010). "String-stable CACC design and experimental validation: A frequency-domain approach." In: *IEEE Trans. on Vehicular Technology* 59.9, pp. 4268–4279 (cit. on p. 54).

Ng, A. Y., D. Harada, and S. Russell (1999). "Policy invariance under reward transformations: Theory and application to reward shaping." In: *ICML*. Vol. 99, pp. 278–287 (cit. on p. 216).

Nocedal, J. and S. Wright (2006). *Numerical Optimization*. Springer, 2nd edition (cit. on pp. 13, 133).

Oh, S. and H. Yeo (2015). "Impact of stop-and-go waves and lane changes on discharge rate in recovery flow." In: *Transportation Research Part B: Methodological* 77, pp. 88–102 (cit. on p. 41).

Orosz, G., J. Moehlis, and F. Bullo (2011). "Delayed car-following dynamics for human and robotic drivers." In: *ASME 2011 International Design Engineering Technical Con-*

ferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, pp. 529–538 (cit. on p. 100).
Orosz, G., R. E. Wilson, and G. Stépán (2010). "Traffic jams: dynamics and control." In: *Philosophical Trans. of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 368.1928, pp. 4455–4479 (cit. on pp. 100, 104).
Ortuzar, J. d. D. and L. Willumsen (2001). *Modelling Transport*. 3rd, Edition, Wiley, West Sussex, United Kingdom (cit. on pp. 117, 133).
Padberg, M. W. (1973). "On the facial structure of set packing polyhedra." In: *Mathematical programming* 5.1, pp. 199–215 (cit. on p. 202).
Papadimitriou, C. H. and K. Steiglitz (1998). *Combinatorial optimization: algorithms and complexity*. Courier Corporation (cit. on pp. 19, 167).
Papageorgiou, M., C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang (2003). "Review of road traffic control strategies." In: *Proceedings of the IEEE* 91.12, pp. 2043–2067 (cit. on pp. 37, 100).
Parker, D. B. (1985). "Learning logic." In: (cit. on p. 17).
Paromtchik, I. E. and C. Laugier (1996). "Motion generation and control for parking an autonomous vehicle." In: *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*. Vol. 4. IEEE, pp. 3117–3122 (cit. on p. 82).
Parsons Transportation Group, Inc. (2002). *High Occupancy Lanes and Value Lanes Study Final Report: High Occupancy Vehicle Facilities Policy Guidelines and Plan for the MAG Freeway System*. §2-2. Arizona Department of Transportation; Maricopa Association of Governments; Regional Public Transportation Authority (cit. on p. 158).
Patire, A., M. Wright, B. Prodhomme, and A. Bayen (2013). "How much GPS data do we need?" In: *Transportation Research Part C* (cit. on p. 116).
Pavone, M., S. L. Smith, E. Frazzoli, and D. Rus (2012). "Robotic load balancing for mobility-on-demand systems." In: *The International Journal of Robotics Research* 31.7, pp. 839–854 (cit. on p. 82).
Peng, P., Q. Yuan, Y. Wen, Y. Yang, Z. Tang, H. Long, and J. Wang (2017). "Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games." In: *arXiv preprint arXiv:1703.10069* (cit. on p. 52).
Peters, J. and S. Schaal (2008). "Natural actor-critic." In: *Neurocomputing* 71.7, pp. 1180–1190 (cit. on p. 68).
Polson, N. G. and V. O. Sokolov (2017). "Deep learning for short-term traffic flow prediction." In: *Transportation Research Part C: Emerging Technologies* 79, pp. 1–17 (cit. on pp. 53, 101).

Pozdnoukhov, A., A. Campbell, S. Feygin, M. Yin, and S. Mohanty (2016). "The SmartBay Project: Connected Mobility in San Francisco Bay Area." In: *The Multi-Agent Transport Simulation MATSim*. Ed. by N. K. Horni A. and K. Axhausen. Ubiquity, London. Chap. 83, pp. 541–546 (cit. on p. 188).

Rahimi, A. and B. Recht (2007). "Random Features for Large-Scale Kernel Machines." In: *NIPS* (cit. on p. 65).

Rahmani, M. and H. N. Koutsopoulos (2013). "Path inference from sparse floating car data." In: *Transportation Research Part C: Emerging Technologies* 30, pp. 41–54 (cit. on p. 115).

Rajamani, R. and C. Zhu (2002). "Semi-autonomous adaptive cruise control systems." In: *IEEE Trans. on Vehicular Technology* 51.5, pp. 1186–1192 (cit. on p. 53).

Rajeswaran, A., V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine (2017a). "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations." In: *CoRR* abs/1709.10087 (cit. on pp. 57, 64).

Rajeswaran, A., K. Lowrey, E. Todorov, and S. Kakade (2017b). "Towards Generalization and Simplicity in Continuous Control." In: *NIPS* (cit. on pp. 64, 65).

Ravani, B. and K. S. Yen (2000). *Development of an Advanced Snowplow Driver Assistance System (ASP-II)*. California AHMCT Program (cit. on p. 26).

Reynolds, C. W. (1987). "Flocks, herds and schools: A distributed behavioral model." In: *ACM SIGGRAPH computer graphics* 21.4, pp. 25–34 (cit. on p. 36).

Rios-Torres, J. and A. A. Malikopoulos (2017a). "A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps." In: *IEEE Transactions on Intelligent Transportation Systems* 18.5, pp. 1066–1077 (cit. on p. 83).

Rios-Torres, J. and A. A. Malikopoulos (2017b). "Automated and cooperative vehicle merging at highway on-ramps." In: *IEEE Transactions on Intelligent Transportation Systems* 18.4, pp. 780–789 (cit. on p. 83).

Roughgarden, T. (2003). "The price of anarchy is independent of the network topology." In: *Journal of Computer and System Sciences* 67, pp. 341–364 (cit. on p. 138).

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). "Learning representations by back-propagating errors." In: *Nature* 323.6088, p. 533 (cit. on p. 17).

Sadigh, D., N. Landolfi, S. S. Sastry, S. A. Seshia, and A. D. Dragan (2016). "Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state." In: *Autonomous Robots*, pp. 1–22 (cit. on p. 83).

Samet, H. and J. Sankaranarayanan (2014). *Path oracles for spatial networks*. US Patent 8,744,770 (cit. on p. 189).

Sanchez-Medina, J. J., M. J. Galan-Moreno, and E. Rubio-Royo (2010). "Traffic Signal Optimization in La Almozara District in Saragossa Under Congestion Conditions, Using Genetic Algorithms, Traffic Microsimulation, and Cluster Computing." In: *IEEE Transactions on Intelligent Transportation Systems* 11.1, pp. 132–141. ISSN: 1524-9050. DOI: 10.1109/TITS.2009.2034383 (cit. on p. 102).

Santi, P., G. Resta, M. Szell, S. Sobolevsky, S. Strogatz, and C. Ratti (2013). "Taxi pooling in New York City: a network-based approach to social sharing problems." In: *arXiv preprint arXiv* 310 (cit. on p. 201).

Sasoh, A. and T. Ohara (2002). "Shock wave relation containing lane change source term for two-lane traffic flow." In: *Journal of the Physical Society of Japan* 71.9, pp. 2339–2347 (cit. on p. 91).

Schrank, D., B. Eisele, and T. Lomax (2012). "TTI's 2012 urban mobility report." In: *Texas A&M Transportation Inst. The Texas A&M Univ. System* (cit. on p. 82).

Schrank, D., B. Eisele, T. Lomax, and J. Bak (2015). *Urban Mobility Scorecard*. Tech. rep. Technical Report August, Texas A&M Transportation Inst. and INRIX, Inc (cit. on p. 166).

Schulman, J. (2016). "Optimizing expectations: From deep reinforcement learning to stochastic computation graphs." Doctoral dissertation. UC Berkeley (cit. on p. 17).

Schulman, J., S. Levine, P. Abbeel, M. Jordan, and P. Moritz (2015). "Trust region policy optimization." In: *International Conference on Machine Learning*, pp. 1889–1897 (cit. on pp. 17, 37, 55, 57, 68, 83, 85, 86).

Schulman, J., P. Moritz, S. Levine, M. Jordan, and P. Abbeel (2016). "High-Dimensional Continuous Control Using Generalized Advantage Estimation." In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on pp. 17, 18, 57, 61, 68, 76, 78, 83, 85).

Selker, T. and P. H. Saphir (2010). "TravelRole: A carpooling/physical social network creator." In: *Collaborative Technologies and Systems (CTS), 2010 International Symposium on*. IEEE, pp. 629–634 (cit. on p. 167).

Sheffi, Y. (1985). *Urban Transportation Networks*. Prentice-Hall, Englewood Cliffs, NJ (cit. on pp. 115, 117, 126, 137).

Sheikholeslam, S. and C. A. Desoer (1992). "A system level study of the longitudinal control of a platoon of vehicles." In: *Journal of dynamic systems, measurement, and control* 114.2, pp. 286–292 (cit. on p. 53).

Shen, W. and L. Wynter (2012). "A new one-level convex optimization approach for estimating origin?destination demand." In: *Transportation Research Part B: Methodological* 46, pp. 1535–1555 (cit. on p. 117).

Shiller, Z. and Y. R. Gwo (1991). "Dynamic motion planning of autonomous vehicles." In: *IEEE Transactions on Robotics and Automation* 7.2, pp. 241–249. ISSN: 1042-296X. DOI: 10.1109/70.75906 (cit. on p. 82).

Shladover, S. E. (2005). "Automated vehicles for highway operations (automated highway systems)." In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 219.1, pp. 53–75 (cit. on pp. 25, 53).

Shladover, S. E. (2007). "PATH at 20History and major milestones." In: *IEEE Transactions on intelligent transportation systems* 8.4, pp. 584–592 (cit. on p. 25).

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. (2016). "Mastering the game of Go with deep neural networks and tree search." In: *Nature* 529.7587, pp. 484–489 (cit. on pp. 17, 57, 58).

Simonin, G. and B. O'Sullivan (2014). "Optimisation for the Ride-Sharing Problem: a Complexity-based Approach." In: *ECAI*, pp. 831–836 (cit. on p. 170).

Son, Y. S., W. Kim, S.-H. Lee, and C. C. Chung (2015). "Robust multirate control scheme with predictive virtual lanes for lane-keeping system of autonomous highway driving." In: *IEEE Transactions on Vehicular Technology* 64.8, pp. 3378–3391 (cit. on p. 82).

Spieser, K., K. Treleaven, R. Zhang, E. Frazzoli, D. Morton, and M. Pavone (2014). "Toward a Systematic Approach to the Design and Evaluation of Automated Mobility-on-Demand Systems: A Case Study in Singapore." In: *Road Vehicle Automation*. Springer, pp. 229–245 (cit. on p. 30).

Steinhaus, H. (1956). "Sur la division des corp materiels en parties." In: *Bull. Acad. Polon. Sci* 1.804, p. 801 (cit. on p. 207).

Stern, R. E., S. Cui, M. L. D. Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, R. Haulcy, H. Pohlmann, F. Wu, B. Piccoli, B. Seibold, J. Sprinkle, and D. B. Work (2017). "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments." In: *CoRR* abs/1705.01693. URL: http://arxiv.org/abs/1705.01693 (cit. on pp. 37, 48, 51, 82, 84, 85, 94–96, 99).

Stevens, M. and C. Yeh (2016). *Reinforcement Learning for Traffic Optimization*. Tech. rep. Stanford University. URL: http://cs229.stanford.edu/proj2016spr/report/047.pdf (cit. on p. 53).

Sugiyama, Y., M. Fukui, M. Kikuchi, K. Hasebe, A. Nakayama, K. Nishinari, S.-i. Tadaki, and S. Yukawa (2008). "Traffic jams without bottlenecks–experimental evidence for

the physical mechanism of the formation of a jam." In: *New Journal of Physics* 10.3, p. 033001 (cit. on pp. 27, 44, 45, 51, 82, 83, 85, 90, 93, 95).

Sukkarieh, S., E. M. Nebot, and H. F. Durrant-Whyte (1999). "A high integrity IMU/GPS navigation loop for autonomous land vehicle applications." In: *IEEE Transactions on Robotics and Automation* 15.3, pp. 572–578. ISSN: 1042-296X. DOI: 10.1109/70.768189 (cit. on p. 82).

SUMO Team (2016). *Simulation/Basic Definition*. URL: http://sumo.dlr.de/wiki/Simulation/Basic_Definition#Defining_the_Time_Step_Length (visited on 12/08/2016) (cit. on p. 102).

Sun, Z. and X. (Ban (2013). "Vehicle classification using GPS data." In: *Transportation Research Part C: Emerging Technologies* 37, pp. 102–117. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2013.09.015. URL: http://www.sciencedirect.com/science/article/pii/S0968090X13002040 (cit. on p. 115).

Sutton, R. S. and A. G. Barto (1998). *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge (cit. on pp. 16, 68).

Sutton, R. S., D. A. McAllester, S. P. Singh, and Y. Mansour (2000). "Policy gradient methods for reinforcement learning with function approximation." In: *Advances in neural information processing systems*, pp. 1057–1063 (cit. on pp. 18, 39, 57, 59, 68, 86).

Swaroop, D., J. Hedrick, C. C. Chien, and P. Ioannou (1994). "A Comparision of Spacing and Headway Control Laws for Automatically Controlled Vehicles." In: *Vehicle System Dynamics* 23.1, pp. 597–625. DOI: 10.1080/00423119408969077. eprint: https://doi.org/10.1080/00423119408969077. URL: https://doi.org/10.1080/00423119408969077 (cit. on p. 54).

Swaroop, D. (1997). "String stability of interconnected systems: An application to platooning in automated highway systems." In: *California Partners for Advanced Transit and Highways (PATH)* (cit. on p. 100).

Synnaeve, G., N. Nardelli, A. Auvolat, S. Chintala, T. Lacroix, Z. Lin, F. Richoux, and N. Usunier (2016). "TorchCraft: a Library for Machine Learning Research on Real-Time Strategy Games." In: *arXiv preprint arXiv:1611.00625* (cit. on pp. 83, 216).

Szita, I. and A. Lörincz (2006). "Learning Tetris using the noisy cross-entropy method." In: *Neural computation* 18.12, pp. 2936–2941 (cit. on p. 18).

Technical Committee ISO/TC 204, Intelligent transport systems (2010). *Intelligent transport systems – Adaptive Cruise Control systems – Performance requirements and test procedures*. ISO (cit. on pp. 37, 53, 100).

Tesauro, G. (1994). "TD-Gammon, a self-teaching backgammon program, achieves master-level play." In: *Neural computation* 6.2, pp. 215–219 (cit. on p. 17).

Tettamanti, T., H. Demeter, and I. Varga (2012). "Route Choice Estimation Based on Cellular Signaling Data." In: *Acta Polytechnica Hungarica* 9.4, pp. 207–220 (cit. on p. 117).
Thrun, S., M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, et al. (2006). "Stanley: The robot that won the DARPA Grand Challenge." In: *Journal of field Robotics* 23.9, pp. 661–692 (cit. on p. 26).
Tibshirani, R. J., H. Hoefling, and R. Tibshirani (2011). "Nearly-Isotonic Regression." In: *Technometrics* 53 (cit. on pp. 118, 130).
Todorov, E., T. Erez, and Y. Tassa (2012). "MuJoCo: A physics engine for model-based control." In: *International Conference on Intelligent Robots and Systems* (cit. on pp. 64, 83, 216).
Toole, J., M. Ulm, M. González, and D. Bauer (2012). "Inferring land use from mobile phone activity." In: *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*, pp. 1–8 (cit. on p. 116).
Transportation (WisDOT), W. D. of (2013). *Unofficial WI Traffic Analysis Guidelines, Draft*. http://www.wisdot.info/microsimulation/index.php?title=Model_Calibration. [Online; accessed 2014-08-30] (cit. on p. 141).
Treiber, M., A. Hennecke, and D. Helbing (2000). "Congested traffic states in empirical observations and microscopic simulations." In: *Physical review E* 62.2, p. 1805 (cit. on pp. 39, 40, 42, 47, 104, 105).
Treiber, M. and A. Kesting (2013). "Traffic flow dynamics." In: *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg* (cit. on pp. 37, 94, 100, 105).
Treiber, M. and A. Kesting (2017). "The Intelligent Driver Model with Stochasticity-New Insights Into Traffic Flow Oscillations." In: *Transportation Research Procedia* 23, pp. 174–187 (cit. on pp. 42, 94).
Trevisan, L. (2011). *Combinatorial optimization: exact and approximate algorithms*. Stanford University (cit. on p. 19).
Tucker, G., S. Bhupatiraju, S. Gu, R. E. Turner, Z. Ghahramani, and S. Levine (2018). "The mirage of action-dependent baselines in reinforcement learning." In: *arXiv preprint arXiv:1802.10031* (cit. on p. 70).
United Nations (2014). "World urbanization prospects: The 2014 revision, highlights. department of economic and social affairs." In: *Population Division, United Nations*. URL: http://www.un.org/en/development/desa/news/population/world-urbanization-prospects-2014.html (cit. on p. 4).
U.S. Dept. of Transportation, Federal Highway Administration (1977). *Module 6. HOV Treatments. Freeway Management Handbook*. 6-7, 8 (cit. on p. 158).

US Energy Information Administration (2018). *Monthly energy review*. Table 2.1 (cit. on pp. 4, 5).

Vahidi, A. and A. Eskandarian (2003). "Research advances in intelligent collision avoidance and adaptive cruise control." In: *IEEE transactions on intelligent transportation systems* 4.3, pp. 143–153 (cit. on pp. 53, 100).

Van Arem, B., C. J. Van Driel, and R. Visser (2006). "The impact of cooperative adaptive cruise control on traffic-flow characteristics." In: *IEEE Trans. on Intelligent Transportation Systems* 7.4, pp. 429–436 (cit. on pp. 54, 82).

Van Roy, B., D. P. Bertsekas, Y. Lee, and J. N. Tsitsiklis (1997). "A neuro-dynamic programming approach to retailer inventory management." In: *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on*. Vol. 4. IEEE, pp. 4052–4057 (cit. on p. 16).

Veeraraghavan, H., O. Masoud, and N. Papanikolopoulos (2003). "Computer Vision Algorithms for Intersection Monitoring." In: *IEEE Trans. on Intelligent Transportation Systems* 4, pp. 78–89 (cit. on p. 115).

Vemuganti, R. (1999). "Applications of set covering, set packing and set partitioning models: A survey." In: *Handbook of combinatorial optimization*. Springer, pp. 573–746 (cit. on p. 202).

Venkatadri, U., K. S. Krishna, and M. A. lk (2016). "On Physical Internet Logistics: Modeling the Impact of Consolidation on Transportation and Inventory Costs." In: *IEEE Transactions on Automation Science and Engineering* 13.4, pp. 1517–1527. ISSN: 1545-5955. DOI: 10.1109/TASE.2016.2590823 (cit. on p. 166).

Vezhnevets, A. S., S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu (2017). "FeUdal Networks for Hierarchical Reinforcement Learning." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 3540–3549. URL: http://proceedings.mlr.press/v70/vezhnevets17a.html (cit. on p. 216).

Vickrey, W. (1961). "Counterspeculation, auctions, and competitive sealed tenders." In: *The Journal of finance* 16.1, pp. 8–37 (cit. on p. 169).

Vinyals, O. (2016). *DeepMind and Blizzard to release StarCraft II as an AI research environment*. https://deepmind.com/blog/deepmind-and-blizzard-release-starcraft-ii-ai-research-environment/. Blog (cit. on p. 83).

Volinsky, C., R. Becker, R. Caceres, K. Hanson, J. Loh, S. Urbanek, and A. Varshavsky (2011). "Clustering Anonymized Mobile Call Detail Records to Find Usage Groups." In: *1st Workshop on Pervasive Urban Applications (PURBA)* (cit. on p. 116).

Wadud, Z., D. MacKenzie, and P. Leiby (2016). "Help or hindrance? The travel, energy and carbon impacts of highly automated vehicles." In: *Transportation Research Part A: Policy and Practice* 86, pp. 1–18 (cit. on pp. 5, 6, 82).
Wahlström, N., T. B. Schön, and M. P. Deisenroth (2015). "From pixels to torques: Policy learning with deep dynamical models." In: *arXiv preprint arXiv:1502.02251* (cit. on p. 44).
Wampler, K. and Z. Popović (2009). "Optimal gait and form for animal locomotion." In: *ACM Transactions on Graphics (TOG)*. Vol. 28. 3. ACM, p. 60 (cit. on p. 18).
Wang, J. and Q. Feng (2008). "An O*(3.523 k) parameterized algorithm for 3-set packing." In: *Theory and Applications of Models of Computation*. Springer, pp. 82–93 (cit. on p. 202).
Wang, L., B. K. Horn, and G. Strang (2016). "Eigenvalue and Eigenvector Analysis of Stability for a Line of Traffic." In: *Studies in Applied Mathematics* (cit. on pp. 100, 106).
Wang, M., W. Daamen, S. P. Hoogendoorn, and B. van Arem (2014). "Rolling horizon control framework for driver assistance systems. Part I: Mathematical formulation and non-cooperative systems." In: *Transportation research part C: emerging technologies* 40, pp. 271–289 (cit. on p. 54).
Wang, W. and M. A. Carreira-Perpinán (2013). "Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application." In: *arXiv preprint arXiv:1309.1541* (cit. on pp. 118, 133).
Wardrop, J. G. and J. I. Whitehead (1952). "Correspondence. Some Theoretical Aspects of Road Traffic Research." In: *ICE Proc: Engineering Divisions 1* (cit. on pp. 115, 126, 138).
WardsAuto (2017). *World Vehicle Population Rose 4.6% in 2016*. URL: http://subscribers.wardsintelligence.com/analysis/world-vehicle-population-rose-46-2016 (cit. on p. 4).
Washington State Department of Transportation (2017). *High Occupancy Vehicle (HOV) lanes*. URL: http://www.wsdot.wa.gov/HOV/ (cit. on p. 167).
Watkins, C. J. and P. Dayan (1992). "Q-learning." In: *Machine learning* 8.3-4, pp. 279–292 (cit. on p. 68).
Weaver, L. and N. Tao (2001). "The optimal reward baseline for gradient-based reinforcement learning." In: *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., pp. 538–545 (cit. on p. 68).
Weiss, G. (1999). *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press (cit. on p. 52).
Werbos, P. (1974). "Beyond Regression:" New Tools for Prediction and Analysis in the Behavioral Sciences." In: *Ph. D. dissertation, Harvard University* (cit. on p. 17).

White, J. and I. Wells (2002). "Extracting origin destination information from mobile phone data." In: *11th Int. Conf. on Road Transport Information and Control, London*, pp. 30–34 (cit. on p. 116).

Wierstra, D., T. Schaul, J. Peters, and J. Schmidhuber (2008). "Natural evolution strategies." In: *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE, pp. 3381–3387 (cit. on p. 18).

Williams, R. J. (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning." In: *Machine learning* 8.3-4, pp. 229–256 (cit. on pp. 18, 57, 59, 68).

Wolsey, L. A. and G. L. Nemhauser (2014). *Integer and combinatorial optimization*. John Wiley & Sons (cit. on p. 167).

Work, D., O.-P. Tossavainen, S. Blandin, A. Bayen, T. Iwuchukwu, and K. Tracton (2008). "An ensemble Kalman filtering approach to highway traffic estimation using GPS enabled mobile devices." In: *47th IEEE Conference on Decision and Control* (cit. on p. 115).

Wu, C. (2016). "Traffic Jammin': Making automated transportation a reality." In: *Berkeley Science Review* (cit. on p. 8).

Wu, C., A. M. Bayen, and A. Mehta (2017a). "Stabilizing traffic with autonomous vehicles." In: *Submission to IEEE International Conference on Robotics and Automation (ICRA)* (cit. on pp. 54, 100).

Wu, C., E. Kamar, and E. Horvitz (2016a). "Clustering for set partitioning with a case study in ridesharing." In: *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, pp. 1384–1388 (cit. on p. 11).

Wu, C., A. Kreidieh, E. Vinitsky, and A. M. Bayen (2017b). "Multi-lane Reduction: A Stochastic Single-lane Model for Lane Changing." In: *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on* (cit. on p. 91).

Wu, C., A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen (2017c). "Flow: Architecture and Benchmarking for Reinforcement Learning in Traffic Control." In: *arXiv preprint arXiv:1710.05465*. URL: https://arxiv.org/abs/1710.05465 (cit. on p. 10).

Wu, C., A. Kreidieh, E. Vinitsky, and A. M. Bayen (2017d). "Emergent Behaviors in Mixed-Autonomy Traffic." In: *Proceedings of the 1st Annual Conference on Robot Learning*. Ed. by S. Levine, V. Vanhoucke, and K. Goldberg. Vol. 78. Proceedings of Machine Learning Research. PMLR, pp. 398–407. URL: http://proceedings.mlr.press/v78/wu17a.html (cit. on pp. 10, 82).

Wu, C., K. Parvate, N. Kheterpal, L. Dickstein, A. Mehta, E. Vinitsky, and A. M. Bayen (2017e). "Framework for Control and Deep Reinforcement Learning in Traffic." In:

Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on (cit. on p. 101).

Wu, C., A. Rajeswaran, Y. Duan, V. Kumar, A. M. Bayen, S. Kakade, I. Mordatch, and P. Abbeel (2018). "Variance Reduction for Policy Gradient with Action-Dependent Factorized Baselines." In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=H1tSsb-AW (cit. on p. 10).

Wu, C., K. Shankari, E. Kamar, R. Katz, D. Culler, C. Papadimitriou, E. Horvitz, and A. M. Bayen (2016b). "Optimizing the diamond lane: A more tractable carpool problem and algorithms." In: *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, pp. 1389–1396 (cit. on pp. 11, 169).

Wu, C., J. Thai, S. Yadlowsky, A. Pozdnoukhov, and A. Bayen (2015). "Cellpath: Fusion of cellular and traffic sensor data for route flow estimation via convex optimization." In: *Transportation Research Part C: Emerging Technologies* 59. Special Issue on International Symposium on Transportation and Traffic Theory, pp. 111–128. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2015.05.004. URL: http://www.sciencedirect.com/science/article/pii/S0968090X15001758 (cit. on p. 10).

Wymann, B., E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner (2000). "TORCS, the open racing car simulator." In: *Software available at http://torcs. sourceforge. net* (cit. on p. 83).

Xie, X.-F., S. F. Smith, L. Lu, and G. J. Barlow (2012). "Schedule-driven intersection control." In: *Transportation Research Part C: Emerging Technologies* 24, pp. 168–189 (cit. on p. 82).

Yadlowsky, S., J. Thai, C. Wu, A. Pozdnukhov, and A. Bayen (2014). "Link Density Inference from Cellular Infrastructure." In: *Transportation Research Board (TRB) 94th Annual Meeting* (cit. on p. 116).

Yang, H. and H.-J. Huang (1999). "Carpooling and congestion pricing in a multilane highway with high-occupancy-vehicle lanes." In: *Transportation Research Part A: Policy and Practice* 33.2, pp. 139–155 (cit. on p. 169).

Yen, J. Y. (1971). "Finding the k Shortest Loopless Paths in a Network." In: *Management Science* 17, pp. 712–716 (cit. on p. 138).

Yen, K. S., H.-S. Tan, A. Steinfeld, C. H. Thorne, B. Bougler, E. Cuelho, P. Kretz, D. Empey, R. R. Kappesser, H. A. Ghaida, M. Jenkinson, S. R. Owen, W.-B. Zhang, T. A. Lasky, and B. Ravani (1999). "Advanced snowplow development and demonstration: Phase I: Driver assistance." In: *AHMCT Rept.# UCD-ARR-99-06-30-03* (cit. on p. 26).

Yeo, H., A. Skabardonis, J. Halkias, J. Colyar, and V. Alexiadis (2008). "Oversaturated freeway flow algorithm for use in next generation simulation." In: *Transportation Research Record: Journal of the Transportation Research Board* 2088, pp. 68–79 (cit. on p. 109).

Young, K. D., V. I. Utkin, and U. Ozguner (1999). "A control engineer's guide to sliding mode control." In: *IEEE Transactions on Control Systems Technology* 7.3, pp. 328–342. ISSN: 1063-6536. DOI: 10.1109/87.761053 (cit. on p. 216).

Yuan, Y.-M., R. Jiang, M.-B. Hu, Q.-S. Wu, and R. Wang (2009). "Traffic flow characteristics in a mixed traffic system consisting of ACC vehicles and manual vehicles: A hybrid modelling approach." In: *Physica A: Statistical Mechanics and its Applications* 388.12, pp. 2483–2491 (cit. on p. 54).

Zhang, J., F. Wang, K. Wang, W. Lin, X. Xu, and C. Chen (2011). "Data-Driven Intelligent Transportation Systems: A Survey." In: *IEEE Transactions on Intelligent Transportation Systems* 12.4, pp. 1624–1639. ISSN: 1524-9050. DOI: 10.1109/TITS.2011.2158001 (cit. on p. 117).

Zhang, W.-B., S. Shladover, D. Cooper, J. Chang, M. Miller, C.-Y. Chan, and F. Bu (2007). *Lane Assist Systems for Bus Rapid Transit, Volume II: Needs and Requirements*. Tech. rep. (cit. on p. 26).

Zheng, Z. (2014). "Recent Developments and Research Needs in Modeling Lane Changing." In: *Transportation Research Part B: Methodological* 60, pp. 16–32 (cit. on pp. 37, 84).