MACHINE LEARNING SOLUTIONS FOR TRANSPORTATION NETWORKS

by

Tomáš Šingliar

MS, University of Pittsburgh, 2005

Submitted to the Graduate Faculty of
Arts and Sciences in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2008

UNIVERSITY OF PITTSBURGH

ARTS AND SCIENCES

This dissertation was presented

by

Tomáš Šingliar

It was defended on

Nov 13, 2008

and approved by

Miloš Hauskrecht, Computer Science, University of Pittsburgh

Rebecca Hwa, Computer Science, University of Pittsburgh

Gregory F. Cooper, DBMI and Computer Science, University of Pittsburgh

Geoffrey J. Gordon, Machine Learning Department, Carnegie Mellon University

G. Elisabeta Marai, Computer Science Department, University of Pittsburgh

Dissertation Director: Miloš Hauskrecht, Computer Science, University of Pittsburgh

MACHINE LEARNING SOLUTIONS FOR TRANSPORTATION NETWORKS

Tomáš Šingliar, PhD

University of Pittsburgh, 2008

This thesis brings a collection of novel models and methods that result from a new look at practical problems in transportation through the prism of newly available sensor data. There are four main contributions:

First, we design a generative probabilistic graphical model to describe multivariate continuous densities such as observed traffic patterns. The model implements a multivariate normal distribution with covariance constrained in a natural way, using a number of parameters that is only linear (as opposed to quadratic) in the dimensionality of the data. This means that learning these models requires less data. The primary use for such a model is to support inferences, for instance, of data missing due to sensor malfunctions.

Second, we build a model of traffic flow inspired by macroscopic flow models. Unlike traditional such models, our model deals with uncertainty of measurement and unobservability of certain important quantities and incorporates on-the-fly observations more easily. Because the model does not admit efficient exact inference, we develop a particle filter. The model delivers better medium- and long- term predictions than general-purpose time series models. Moreover, having a predictive distribution of traffic state enables the application of powerful decision-making machinery to the traffic domain.

Third, two new optimization algorithms for the common task of vehicle routing are designed, using the traffic flow model as their probabilistic underpinning. Their benefits include suitability to highly volatile environments and the fact that optimization criteria other than the classical minimal expected time are easily incorporated.

Finally, we present a new method for detecting accidents and other adverse events. Data collected from highways enables us to bring supervised learning approaches to incident detection. We show that a support vector machine learner can outperform manually calibrated solutions. A major hurdle to performance of supervised learners is the quality of data which contains systematic biases varying from site to site. We build a dynamic Bayesian network framework that learns and rectifies these biases, leading to improved supervised detector performance with little need for manually tagged data. The realignment method applies generally to virtually all forms of labeled sequential data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

There are many people without whom this thesis would never see the light of day. The biggest thanks go, in equal share, to my advisor Professor Miloš Hauskrecht and my wife, Zuzana Jurigová. Miloš has been my guide through the world of research, teaching me the art of science. It is to him that I owe my first real touches with the field of artificial intelligence and machine learning, which I suspect will remain my lifelong fascination. It boggles my mind to imagine the efforts he had to expend to provide me with all the funding, time and resources that it took to get it done. He celebrated with me in good times and stood behind me in tough ones. I can only hope he will remain a mentor and a friend for a long time. Thank you!

Zuzana has been by my side through it all and she must never leave. She is wisdom in my foolishness, safe arms in my fear, a gentle push when I cannot go on alone. She is more to me that can be put in words. Just - thank you!

I am very grateful to all mentors who helped me along, inspired, advised and admonished me. My special thanks belongs to the members of my thesis committee who gave generously from their time and expertise to help chisel this capstone: Professors Greg Cooper, Geoff Gordon, Rebecca Hwa and Liz Marai. The AI faculty at the department have always been there to critique ideas, broaden horizons and spark new thoughts. Professors Rebecca Hwa, Diane Litman and Jan Wiebe taught me about processing natural language, maybe the ultimate feat in artificial intelligence. Many of the other faculty at the Department of Computer Science also left a lasting impression in me. When I grow up, I'd like to have Bob Daley's sharp wit with which he approaches both life and research, Kirk Pruhs' depth of knowledge and teach like John Ramirez. Thanks!

Dr. Daniel Mosse has led many projects that sustained my research with funding.

I am thankful for the inspiration that is Dr. Roger Day, someone in whom exceptional qualities as a teacher, researcher and above all, a kind and compassionate human being are combined. Professors Louise Comfort and J S Lin were collaborators who each brought a different perspective into my thinking about infrastructure and complex systems and made me appreciate the vastness of the world behind these academic walls.

It was a great thing for me to meet Dr. Denver Dash, who led me through the aspects of doing great research outside of academia and who is always a wellspring of the craziest innovative ideas. The work with him also allowed me to meet Dr. Eve Schooler and many of the admirable people at Intel.

Thank you all!

Parts of this thesis and closely related work were published at

- ECML-2007 [147], the European Conference on Machine Learning,
- PKDD-07 [146], the Conference on Principles and Practice of Knowledge Discovery in Data,
- ICML-06 Workshop on Machine Learning in Event Detection [145],
- ISAIM-08 [148], the International Symposium on AI and Math and
- UAI-08 [143], the Conference on Uncertainty in Artificial Intelligence.

This thesis is dedicated to my wife Zuzana who has always supported me unflinchingly; and our parents, who gave up our presence for us to pursue goals distant in space and time.

Tomáš Šingliar

1.0 INTRODUCTION

Transportation networks are among the most complex artifacts developed by humans. Accurate models of such systems are critical for understanding their behavior. Traffic systems have been studied with the help of physics-inspired traffic models and traffic simulators. The model parameters would be tweaked until the predictions of the model agreed with empirically collected evidence. However, the evidence was often scarce and hard to collect. As a result, the models were often less than robust.

A fundamental shift has occurred with advances in sensor instrumentation and automatic traffic data collection. Traffic sensors today are capable of measuring a variety of traffic-related quantities, such as speed and vehicle counts. Data collected from the many sensors now placed on highways and roads provides a novel opportunity to better understand the behavior of complex systems, including local and global interactions among traffic flows in different parts of a transportation network.

The main objective of this work is to study methods and tools that make use of the traffic sensor data to build better and more robust traffic models and use them to support a variety of inference, decision making and monitoring tasks. A unifying theme is the application of machine learning and statistical modeling methodology. The work straddles the boundary between theory and application. In the process of attacking transportation management issues, novel models and algorithms are contributed to machine learning: the mixture-of-Gaussian-trees generative model, the sequence label realignment framework with its associated general inference algorithm, and new planning algorithms that can robustly handle highly uncertain environments.

Numerous decision-making protocols in road traffic management are outdated in the sense that they do not realize the full potential of the recently increased availability of traffic flow data. Solutions for common tasks are based on algorithmic problems formulated with simple inputs, commensurate with limited data-collection capabilities in the past. As the explosive developments in information collection capabilities that we experienced in the last few decades transformed the data foundation of decision making, many established algorithmic problem formulations have suddenly become oversimplifications of the actual problem of optimizing with respect to all available information.

Management of traffic with the goal of maximizing throughput is important to areas with increasing demand and congestion. Consequently, much hope rests with Intelligent Transportation Systems (ITS) as a solution that will enable a greater density of transportation. ITS put a variety of new sensing and actuation

equipment at the fingertips of civil engineers and computer scientists: an unprecedented extent of sensor and camera networks, integration with cell phone networks and onboard GPS. Control capabilities such as ramp metering, [1] reversible lane control and variable signal timing have been around for some time, but new efficiencies are now made possible by the data revolution. The need to use the actuation capabilities effectively opens new classes of optimization problems.

The central challenge standing before us is the scale and complexity of behavior of the underlying physical system. Traffic flow is the result of many seemingly-random human decisions combined together through natural and social laws. It is subject to feedback effects such as grid locking and phase transitions such as jam formation. Machine learning algorithms often surprise us with their ability to disentangle, or at least imitate such complex dependencies; and we show here that they can also be fruitfully applied to this domain.

For instance, I develop a stochastic model of traffic flow that uses sensor and infrastructure information to characterize and predict behavior of vehicular traffic in time and space. The main purpose of such models is operate within the context of an Intelligent Transportation System to support decision tasks faced by those who are responsible for the functioning of transportation networks, as well as by those who consume their services. Current traffic flow models operate with deterministic quantities and are calibrated by expert judgment. A statistical modeling approach leveraging sensor data enables this redesign along the lines of stochasticity and automated calibration from data.

Our work on vehicular traffic models generalizes to other complex network systems that underpin our lifestyle: transportation, distribution and communication infrastructure.

1.1 PIECES OF THE PUZZLE

The research is divided into four main chapters: Chapter 3 develops a generative model to describe traffic patterns generated by the highway system. In Chapter 4, I design a macroscopic dynamic probabilistic model for road traffic prediction. Chapter 5 explores optimization problems of vehicle routing using the dynamic model. Chapter 6 describes a sequential data realignment framework to aid in supervised learning of traffic incident detection.

Naturally, the aims of the respective chapters interlock: The path search and optimization algorithms of the Chapter 5 require an estimate of the future costs of traversing road segments. These are provided by predictive inference in the Markov model of traffic flow state built in Chapter 4. While this is the main use of the flow model in the dissertation, it is a general flow model with wider uses, examples of which include anomaly detection and modeling the effects of disturbances such as construction. The models developed in Chapter 3 have the basic goal of supporting other efforts with clean data and a description of traffic behavior. A density-estimation approach is taken at the macroscopic level (entire network), where a model

[1] The Glossary on page 97 defines these and other traffic management terms used throughout the thesis.

of the flow volume distribution is learned and used to fill in missing values. Chapter 6 capitalizes on the gained insights and makes a step towards automatically calibrated incident detection models by casting the problem as supervised learning. The biggest hurdle there appears in a (perhaps) surprising place: the biased and noisy labeling of the incident data. I develop a dynamic Bayesian network formulation to rectify the bias. In yet another example of practical problems inspiring theoretical developments, it turns out that the exponentially faster inference algorithm used for the "cleaning" network generalizes to a class of dynamic Bayesian networks that often appear in diagnostic and detection tasks.

But first things first: let us have a look at the data.

2.0 TRAFFIC FLOW DATA

Summary. *In this chapter we describe the traffic data that we have available and which will serve to train and evaluate our models. It is intended to give a degree of familiarity with the data and basic traffic patterns.*

2.1 TRAFFIC FLOW DATA COLLECTION

The data are collected by a network of sensors, mostly loop detectors[1] operated in the Pittsburgh metropolitan area by a private company under contract from Pennsylvania Department of Transportation (PennDOT). There are about 150 active sensors that record data about the traffic flow at their respective locations. To get a sense of their placement, please refer to Figure 1.

Typically, each mainline lane has a separate loop detector and data is reported per lane. It is rare for off-ramps and onramps to have a detector. Whenever one lane has a sensor, all mainline lanes going in the same direction do. The collection of lane sensors in the same location is referred to as a detector station.

These stations report various quantities regularly to the Traffic Management Center, using a data network built alongside the highways. Three traffic quantities are normally observed and aggregated over a time period: the average *speed*, *volume* (number of passing vehicles) and *occupancy* (the percentage of road length taken up by cars—"traffic density"). The aggregation of our data is at 5-minute intervals; the observed shorter-interval volumes are summed, the occupancies and speeds are averaged.

Two years worth of data are available to us, the entire years 2003 and 2004. On average, about 12% of the data are missing. However, they do not miss uniformly at random as missing data are often a result of a single detector failure that normally takes several days to be repaired. Quality tends to be better in the central region of the city. Data quality information is included in the form of a counter that says how many basic detector cycles yielded a measurement that is valid from the standpoint of the internal quality control algorithm of the sensor. Most of the time, the counter stands at its maximum value.

The data has been imported to a MySQL database and indexed to support the various queries we will need.

[1] See the Glossary (page 97) for definitions of traffic management terms. In the electronic version of this thesis, traffic management terms are made into clickable links to the Glossary.

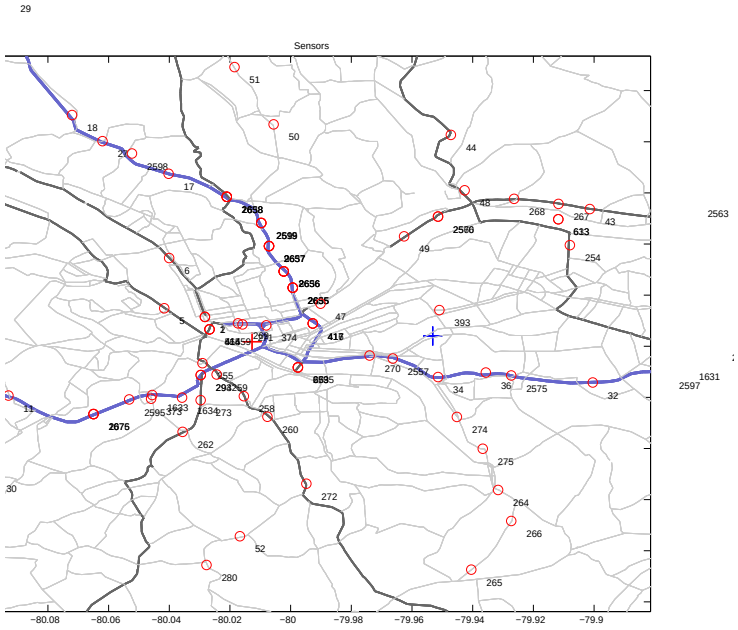


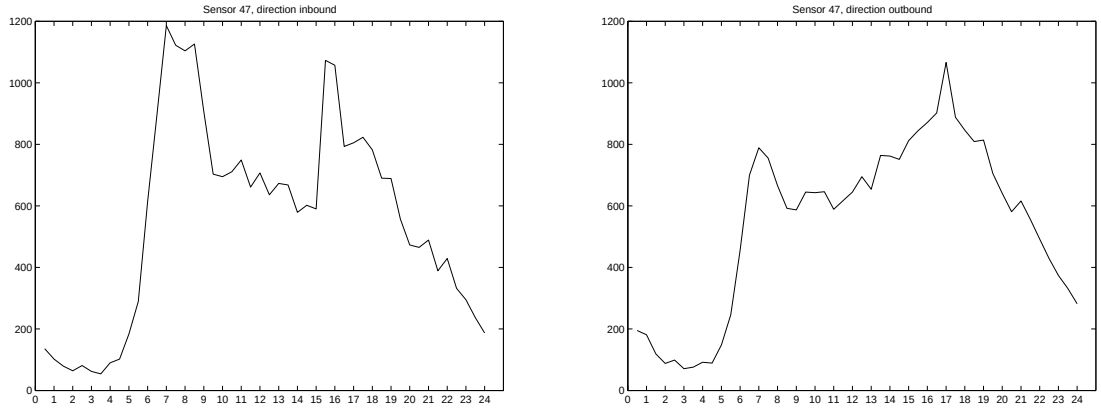
Figure 1: The road network in Pittsburgh with placement of sensors.

2.2 TRAFFIC PATTERNS

Global. The overarching traffic pattern is the diurnal variation and the workday/weekend dichotomy. Workday traffic volumes exhibit a characteristic variation with high volumes during the rush hours and lower volumes at other times. Furthermore, some sensors mainly register traffic in the morning hours (inbound sensors), some have their peaks in afternoon hours (outbound sensors) and many have a mixed traffic pattern. A typical volume profile is shown in Figure 2. Slight but consistent differences are also observed between workdays.

Local. Let us look at the saturation effects that arise when the number of vehicles entering a road segment grows beyond a certain limit. The throughput of a roadway increases linearly with the number of cars entering it until the point of a phase transition when the traffic starts to jam and throughput declines. This is demonstrated by Figure 3a: The correlation of volume and speed is positive at lower volumes, but negative at high volumes that occur during the daily peaks.

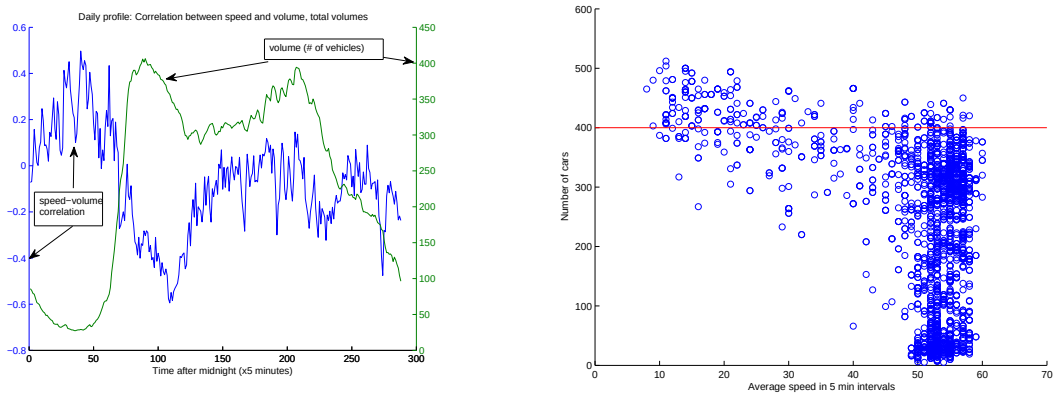
It is a part of traffic manager's expert knowledge that to keep the traffic flowing freely, you need about one lane for each 1200 cars per hour in order to prevent traffic from backing up. Figure 3b, looking at one sensor, confirms that this seems to be the case. The sensor has 4 lanes, so 400 cars in 5 minutes corresponds to 1200 cars per hour per lane. The horizontal (x-) axis represents speed, while the y-axis represents the



(a) Inbound sensor flow profile

(b) Outbound sensor flow profile

Figure 2: Daily traffic profiles. Time is on the horizontal axis, hourly volumes per lane on the vertical one. Both profiles represent sensor 47, a) in the inbound direction, b) in the outbound direction.



(a) Correlation of flow and speed in time

(b) Joint distribution of volume and speed

Figure 3: Local flow properties. **a)** Volume (right y-axis) and its correlation to speed (left y-axis) during the day (x-axis in multiplies of 5-minutes. The graph represents 24 hours.

b) Volume-speed scatterplot diagram, Ft Pitt tunnel sensor inbound. The horizontal (red in color printouts) line represents 1200 vehicles per lane per hour.

number of cars passing. Each point is therefore a paired measurement of car count and speed. Most of the time, the traffic is uncongested and vehicles pass at a mean speed of about 50-55 miles per hour. However, at around 300 vehicles per interval, instances of lowered speed begin to appear and above 400 cars per interval (red horizontal line), the average speed decreases significantly due to congestion.

3.0 A STATIC DESCRIPTION OF TRAFFIC PATTERNS

Summary. *Traffic flow data, as any real-world measurement, suffers from missing or incorrect data and noise. A powerful approach to these problems is to model the joint density of the data and use inference to clean it. Our goal is to devise a model that can capture the dependencies of data dimensions that result from interactions between traffic flows. As with all statistical models, one needs to balance the tension between model complexity and accuracy of representation. In this chapter, a generative model called "mixture of Gaussian trees" (MGT) is developed. It can model densities with fairly complex covariances nearly as well as a full-covariance Gaussian, but does so with only a linear number of parameters. This squares well with the intuition that traffic flow dependencies are local and in nature, being moderated by the physical connectivity of the road network. We use MGT to model the joint distribution of all traffic volumes, but avoid making any assumptions restricting its use to the particular domain.*

3.1 INTRODUCTION

Density estimation is a task central to machine learning. Once a joint probability density of multiple variables is found, it can be used to answer almost any questions about the data. No learning is, however, possible without selecting a bias that embodies our basic assumptions about the data. Ideally, we look for a model that is simple, can explain data well and admits efficient inference and learning algorithms. Some of these goals are naturally opposed, for instance model complexity and accuracy.

Contribution

Because so many quantities describing real-world systems are continuous-valued, continuous multivariate density estimation is an important and well studied topic and many models of varying uses have been developed. In this chapter, a new model is proposed that stands out on low complexity, while it models the types of continuous distributions arising in networks nearly as well as the complex (in terms of model space dimension) multivariate Gaussian.

Traffic quantities measured at different locations on highways are not independent. Spatially close locations experience correlations due to physical interaction between their respective traffic flows. On one hand, this makes the analysis of the network system more challenging. On the other hand, the dependencies can be profitably used to infer information about portions of the roadway where the sensor is malfunctioning or

not present at all. To take advantage of the dependencies, models that compactly capture the covariance of traffic variables are needed. It should also embed a model bias that matches the local nature of dependencies in traffic networks.

We approach the problem by modeling the dependencies with the most complex[1] Bayesian network still tractable, a tree. Since different dependencies may be more prominent in different traffic modes (for instance, times of day), we combine several trees in a mixture model. The mixture variable "selects" which tree dependency structure best represents the particular data point.

3.2 STATE OF THE ART

Two approaches to traffic modeling exist that parallel the division that exists in machine learning: generative joint probability models and discriminative, conditional models. The goal of a joint probability model is to fully describe the distribution of variables in the complete high-dimensional space. Generative models are better equipped to deal with missing values and support general probabilistic queries. Conditional models represent conditional dependencies among (subsets of) variables. They typically do not define a full joint probabilistic model as they lack a prior over data. Their advantage is that they usually directly optimize the desired performance metric.

3.2.1 Joint probability (generative) models

Multivariate Gaussian

The multivariate Gaussian [47] is the most commonly used distribution, as it naturally arises in very many problems. It also appears to be well suited for modeling traffic volumes [12]. The number of cars passing during a given interval can be thought of as a result of a stochastic process in which drivers choose to take the particular segment with a fixed probability p, resulting in a binomial distribution of the number of observed cars. The binomial distribution, for large N, is well approximated by the Gaussian. The multivariate Gaussian model is characterized by its mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. It is described by the probability density function (pdf)

$$p(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left((x - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(x - \boldsymbol{\mu})\right)$$

The covariance matrix $\boldsymbol{\Sigma}$ is assumed to be positive semi-definite. The parameters are usually learned from observed traffic data using maximum-likelihood estimates. The quality of the estimate depends on the number N of data points available. One learning disadvantage of the Gaussian distribution is that the number of its parameters grows quadratically in the number of dimensions, leading to high uncertainty of the covariance estimate. The problem of parameter dimensionality can be addressed by introduction of independence assumptions or transformation of the original covariate space into a lower-dimensional one.

[1] Of course, the scale of model complexity is close to a continuum and what can be considered "tractable" depends on application and size the problems presented.

The axis-aligned covariance Gaussian [47] assumes all variables are independent. Thus, the model explicitly disregards correlations between the component variables by restricting the covariance matrix to the diagonal. An even more severe restriction is a spherical Gaussian with $\mathbf{\Sigma} = \sigma^2\mathbf{I}$. The terms "axis-aligned" and "spherical" refer to the shape and orientation of equiprobability surfaces of the probability density function.

Bayesian belief networks

3.2.1.1 Modeling conditional independence Graphical models [72, 91] reduce the complexity of the model by explicitly representing conditional independences among the variables. In general, the models require smaller numbers of parameters. A Bayesian network consists of a directed *acyclic* graph and a probability specification. In the directed graph, each node corresponds to a random variable, while edges define the decomposition of the represented joint probability distribution. The probability specification is the set of conditional distributions $p(X_i|pa(X_i))$, $i = 1, \ldots, D$:

$$p(\mathbf{X}) = \prod_{i=1}^{D} p(X_i|pa(X_i)),$$

where $pa(X_i)$ are the parents of X_i in the graph. The set $\{X_i\} \cup pa(X_i)$ is referred to as the family of X_i. Intuitively (but not exactly), an edge in a Bayesian network represents a causal influence of the parent on the child.[2]

Bayes nets and traffic

Bayesian belief networks have been used to model both local dependencies between traffic flows at neighboring locations and global congestion patterns. In Sun et al.'s work [159, 158], the dependence structure is constructed manually as a Bayesian network, following the road connectivity. The learning component then reduces to parameter learning.

The JamBayes system [69] is interesting in that it learns directly to predict the occurrence of traffic jams in Seattle area. Thus the prediction task is binary. Moreover, they attempt to predict *surprising* traffic jams – those that are not easily implied by observing the time series. JamBayes uses structure learning to get a Bayesian network suitable for their task. However, their task is very different from what we attempt here. The space they are modeling is essentially the discrete set $\{0, 1\}^d$ and their data is situated within the road network only implicitly. Learning in JamBayes optimizes prediction accuracy and the resulting structure has very little to do with the original network.

Markov random fields

Bayesian belief networks cannot represent cyclic interactions and dependencies well [23]. Bidirectional and circular interactions are better modeled by Markov random fields (MRFs) [23, 86], synonymous with undirected graphical probability models [91]. MRFs model local interactions between variables by means of local potentials φ. The joint distribution is given as a product of F factors:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{i=1}^{F} \varphi_i(\mathbf{x}),$$

[2] The exact concept of (in-)dependence in Bayesian networks is closely tied to the graph theoretical concept of directed separation [91].

where the normalizing factor Z is such that we obtain a proper probability distribution, i.e. $Z = \sum_{\mathbf{x}} \prod_i \varphi_i(\mathbf{x})$. Domains of the factors are cliques of the underlying graph. Ideally, a factor should range over a small subset of X. In the special case of Bayesian networks, the factors correspond to the conditional probability distributions over families.

Note that (1) the marginal of a multivariate Gaussian is a Gaussian and (2) product of two Gaussians is a Gaussian up to a normalization constant. Hence, if the potential functions φ_i are themselves Gaussian, the entire joint distribution implied by an MRF is a Gaussian. Gaussian MRFs do not add representative power over a normal distribution, but may encode a multivariate normal more compactly. They are popular in fields such as image reconstruction [54] because elegant algorithms can be formulated on pixel grids and other regular lattices. However, they are rarely seen applied to large irregular structures where the graph cannot be triangulated [72] neatly in advance.

Trees – efficient inference

Inference in general graphical models is very difficult. Even approximate inference in Bayesian networks is NP-hard [36]. In general Markov random fields, additional difficulty lies in estimating Z, also known as the *partition function*. [3] However, tree-structured graphical models admit efficient (linear) inference, a fact that is directly connected to their low treewidth and the complexity of the join tree algorithm [72]. This favorable property will take a central role in the development of our algorithm.

3.2.1.2 Dimensionality reduction and latent space models Another road to reducing model complexity (number of parameters) is to reduce data dimensionality. Transformations such as PCA (Principal Component Analysis) [47] and its variants can reduce the high-dimensional multivariate space to a more compact approximate representation. With PCA, the data $\mathbf{X}$ is projected onto a lower-dimensional subspace such that maximum variance is retained. The original data can be reconstructed from the lower-dimensional representation Y by means of the basis matrix V:

$$\mathbf{X}_{n\times d} \approx Y_{n\times k} V_{k\times d},$$

where $d \gg k$. Once the data is represented with a lower dimension, any model can be used with less risk of overfitting. In communication networks, PCA has been successfully applied [98] to decompose the network flow data into a smaller number of component "eigenflows". Recently [71] proposed a distributed implementation of PCA that permits an approximate computation in very high dimensional spaces by splitting the matrix into manageable chunks, one per computation node, and bounding the error incurred by deciding to not communicate updates between nodes.

Latent space models

Latent space models [16] achieve dimensionality reduction by assuming that the observed data is a noisy function of a lower-dimensional set of unobserved random variables. They correctly capture the real-world fact that behavior correlations are often caused by some common factors (time of day, weather, large events, etc.). Latent variable models are available and being developed for a variety of data types, including the

[3] Inference in GMRF is simplified by their joint normality.

classical Gaussian [162], multinomial [68, 17] and binary [138, 144]. Non-linear manifolds in the data space can also be modeled [34].

Kalman filter is a dynamic latent space model usually thought of as a temporal tool. We will discuss them in more detail in the following chapter, focusing on state prediction. Kalman filtering can also be applied spatially, to fill in data using the readings from neighboring sensors [56]. However, it is susceptible to breaking at discontinuities in the flow effected by onramps, exits or change in road infrastructure [58, 59]. Kalman filter

3.2.1.3 Mixture models The Gaussian distribution is unimodal. Mixture models [47] overcome the problem of unimodality by assuming that the observations come from several different distributions. There is an unobservable choice variable that selects which component distribution the particular data point comes from. [4] The type of the latent variable distinguishes between a mixture model (discrete) and a latent space model (continuous). In principle, given enough components, a Gaussian mixture model can approximate any distribution with arbitrary precision [110]. The joint probability distribution implied by a mixture model with K components is a sum of the component distributions M_i weighted by the mixture probabilities $p(m = i)$: Mixture models

$$p(\mathbf{x}) = \sum_{i=1}^{K} p(m = i) M_i(\mathbf{x}), \tag{3.1}$$

where m is the mixture variable and M_i is the i-th mixture component, a probability distribution over the space of x.

3.2.1.4 Nonparametric models Non-parametric density estimation models [47, 65] eschew choosing a particular model for the data D. Instead, the data density is estimated by putting a small kernel around each data point $x_i \in D$. If the kernel is itself a probability distribution (integrates to 1), the density implied by the model is

$$p(x|D) = \frac{1}{|D|} \sum_{i=1}^{|D|} K(x, x_i), \tag{3.2}$$

where $K(x, x_i)$ is the kernel function with a maximum at x_i. This functional form implies high values of $p(x|D)$ in regions of high concentration of training data points, which is what we need. In essence, such model can be understood as a mixture model with as many components as there are data points. The largest issue with non-parametric model is the choice of the kernel. A popular choice for the kernel is a radial basis kernel (such as a spherical Gaussian). We will use axis-aligned Gaussian kernels to model local traffic behavior later. An important choice is the bandwidth of the kernels, which controls the smoothness of the resulting distribution. For instance, a very low-variance Gaussian kernel will produce a spiky distribution with many local maxima; increasing the variance will result in a smoother probability surface.

[4] The choice variable often becomes the class variable in classification problems, leading to classification models such as LDA (Linear) and QDA (Quadratic Discriminant Analysis) [47].

3.2.2 Conditional (discriminative) models

Given the strong correlations between flows at points in time and space, it is not surprising to find that traffic data imputation literature is dominated by variants of linear regression [65]. Linear regression is a mainstay of statistical modeling. The variable of interest y is modeled as a linear combination of other covariates $\mathbf{x}$:

$$y = \theta_0 + \boldsymbol{\theta} \cdot \mathbf{x},$$

where $\boldsymbol{\theta}$ is a vector of weights.

Chen [24] uses a spatial linear regression approach to predict volumes. The volume reading of sensor is regressed on the readings of neighboring sensors at the same time. The paper reports a relative error of approximately 11% with this method.

The conditional autoregressive (CAR) model embodies similar intuitions about locality as the MGT model presented below. It was first used to model neighboring pixels in a raster image [15]. The model assumes that the volume $y(s)$ observed at location s is a linear combination of volumes at adjacent locations:

$$y(s) = \epsilon_s + \sum_{r \in N(s)} \theta_r^s y(r), \tag{3.3}$$

where $N(s)$ denotes the neighborhood of s, a parameter θ_r^s is the linear coefficient of r's volume in s's equation, and $\epsilon_s \sim \mathcal{N}(0, \sigma_s^2)$ is Gaussian additive noise.

The intuition is that traffic flows at places not directly adjacent are unlikely to influence each other except via the situation on a road segment(s) connecting them. In other words, a "spatial" Markov property [99] holds. The neighborhood $N(s)$ serves the function of the Markov blanket; that is, given the state in the neighborhood, s is independent of the state at other locations.

[57] use a pairwise regression model with readings from neighboring lanes as input covariates together with upstream and downstream sensor measurements. Pairwise regression is unique in that the output variable is regressed on each input variable separately. The prediction is taken to be the median of the predictions with the isolated covariates.

In [83], the authors consider a hypothetical ad hoc wireless network between vehicles and use an adjusted linear-regression model to fill in the data lost due to connectivity problems inherent in such a mobile setup. Unfortunately, the technique was not evaluated on real data and the evaluation leaves a more quantitative methodology to be desired.

3.3 GAUSSIAN TREE MODELS

The model-complexity problem of the full multivariate Gaussian model is often avoided by assuming that all variables are independent. As a result, the learning problem decomposes to D univariate learning problems,

where D is the dimensionality of the data. The drawback is that ignoring all interactions is unrealistic for traffic networks that exhibit strong correlation between readings of neighboring sensors.

Our model exploits the intuitive observation that traffic variates do not causally (co-)depend if they are associated with distant locations. The dependencies are the result of and transmitted by interactions of physical traffic flows. Since we want to preserve the general applicability of the model, we will not restrict the structural learning to (any derivative of) the network topology.

3.3.1 Mixture of Gaussian Trees

Bayesian networks [72] are a popular formalism for capturing probabilistic dependencies. Being based on directed acyclic graphs, they model causal relations well. However, some application domains exhibit cyclic causal patterns that cannot be naturally expressed as a Bayesian network, with the causes being parent variables of the effects. For instance, traffic congestions are subject to grid locking. Undirected graphical models such as Markov random fields avoid this representational issue, but often at a great computational cost.

One way to address the problem of cyclic interactions is to approximate the underlying distribution with a simpler dependence structure that permits both efficient learning and inference. Having the maximum number of edges without introducing cycles, tree structures are a natural choice. By committing to a single tree, we capture the maximum number of dependencies without introducing a cycle; but inevitably, some dependencies will be ignored. In the *mixture of trees* model [112], the missed dependencies may be accounted for by the remaining mixture components. Mixtures of trees are a special case of Bayesian multinets [53].

Meilă developed the model in a discrete variable setting [112]. The *mixture of Gaussian trees (MGT)* model uses Gaussian instead of discrete variables. The MGT model is related to mixtures of Bayesian networks [161], but differs in how it performs structure search. For trees, the optimal structure can be computed exactly from mutual information, while [161] relies on a heuristic to score the candidate structures and greedily ascend in the model space.

Let there be a set of variables $\mathbf{X}$, such as measurements from different sensors in the traffic networks. Mixture of Gaussian trees models dependencies between the random variables in $\mathbf{X}$ by constructing several tree structures on the same set of variables, each tree encoding a different subset of the dependencies. The tree structures are then combined together in a mixture model.

Definition 1 *The MGT model consists of:*

- *a collection of m trees with identical vertex sets $T_1 = (X, E_1), \ldots, T_m = (X, E_m)$, where each node $x_v \in X$ with parent x_u has a conditional probability function*

$$x_v \sim \mathcal{N}(\mu_v + c_v x_u, \sigma_v).$$

Thus, a child node's value is a noisy linear function of its parent's value. The parameters of the conditional probability distribution associated with the node x_v are (μ_v, c_v, σ_v).

- *mixture weights* $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$ *such that* $\sum_{k=1}^{m} \lambda_i = 1$. *Graphically, this can be represented by an unobservable* m*-valued multinomial mixture variable* z *that selects which tree is "switched on".*

Because of the conditional linear Gaussian local CPDs, a distribution represented by a single tree is jointly Gaussian, but clearly not all Gaussian distributions are representable by trees. Thus, the space of MGT distributions is a proper subset of distributions representable by a mixture of Gaussians with the same number m of components.[5] Note that it is always possible to orient a tree so that each node has at most one parent. Such a tree-structured directed graph encodes the same set of independencies under d-separation as an undirected tree does under graph separation.

Let $T_k(\mathbf{x})$ also denote the probability of $\mathbf{x}$ under the distribution implied by the tree Bayesian network T_k. The joint probability for the mixture model is then:

$$p(x) = \sum_{k=1}^{m} \lambda_k T_k(x). \tag{3.4}$$

(Compare Equation 3.1.)

3.3.2 Inference in the MGT Model

Any probabilistic query in the form $p(\mathbf{y}|\mathbf{e})$ can be answered from the joint distribution (Equation 3.4) by taking:

$$p(\mathbf{y}|\mathbf{e}) = \frac{p(\mathbf{y}, \mathbf{e})}{p(\mathbf{e})} = \frac{\sum_i \lambda_i T_i(\mathbf{y}, \mathbf{e})}{\sum_i \lambda_i T_i(\mathbf{e})} \tag{3.5}$$

Both the numerator and denominator represent m instances of inference in tree Gaussian networks, which is a linear-complexity problem [140] and well supported in the Bayes Net Toolbox [118] for Matlab.

3.3.3 Learning in the MGT Model

The maximum likelihood parameters for the MGT model can be obtained by the EM algorithm.

Let $\gamma_k(n)$ denote the posterior mixture proportion:

$$\gamma_k(n) = \frac{\lambda_k T_k(\mathbf{x}^n)}{\sum_i \lambda_i T_i(\mathbf{x}^n)}.$$

The $\gamma_k(n)$s can be interpreted as "responsibility" of tree k for the n-th data point. Computing $\gamma_k(n)$s constitutes the E-step. The quantity $\Gamma_k = \sum_{n=1}^{N} \gamma_k(n)$ takes on the meaning of the expected count of data points that T_k is responsible for generating. Let us also define the distribution P_k associated with T_k over the set of data points by $P_k(\mathbf{x}^n) = \frac{\gamma_k(n)}{\Gamma_k}$. This distribution represents a weighting of the dataset from which we'll learn the structure and parameterization of T_k.

Three quantities must be estimated in each M-step: (1) the structure of trees that constitute the mixture components, (2) their parameterization and (3) the mixture proportions.

We need to select the structure that maximizes the Gaussian likelihood of the weighted data ascribed to Tree structure

[5] which in turn is, asymptotically as $m \to \infty$, all continuous distributions with support $\mathbb{R}^D$ [67].

the component in the E-step. Chow and Liu [30] have shown how to accomplish this. Given any distribution $\mathcal{D}$, the Chow-Liu (CL) procedure finds an undirected graph structure that optimally approximates the given distribution. In our case, the distribution $\mathcal{D}$ is given empirically as a training data set (histogram) $\mathcal{D} = \{\mathbf{x}^n, w^n\}_{n=1}^N$, where $w^n = P_k(\mathbf{x}^n)$. The algorithm selects a tree model T such that the KL-divergence (or equivalently, the cross-entropy) between the responsibilities computed in the E-step and the tree distribution is minimized:

$$T_k^{new} = \underset{T_k}{\operatorname{argmax}} \sum_{i=1}^{N} P_k(x^i) \log T_k(x^i). \tag{3.6}$$

Equivalently, this maximizes the marginal likelihood of the model. The minimization of the cross-entropy is accomplished by finding a Maximum Weight Spanning Tree (MWST), where the edges are weighted by the mutual information between variables they connect. The structure update for a tree T_k thus requires that we compute the mutual information between all variables $x_u, x_v \in X$. Mutual information between x_u and x_v under distribution p is defined in the discrete case as

$$I_k(x_u, x_v) = \sum_{x_u, x_v} p(x_u, x_v) \log \frac{p(x_u, x_v)}{p(x_u)p(x_v)}. \tag{3.7}$$

That is, the sum is performed over all values of x_u and x_v, yielding $O(r^2)$ complexity for this step if both x_u and x_v can take on r values.

In the continuous case, this is computationally infeasible without making a distributional assumption. Since the modeling assumption is that x_u and x_v are jointly distributed as a mixture-of-Gaussians and one tree is to represent one of the mixture components, we treat $P_k(x^i)$ as a weighted sample from a Gaussian. Estimating the parameters of the Gaussian entails computing the maximum-likelihood estimate of the covariance matrix $\hat{\mathbf{\Sigma}}_k$, from the data D weighted by the posterior contributions $\gamma_k(n)$. Then we can obtain the posterior mutual information for the $k-$th tree in closed form:

$$I_k(x_u, x_v) = -\frac{1}{2} \log(|\hat{\Sigma}_k|/(\sigma_u^2 \sigma_v^2)), \tag{3.8}$$

where $\hat{\Sigma}_k$ is the maximum likelihood estimate of the 2×2 covariance matrix of x_u, x_v. σ_u^2 and σ_v^2 are the diagonal elements of $\hat{\Sigma}^k$ and $|\cdot|$ denotes the determinant of a matrix. Intuitively, if the off-diagonal covariance elements are large compared to the variances themselves, then knowing x_v tells us much about the value of x_u. This is the meaning of mutual information.

When we have the mutual information, we can run the MWST procedure. After we have determined the optimal structure, we orient the tree by picking a vertex at random and directing all the edges away from it. In this manner we achieve that every vertex has at most one parent. Mutual information is symmetrical, which means that any orientation of edges yields a spanning tree that is optimal in the Chow-Liu sense.

Parameter learning

Once the structure is learned, we proceed to parameter learning. It is unsurprising to derive that the M-step update for $\boldsymbol{\lambda}$ is to match the expected empirical marginal: $\lambda_k = \frac{\Gamma_k}{N}$.

Now consider an arc $u \to v$ and recall that $x_v \sim N(\mu_v + c_v x_u, \sigma_v)$. We have data in the form $D_{uv} = \{(x_u^{(i)}, x_v^{(i)}, w^{(i)})\}_{i=1}^N$, where the weight $w^{(i)}$ corresponds to $P_k(x^i)$ computed in the E-step. We can update

the parameters of v, denoted $\theta_v = \{\mu_v, c_v, \sigma_v\}$, by maximizing the likelihood of the data $P(D_{uv}|\theta_v)$. This is accomplished in the standard manner of equating the derivatives of the completed log-likelihood w.r.t. c_v, μ_v and σ^2 to 0. We obtain that the update of μ_v and c_v is the solution of the following linear system:

$$\begin{pmatrix} \sum_{n=1}^{N} w^{(i)} & \sum_{i=1}^{N} x_u^{(i)} w^{(i)} \\ \sum_{i=1}^{N} x_u^{(i)} w^{(i)} & \sum_{i=1}^{N} x_u^{(i)} x_u^{(i)} w^{(i)} \end{pmatrix} \begin{pmatrix} \hat{\mu}_v \\ \hat{c}_v \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{N} x_v^{(i)} w^{(i)} \\ \sum_{i=1}^{N} x_v^{(i)} x_u^{(i)} w^{(i)} \end{pmatrix}. \tag{3.9}$$

Knowing μ_v and c_v we can estimate σ_v^2:

$$\sigma_v^2 = \left(\sum_{n=1}^{N} w^{(i)} \right)^{-1} \sum_{i=1}^{N} (x_v^{(i)} - \mu_v - c_v x_u^{(i)})^2 w_v^{(i)} \tag{3.10}$$

This completes the M-step description. E- and M- steps are alternated until the expected complete log-likelihood stabilizes.

Model selection

The parameter that remains to be chosen is the number of mixture components (trees). We propose that the search be performed by learning the model with increasing number of components until the Bayesian Information Criterion (BIC) no longer decreases. The BIC is defined as an approximation to the integrated likelihood [139, 29]:

$$BIC(k) = -2 \ln p(D|k, \hat{\theta}_k) + \psi_k \ln N \tag{3.11}$$

where D is the training data, $\hat{\theta}_k$ is the ML estimate of parameters and ψ_k is the number of free parameters in model with k components. BIC trades off model fit for model complexity. Note that the first term grows approximately linearly with the number of training data points N, while the second grows logarithmically. Thus on small datasets BIC will favor simple models, while large training sets will support more complex models.

3.4 EXPERIMENTAL EVALUATION

While the proposed MGT model is generally applicable to any continuous multivariate distribution, its development was inspired by traffic networks and we therefore test it on a dataset of traffic variables. The testing task is to impute, or "fill in", missing data. We use three metrics to compare the performance of the MGT model to full- and diagonal-covariance Gaussians and the conditional auto-regressive model.

3.4.1 Data

The data was collected by 75 sensors monitoring the freeways in the densely connected and instrumented central area of Pittsburgh. Each data point is thus a vector consisting of the numbers of vehicles passing the respective sensors during a five-minute interval. The dataset contains all measurements during the time interval between 8am and 9am of all workdays throughout one year. The correlations between sensors are

high and we expect that this will be a challenging dataset for the structure search algorithms, causing them to capture spurious correlations.

Data from some sensors are missing, typically in either short periods (malfunctions that can be resolved by resetting the sensor) or longer streaks (malfunctions that require servicing the sensor). Sensor outages at distinct sensor stations appear independent, as intuitively expected. Data points with missing data were discarded so that we have a complete dataset with observed ground truth for the evaluation. The data is divided into training and testing sets in 70/30 random splits.

3.4.2 Evaluation metrics

In order to assess the quality of distribution modeling, we use the following metrics: a log-likelihood score, relative error and coefficient of determination. The complexity of the model is accounted for by also reporting the BIC score obtained in training of the model. After the model is trained on the training set, some variables in each data point of the testing set are chosen to be hidden; they will be denoted by $h^{(n)}$, while the remaining (observed) variables will be denoted by $e^{(n)}$. Denote the set of hidden variables by H.

We compute the log-likelihood score

$$LLS(H|\theta_M) = \sum_{n=1}^{N} \log p(h^{(n)}|e^{(n)}, \theta_M) \tag{3.12}$$

This score reflects how well the model predicts the set of chosen values, given the remaining observations. As this measure is computed on a subset of the unseen test set and the sample space of observables in all models is the same, it is not skewed by the different complexity of the models.

The coefficient of determination is a classical measure of predictor quality and can be interpreted as the proportion of the data variance explained by the model. It is obtained as $1 - RSS/TSS$, where RSS and TSS are respectively the residual and the total sum of squares, defined as follows. The prediction given by model M is the mean of $p(h^{(n)}|e^{(n)}, M, \theta_M)$. Denoting the actually observed value of the hidden variable h by $x(h)$ and the model's prediction by $y(h)$,

$$RSS = \sum_{h \in H} (x(h) - y(h))^2.$$

The total sum of squares is defined as

$$TSS = \sum_{h \in H} (x(h) - E(x(h)))^2,$$

that is the error of the simplest predictor possible — the constant function. The relative error is defined naturally as $e_{rel} = |x(h) - y(h)|/x(h)$.

Multivariate metrics such as the LLS, which considers all hidden values in a particular data point jointly, reflect the model prediction quality better and should be given more weight than the univariate scores such as the coefficient of determination, which look at each missing value in isolation.

3.4.3 Evaluation setup and parameters

In the reported experiment, 20% of the values are omitted at random from the testing set; the values omitted are fixed across the methods so that each method encounters the same set of missing data. This ensures comparability of the quality metrics across the methods. Statistics were collected from 20 train/test splits.

The product of univariate Gaussians is learned using the ML estimate of mean and variance for each dimension separately. Conditioning is trivial for the model: $p(h^{(n)}|e^{(n)}, \mu, \sigma) = p(h^{(n)}|\mu, \sigma)$. Gaussians

The full covariance Gaussians are parameterized from the data by the maximum likelihood estimates. Conditionals are obtained as $p(h^{(n)}|e^{(n)} = f, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}(\overline{\boldsymbol{\mu}}, \overline{\boldsymbol{\Sigma}})$, where

$$\overline{\boldsymbol{\mu}} = \boldsymbol{\mu}_h - \boldsymbol{\Sigma}_{he}\boldsymbol{\Sigma}_{ee}^{-1}(\boldsymbol{\mu}_e - f) \tag{3.13}$$

$$\overline{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_{hh} - \boldsymbol{\Sigma}_{he}\boldsymbol{\Sigma}_{ee}^{-1}\boldsymbol{\Sigma}_{eh} \tag{3.14}$$

and $\boldsymbol{\Sigma}_{..}$ are the respective block submatrices of $\boldsymbol{\Sigma}$. For instance, $\boldsymbol{\Sigma}_{uv}$ is the matrix obtained from $\boldsymbol{\Sigma}$ by extracting all the rows in the set u and all columns in v.

Since the conditional auto-regressive model (CAR) model may not define a proper distribution[6], we obtain a large number (10^6) of samples with the Gibbs sampler. We allow 10 Gibbs updates between taking each sample to mix the chain and decrease correlations between the samples. CAR model

CAR linear weights were estimated using a the ridge-regression procedure [65] with the ridge penalty set to $\lambda = 30000$.

$$\underset{\boldsymbol{\theta}^s}{\text{minimize}} \sum_{n=1}^{|D|} \left(y^{(n)}(s) - \sum_{r \in N(s)} \theta_r^s y^{(n)}(r) \right)^2 + \lambda ||\boldsymbol{\theta}^s||, \tag{3.15}$$

where $||\cdot||$ is the L_2 (Euclidean) vector norm. This is an instance of a typical learning problem of the form "minimize $Err[D|\theta] + \lambda J[\theta]$". The first component favors a good fit to the data, while the second pushes for some desirable characteristic of the parameters θ, in this case their small magnitude under Euclidean norm. λ is a parameter that balances between these competing goals. The ridge penalty λ was calibrated in a separate cross-validation experiment. The noise variance σ_s^2 of the CAR model is equal to the mean squared residual.

The remaining question is how to define the neighborhood $N(s)$ for the CAR model. We use as the neighborhood all sensor locations within 3 miles of s. With sensors spaced approximately 1 mile apart, this gives us fairly small neighborhoods with ≤ 10 sensors (except for the central densely instrumented area). It is better to err on the side of including some potentially irrelevant sensor locations as the regularized learning method should winnow down the uninformative features.

[6] Alternatively, we could define a joint distribution as a product of the Gaussians implied by the local regressions:

$$p(\mathbf{X}) = \prod_i p(X_i|N(X_i), \theta^{x_i})$$

However, such distribution would not necessarily represent the empirical joint distribution of $\mathbf{X}$ well due to circular dependencies. To use a MRF analogy, it is more akin to pseudolikelihood than the true likelihood.

We learn the Mixture of Gaussian Trees (MGT) model using the EM algorithm described in Section 3.3.1, using 1,2,3 and 5 mixture components. The LL score is computed by conditional inference as described in Section 3.3.1 as well.

Mixture of trees

3.4.4 Evaluation results

Evaluation results are shown in Table 1. The 3-component MGT yielded the best log-likelihood score, closely followed by the conditional autoregressive model. However, the differences in the likelihood score are not statistically significant among the models, with the exception of the diagonal Gaussian. The MT model achieves this performance with many fewer parameters. The difference is reflected in the BIC complexity penalty. The CAR model is the worst on this metric. The BIC suggests that a single-tree model might be appropriate, although the likelihood is higher for mixtures of 2 and 3 trees. Further in favor of the mixture model, the MGT model also has an embedded structure-learning component, while the CAR model operates with informed pre-defined neighborhoods. Therefore MGT achieves this performance with less prior information. MGT is very good at modeling the training data, yielding low BIC scores. This can lead to some amount of overfitting: the 5-component MGT shows signs of testing performance deterioration. Overall, the MGT is the best generative model on distribution modeling (the log-likelihood score).

Data likelihood

The relative error results confirm our original intuition that the MGT stands between the full and independent Gaussian in performance and complexity. The CAR model is widely used for a reason: it produces the best mean prediction. However, this comes at a high model and inference complexity price when more than one value is missing. Furthermore, it is not clear how to learn such a discriminative model from incomplete data. The story told by the coefficient of determination agrees with the relative error.

Relative error

We note the disproportionately high BIC scores of the full-Gaussian, independent Gaussian and CAR models. In the independent Gaussian case, this is caused by poor modeling of the dependencies in data. On the other hand, the full-Gaussian and CAR models suffer a high complexity penalty. This cautions us that normally we do not have the data to fit a full covariance Gaussian. A variety of factors is observable in traffic networks, of which the most obvious is the variability with the time of day. The correct way to deal with observable factors is to condition on them, which cuts down the training data severely. The full covariance Gaussian will likely meet scaling-up problems in such a context.

In terms of time complexity, the plain Gaussian representations are clearly the fastest. However, asymptotically, the mixture of trees is faster. The slower results here just reflect the fact that much heavier machinery (a general Bayesian network library) is invoked for the mixture of trees algorithm. The CAR model is easy to learn (it is just regularized regression), but the inference is very slow.

Method	# prms	LLS	BIC	Relat err	Coef of det	Time (learn)	Time (infer)
$N(\mu, \mathbf{\Sigma})$	2,925	$-8,448(664.2)$	186,910 (6,751)	0.039(0.0022)	0.889(0.012)	0.012 (3e-4)	0.36 (0.06)
$N(\mu, \text{diag}(\boldsymbol{\sigma}))$	150	$-13,314(1,036.3)$	197,910 (7,361)	0.073(0.0050)	0.638(0.019)	0.001 (0.001)	0.01 (0.001)[a]
CAR	1,277	$-8,162(598.3)$	203,126 (5,970)	0.037(0.0023)	0.868(0.016)	0.421 (0.07)	5671 (493)
SingleTree	224	$-8,282(616.6)$	13,667 (784.1)	0.057(0.0056)	0.765(0.050)	229 (42)	1.76 (0.17)
MixTrees(2)	449	$-8,176(638.6)$	17,159 (4,299)	0.053(0.0050)	0.766(0.052)	720 (97)	5.48 (0.66)
MixTrees(3)	674	$-8,158(632.0)$	24,562 (12,995)	0.055(0.0141)	0.704(0.176)	814 (98)	5.65 (1.47)
MixTrees(5)	1,124	$-8,226(624.2)$	67,787 (32,787)	0.197(0.4567)	0.305(0.341)	2305 (566)	15.04 (2.24)

Table 1: The likelihood scores (larger is better), and BIC scores (smaller is better), relative errors (smaller is better) and coefficients of determination (larger is better). The parenthesized numbers are the standard deviations across test splits. Statistical significance of the differences is visualized in Figure 4. The time is measured for the entire dataset of approximately 400 data points with 75 dimensions, for both learning (obviously) and inference (less so). The time unit is seconds.

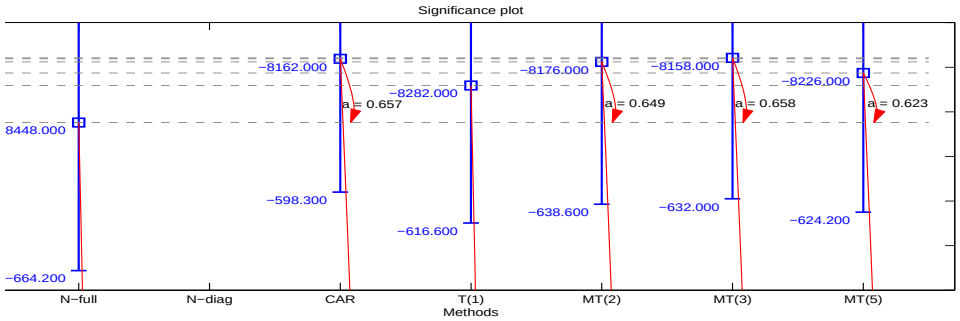
[a] looking up missing values takes longer than learning

3.5 CONCLUSIONS

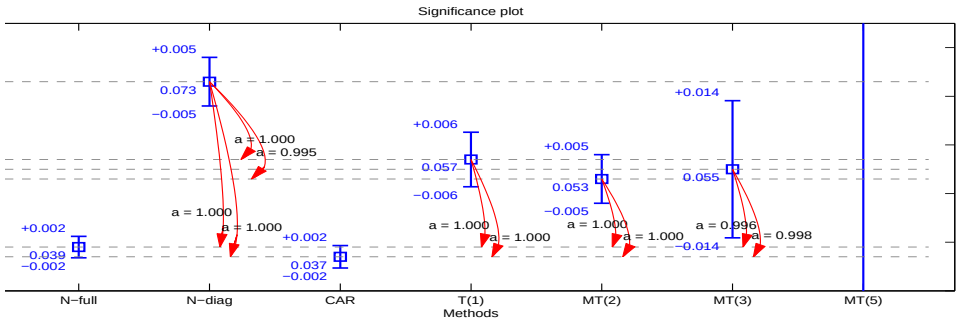
We developed and presented a new mixture of Gaussian trees model that provides a middle ground in between the data-hungry full covariance and the unrealistic all-independent Gaussian models. The model complexity (number of parameters) can be controlled by choosing the number of mixture components, but is always linear in the number of dimensions. We have explored several other methods for modeling traffic volume and used several measures to compare their performance: predictive likelihood, relative error and coefficient of determination. While the proposed model was not the best performer on all metrics, it did unequivocally outperform the other model with a linear number of parameters and it never fell far behind the more complex models.

If data are plentiful, the full-covariance Gaussian can be estimated with a high accuracy. However, in the more realistic case when data is scarce, our mixture-of-trees model, with a number of parameters that scales more favorably with the dimension of the dataset, offers itself as the method of choice when multiple neighboring measurements may be missing.

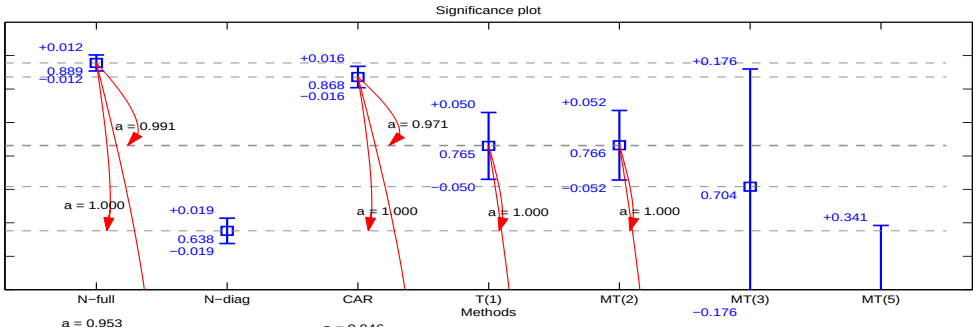
Many interesting research issues remain open for the mixture-of-trees model. For example, when learning from small sample-size datasets, it would be advantageous to generalize the Bayesian version of the MT learning algorithm [111] to handle the distributions in the exponential family (including Gaussians that we used here). Automatic model complexity selection and greater resistance to overfit are among the expected benefits of applying the Bayesian framework.



(a) Log-likelihood score



(b) Relative error



(c) Coefficient of determination

Figure 4: Statistical significance plots for the four metrics shown in Table 1. On the x-axis we have the methods we are comparing, on the y-axis the respective evaluation metrics. The (blue) squares show the mean values of the metric and the error bars around them are one standard deviation. Both the squares and the error bars are annotated to the left with their actual value. The (red) arrows represent a comparison between the method which sits at the origin of the arrow to the method sitting on the dashed (gray) line which the arrow points to. The arrows are annotated with values of the Student t CDF, thus values near 1.0 are highly significant. The arrows that can be easily inferred were deleted for sake of presentation clarity. In figures a) and c), some of the arrows intentionally point below the cropped area of the plot to an outlier method, they are all annotated with significance 1.0.

4.0 PROBABILISTIC TRAFFIC PREDICTION WITH A DYNAMIC MODEL

Summary. *The previous chapter looked at static, atemporal snapshots of traffic state. In contrast, we are now interested in its dynamic behavior and build a probabilistic model that predicts future traffic state from the current state. The principle of traffic prediction is simple: Flow volume on a target segment can be predicted by the current flows on surrounding segments and their speeds. In a perfect world where all traffic parameters are known, traffic flows can be inferred directly from the law of flow conservation. However, many unobservable quantities and uncertain outcomes complicate the problem. Real traffic flows and speeds fluctuate randomly and are not always accurately measured. We develop and test a new probabilistic traffic prediction model that combines macroscopic flow models based on flow conservation laws with stochastic volume-speed functions that are extracted from traffic data, marrying elements of macroscopic flow models with probabilistic reasoning. The model is tested on the task of predicting flow volume and generates a superior predictive distribution as measured by future data likelihood.*

4.1 INTRODUCTION

The focus of this chapter is the development of a probabilistic model capable of predicting the future state of vehicular traffic flow in a highway network. We seek a traffic prediction method that can provide a probability distribution over future traffic flow states at arbitrary times.

Our model is loosely inspired by macroscopic traffic models [94] known from civil engineering. Macroscopic flow models divide the traffic network of interest into segments and define a set of difference equations describing how the "fluid" formed by the vehicles moves in the network. They are not very suitable for "online" operation because it is difficult to adjust their state by incorporating incoming information. This is mostly because they operate with deterministic quantities, such as volume-speed relationships, ignoring the stochastic nature of the data for sake of simplicity. However, for quantitative decision problems, such as lane control, signal timing and route planning, it would be rather useful to have a distribution.

Most previous efforts to give a predictive distribution over traffic flows have relied on general-purpose time-series models such as the autoregressive moving average models (ARMA) [2, 38, 27, 165]. Such statistical models make the assumption of stationarity of the underlying process. This assumption is often violated as

observable traffic conditions can evolve differently at different times. Moreover, while time-series analysis models are probabilistic, they are ignorant of the underlying process that generates the data. We seek to improve their prediction by building a model infused with knowledge about the macroscopic dynamics of the underlying physical system. The result is a Markov chain model whose hidden state transition dynamics mimic the flow equations.

A critical element in a flow model is how flow speed is determined when unobserved. Speed is in a stochastic relationship with the degree of capacity utilization. Traditional flow models [164] work with a single functional description of the relationship, fixed throughout the road network. More recent work [25] suggests that a more flexible, probabilistic description is more appropriate. Our model describes the relationship with a non-parametric kernel estimate of the joint volume-speed distribution learned from past observed data. The volume-speed relationships in our model are thus made probabilistic and location-specific.

The main inference task we want to support with our model is volume and travel time prediction. Unfortunately, the stochastic quantities and nonlinearities in the dynamic model do not lend themselves to a compact representation. This complicates the design of an efficient inference algorithm. Markov Chain Monte Carlo (MCMC) methods [3] are a very general approach to approximate inference that only require that we be able to sample from the distributions in question. We resort to MCMC methods and implement a particle filter [153] for approximating the predictive distributions of traffic quantities.

We evaluate the quality of the model and the inference procedure on real-world data obtained from traffic sensors installed on an urban freeway. We compare our method to distributions obtained by the ARMA models. We find that our model exceeds the ARMA performance in terms of predictive likelihood. The prediction improves the most for downstream segments, which we attribute to the fact that those are the segments about which the model has the most information from having observed the vehicle stream for a longer time.

4.2 STATE OF THE ART

Three basic approaches to predicting traffic have prevailed in the literature and in practice. We can attempt to simulate the behavior of individual cars; the aggregate behavior of some co-occurring set of vehicles; or approach the problem as a time series without further studying the minutiae of the process generating the data. Recently, hybrid approaches such as this work and [116] have appeared.

4.2.1 Microscopic models

It is the industry standard to use micro-level simulation for offline analysis of traffic when real-time responsiveness is not required. Microscopic simulation involves simulating a population of agents corresponding to vehicles in actual 3D space, subject to the full range of physical laws. Each agent possesses a decision routine

designed to approximate the reaction of a human driver to surrounding conditions. The larger-scale traffic flow properties emerge from the collective behavior of the simulated agents. There are numerous parameters in microscopic models and it is usually unclear how these parameters should be set for the model to match observed data.

Many simulators have been developed (PARAMICS [151], AIMSUN [150], TRANSIMS [149],...). The leading commercial products have impressive modeling and graphics capabilities and can run in a parallelized manner, but cannot yet obtain probabilistic estimates of many interesting quantities in real time because of the complexities of microscopic models. Probabilities are obtained by repeated runs of the same model.

4.2.2 Macroscopic models

Macroscopic models take a higher-level, more aggregated view. The network of interest is divided into segments and the reasoning works with summary quantities such as vehicle numbers, average speeds and spatial vehicle densities. Macro-models usually take the form of a set of difference equations that involve no stochastic quantities. Let $\mathbf{S}_i^t$ represents the state vector of traffic quantities (volumes, speeds, densities) at a time t and location i. Then a macroscopic model is essentially a vector of equations:

$$\mathbf{S}_i^{t+1} = F(\mathbf{S}_i^t, \mathbf{S}_{i-1}^t, \mathbf{S}_{i+1}^t),$$

where F is usually a complex, non-linear, parameterized function of previous state at the i-th segment $\mathbf{S}_i^t$ and the state at neighboring segments $\mathbf{S}_{i-1}^t, \mathbf{S}_{i+1}^t$. The neighboring segments are dictated by network connectivity. The prediction of traffic behavior is carried out by extrapolating the solution of the differential equations with respect to time. An example of a macroscopic flow model very popular in the traffic literature is the METANET model [94]. The equations defining the model are summarized in Figure 5.

A comprehensive macroscopic traffic flow model always relies on a number of external parameters. For our model, the most critical external inputs are the way of characterizing demand and the fundamental diagram, which we now describe.

OD matrix

4.2.2.1 Demand characterization First, macroscopic flow models require a description of inflow to and outflow from the measured and modeled part of the road network. Typically, these are given in terms of the OD (origin-destination) matrix [103, 108, 75, 87] . The model represents the demand in the network, specifying for any (i, j)-pair of system entry/exit points (in our case, onramps/offramps) how many vehicles wish to travel from i to j. The drivers' decisions are of course unobservable and must be estimated in aggregate. For our purposes, a simpler characterization suffices: a net inflow-outflow (IO) model provides the distribution of net demand injection into the network for each segment, conditioned on the time of day. The O-D matrix can be also used to define turning proportions for interchanges. Section 4.3.4 discusses our derivation of turning proportion matrices.

The highway link is divided into N segments of length L_i. The outflow of segment i at time interval k of is $q_i(k) = \rho_i(k)v_i(k)\lambda_i$, where $\rho_i(k)$ is the vehicle density at time k and λ_i is the number of lanes. The flow conservation law gives

$$\rho_i(k+1) = \rho_i(k) + \frac{T_f}{L_i\lambda_i}(q_i(k) - q_{i+1}(k)),$$

where T_f is the simulation time interval.

The speed update consists of three terms, one for speed change due to in/outflow of vehicles, one for speed-density dependency and one for compression/decompression when drivers observe higher/lower density downstream:

$$v(k+1) = v_i(k) + \frac{T_f}{\tau}\left[V(\rho_i(k)) - v_i(k)\right] \tag{4.1}$$

$$+ \frac{T_f v_i(k)}{L_i}(v_i(k) - v_{i+1}(k)) \tag{4.2}$$

$$+ \frac{\nu T_f}{\tau L_i}\frac{\rho_{i+1}(k) - \rho_i(k)}{\rho_i(k) + \kappa} \tag{4.3}$$

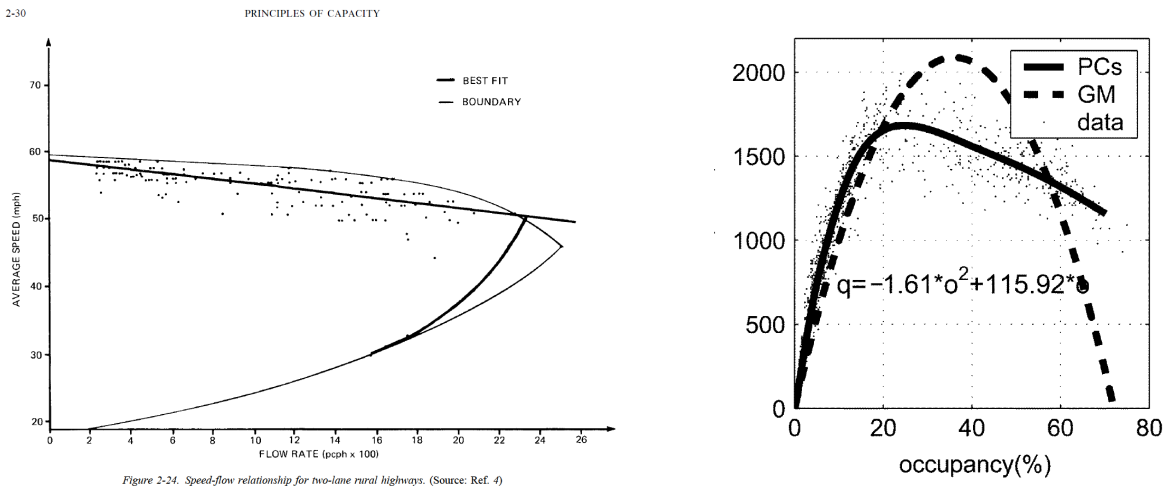
where τ, ν, κ are model parameters. The critical term is in the first equation, where speed is determined as a monotonically decreasing function of density ρ:

$$V(\rho_i) = v_{free}\exp\left[-\frac{1}{a}\left(\frac{\rho_i}{\rho_{crit}}\right)^a\right]$$

with a a model parameter, v_{free} the free-flow speed and ρ_{crit} the critical (jamming) density.

MetaNet also includes an origin (onramp) queue model which is not shown here, but does not address intersections and interchanges on its own.

Figure 5: MetaNet **model overview.** Note the deterministic nature of the relations and the difficult-to-interpret parameters τ, ν and κ that do not correspond to any intuitive physical quantity and a parameter a that calibrates how close the vehicle density must get to the critical density before slowdown occurs. The difficulty of calibrating these is among the main drawbacks of macroscopic models.



(a) HCM fundamental diagram

(b) Principal curve fundamental diagram

Figure 6: Fundamental diagram representations. **a)** A single-curve representation of a fundamental diagram. Figure reproduced from the Highway Capacity Manual [163]. **b)** FD representation with principal curves fits better than a quadratic function as argued by [25]. The HCM version is a volume-speed relationship, while Chen et al work with occupancy-volume.

4.2.2.2 Fundamental diagrams One of the critical parameters of a macroscopic model is its volume-speed relation, how the equilibrium speed of traffic flow changes in response to its density. This relation is described in terms of the so called fundamental diagram [137].

Traditionally, traffic engineers have worked with a deterministic volume-density diagram, neglecting the stochastic variations in the two quantities [109]. In the Highway Capacity Manual, the authoritative compendium of traffic engineering knowledge, the diagram has no particular functional form and is represented graphically (Figure 6a)).

In the civil engineering literature, the volume-speed relation is typically represented by a quadratic curve [55]. However, recently some researchers have contended [25] that a quadratic curve is not the best fit to the data and there is no physical reason why this should be a good model. Indeed, we observe in our own data a wide variety of curves that do not conform to a parabolic shape. Chen at al used principal curves [65], a non-parametric approach, and showed them to be a much better fit than quadratic curves (Figure 6b)).

The fundamental diagrams are important for calibration of microscopic simulators. In particular, a good match in between data generated by the simulator and the actually observed data [81, 82]. is considered evidence that the microscopic model is well calibrated [137].

4.2.3 Learning approaches

The third alternative are models that do not attempt to represent in detail the physical processes that give rise to observed traffic patterns. Instead, they apply and adapt general-purpose statistical and machine learning methods to model the traffic behavior. Interestingly, these models may work even if they do not reflect the underlying structure of the traffic network. For example, a recent, quite successful approach to congestion state forecasting [69] entirely dispenses with modeling traffic, using Bayesian network structure learning to come up with a predictive model. The resulting models retain no hint of the topographic connectivity structure.

The learning approach is well-founded and attractive, but we recognize the need to model the physical processes where general-purpose learning models such as ARMA fail to capture the behavior satisfactorily. The interest in the field has also been moving in that direction, as documented by [165], a comprehensive review that surveys the state of the art of not only the algorithmics of short-term traffic prediction, but of the entire design and development process as well.

4.2.3.1 Time-series models Time-series modeling approaches including autoregressive moving average models, hidden Markov models and predictive state representations are a natural choice for sequential traffic data.

Auto-regressive moving average

Variants of ARIMA models are the earliest and still the most commonly taken approach to traffic prediction [2, 38, 27]. Autoregressive (AR) moving average (MA) models model each point in a time series as a linear combination of previous values:

$$y^t = \epsilon^t + \sum_{j=1}^{p} \varphi_j y^{t-j} + \sum_{j=1}^{q} \theta_j \epsilon^{t-j} \tag{4.4}$$

where $\epsilon^t \sim N(0, \sigma^2)$ is i.i.d. Gaussian noise. The first sum expresses the Markovian dependence on previously observed values. The second sum carries the effect of noise into subsequent time points. The MA component retains the effect of perturbations for a few time steps. θ_j and φ_j are the parameters to be fitted in learning.

Integrated (arIma) models capture the sequence of differences instead of the original values. Autoregressive sequence models are now rarely used in their plain flavor, due to their recognized drawback of accurately predicting on average, but under-predicting the more extreme events [38, 61]. This is a consequence of the fact that the Gaussian assumptions they are based on are ill-matched to the bimodal (free flow/congestion) behavior characteristic of freeway traffic.

Predictive state representation

Predictive linear Gaussian (PLG) reverses ARMA's view [135, 168]. Instead of assuming the state is defined by p recent observations, the state of a PLG model is given by a jointly Gaussian expectation over next p data points. This might be advantageous if there is other information to be folded into the prediction, such a profile of daily variation. Among disadvantages of PLGs is the learning algorithm — the parameters are set by a moment matching procedure that seems to require extensive learning data. The kernelized version mitigates the problem, at a rather formidable computational cost.

Subspace identification methods [141] are a good alternative that can learn both Kalman Filters and predictive linear Gaussians.

Spatial hidden Markov

In [12], a hidden Markov model is introduced whose slices represent *spatial* locations. The observations in the HMM are entire temporal sequences, assumed to be generated as a sum of radial basis functions. The hidden state space is the space of the RBF parameters. This model elegantly solves the problem of non-uniform spacing of detector stations by assuming a finer coverage and treating not-covered locations as missing data. It seems a more successful attack than [97], who used a set of coupled hidden Markov models to capture spatiotemporal dynamics without much success. The probable cause of the failure is that information propagates at varying speeds in the underlying physical system, while the model assumes constant slice spacing. A possible remedy for the shortcoming is to formulate the task in terms of a semi-Markov decision process [70].

Coupled hidden Markov models

4.2.3.2 Kalman filters Kalman filters are a mainstay of state estimation and tracking. One of the seminal if straightforward applications of Kalman filtering [77] to traffic volume prediction is [122]. A Kalman filter is the continuous state space analog of a hidden Markov model. The state of the Markov chain is represented by a k-dimensional vector $\mathbf{x}$ of continuous values. The evolution of the state happens in discrete time according to the equation

$$\mathbf{x}^t = \mathbf{F}^t \mathbf{x}^{t-1} + \mathbf{w}^t, \tag{4.5}$$

where $\mathbf{F}^t$ is the state transition model in the form of a $k \times k$ matrix. $\mathbf{w}^t$ is Gaussian noise with mean $\mathbf{0}$ and covariance $\mathbf{Q}^t$. At time t, a noisy observation $\mathbf{o}^t$ of the true state is drawn from $\mathcal{N}(\mathbf{H}^t \mathbf{x}^t, R^t)$. The transition and observation noises are independent of each other and of their realizations in the previous steps. The superscripts ^{t} can be dropped if, as is usually the case, the parameters are stationary in time.

Kalman filtering and other time-series prediction algorithms are a good method when only a few consecutive samples from each series are missing or invalid [37]. If the error persists, the time-series prediction converges towards the mean or is thrown off completely by the invalid values, depending on the quality of the data screening algorithm applied to the raw source.

We obtain the *extended Kalman filter* (EKF) formulation if the linear state dynamics $\mathbf{x}^t = F\mathbf{x}^{t-1}$ and/or the observation model is replaced by a non-linear function, e.g. $\mathbf{x}^t = f(\mathbf{x}^{t-1})$. The matrix F is then the Jacobian $F = \frac{\partial f}{\partial \mathbf{x}}$. EKF can perform poorly f is a very non-linear function, since only the mean is propagated through the state dynamics equations.

Unscented Kalman filter [74] is better than Extended Kalman filter when the state dynamics are highly non-linear. It selects a number of points (sigma points) so that they represent the variance of the state estimate better. These are propagated through the transition function and the mean and covariance estimates are reconstituted from the result. The same approach is taken to processing the observation. [116] use UKF as the state of the art traffic prediction technique to analyze the performance of their algorithm.

4.2.3.3 Other learning paradigms

Neural networks have been used for numerous traffic prediction purposes [28, 26, 31, 33], sometimes with disputable results [43]. The disadvantage of neural networks is the difficulty of interpreting their learned parameters in the real world. Thus when something goes wrong, it is virtually impossible to track down the problem.

Neural networks

Non-parametric regression methods [32] are an appealing approach when large databases of traffic measurement are available. The design of a non-parametric technique—typically a variant of k-nearest neighbors—revolves around finding a distance measure that accurately reflects similarity between traffic states [152, 123]. [92] find that better prediction performance is achieved when the distance metric is defined on a short history rather than the traffic state alone. This implies a degree of non-Markovianity in traffic state sequence distribution. In large networks, the curse of dimensionality may prevent us from finding a good similarity measure. Non-parametric regression then degrades to essentially random sampling from training data.

Nonparametric regression

4.2.4 Hybrid models

In a rare example of work similar to our approach, Mihaylova et al [116] have implemented a stochastic version of the MetaNet flow model [94]. Due to the presence of intractable distributions, they use a MCMC inference algorithm, as do we in our own model. The difference between this work and theirs is twofold: (i) the form of the flow equations and (ii) the approach to modeling the volume/speed relationship. In addition, their model does not consider intersections. A fuller discussion of the distinctions between the models is deferred until after the presentation of the principles of our model.

4.3 THE DYNAMIC FLOW MODEL

The dynamic model represents the behavior of the traffic system in time. It is fully described by the conditional density $p(\mathbf{s}^{t+\delta t}|\mathbf{s}^t)$, where $\mathbf{s}$ are traffic states and δt is the model time step. This time step δt is usually much shorter than the prediction horizon Δt. The traffic state $\mathbf{s}$ decomposes into demand, volume and speed vectors $\mathbf{q}$, $\mathbf{y}$ and $\mathbf{v}$, respectively: $\mathbf{s} = (\mathbf{q}, \mathbf{y}, \mathbf{v})$. *Demand* is the number of vehicles present on a road segment at a given moment. *Volume* is the number of vehicles passing a point on the roadway during a time interval. The demand is an unobservable quantity, while volumes are directly measured. The components of vectors $\mathbf{q}$, $\mathbf{v}$ and $\mathbf{y}$ correspond to individual road segments, identified by lower index such as i. Upper indices will denote time. Colon notation is used to specify sequences. Thus, $v_i^{1:t}$ is the vector of speeds observed at segment i, at times 1 through t.

To build the model, we assume that the previous state is sufficient to explain correlations among traffic quantities from neighboring segments. Hence the requirement that the model step be relatively short. Then,

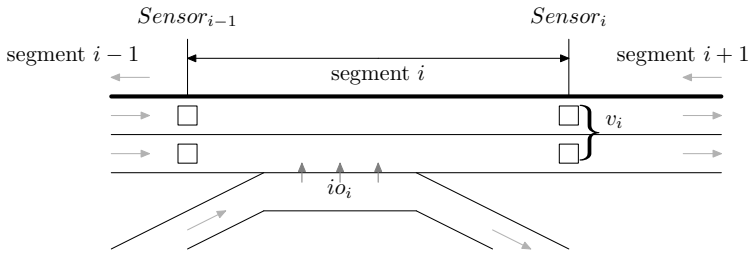


Figure 7: Quantities of interest related to a physical highway. Only one travel direction is shown, according to the assumption that opposing flows do not interact.

we can decompose the joint over the entire network into a product over the segments:

$$p(\mathbf{s}^{t+\delta t}|\mathbf{s}) \quad = \quad \prod_{i=1}^{S} p(q_i^{t+\delta t}, y_i^{t+\delta t}, v_i^{t+\delta t}|\mathbf{s}^t),$$

where S is the number of road segments. To further decompose the model we assume that interactions occur mostly between demands and volumes, while speeds v_i^t are thought of as a "dependent variable". Hence the distribution further decomposes to:

$$p(\mathbf{s}^{t+\delta t}|\mathbf{s}) = \prod_{i=1}^{S} p(q_i^{t+\delta t}|\mathbf{s}^t) p(y_i^{t+\delta t}|\mathbf{s}^t) p(v_i^{t+\delta t}|y_i^{t+\delta t}) \tag{4.6}$$

In the following we give the details and rationale for each of the model components.

4.3.1 Flow conservation dynamics

The hidden state dynamics is derived from the law of flow conservation. The distribution of the demand on a segment i, $p(q_i^{t+\delta t}|\mathbf{s}^t)$ depends on the previous state and the volume shifting to and from neighboring segments:

$$q_i^{t+\delta t} = q_i^t + y_{i0}^t + y_{i-1}^t - y_i^t, \tag{4.7}$$

where the term y_{i0}^t captures the expected inflow/outflow through the unmeasured on- and off-ramps on the i-th segment. These unobserved volumes are the major source of uncertainty that enters the flow equations. y_{i-1} refers to the outflow of the upstream segment which by definition equals the inflow into the i-th segment. The distribution y_{i0} corresponds to the difference of upstream and downstream flows and can simply be represented non-parametrically by storing the observed differences $y_i - y_{i-1}$ for each time of the day. Note that Equation 4.7 does not have a noise term; any noise is to be subsumed by the inflow/outflow uncertainty.

Volume passing the i-th segment endpoint during the interval $t : (t + \delta t)$ can be estimated by assuming that vehicles are homogeneously distributed throughout the segment. Then we can write what is known as

the "fundamental equation of flow". By analogy with gas flow, the *mass flow* $y_i^{t+\delta t}$ is proportional to the diameter of the pipe (number of lanes λ_i, the density of the gas ρ and the speed v_i:

$$y_i^{t+\delta t} \approx \lambda_i \rho v_i^t \delta t = C \lambda_i \frac{q_i^t}{L_i} v_i^t \delta t + \epsilon_i, \tag{4.8}$$

where ρ is the density (expressed in vehicles/mile), L_i is the length of the segment and ϵ_i is zero-mean Gaussian noise with variance ρ_i^2. C bundles the unit conversion constants. The noise variance can be estimated by a least squares procedure from observed volume data.

4.3.2 Volume-speed characteristic

The third term of the traffic model in Equation 4.6, $p(v_i|y_i)$, captures how speed on a particular segment of the road depends on its traffic volume. In our data, this relationship varies widely depending on physical properties of the road infrastructure.

In the transportation literature traffic volumes and speeds are related via the so-called fundamental diagram [82]. Customarily one expresses the fundamental diagram as a deterministic function [109], e.g. $v_i = f(y_i)$. [1] Intuitively, the diagram is intended to describe the "response" of the infrastructure to injection of traffic in terms of what speed it will permit. However, this expedient engineering approach ignores the random nature of the dependency, the existence of multiple flow modes and other aspects of traffic flow.

To address the limitations of the traditional deterministic model, we represent the relation between the volume and speed using the conditional density $p(v_i|y_i)$. We actually estimate the joint $p(v_i, y_i)$, separately for each segment i, from historical data by a kernel estimate. A small Gaussian kernel is centered on each data point $d^{(n)} = (y_i^{(n)}, v_i^{(n)})$, with axis-aligned covariance equal to the empirically observed variance in the respective covariates. The resulting probability distribution at $x = (v_i, s_i)$ is

$$p(x) = \sum_{n=1}^{N} \mathcal{N}(x; d^{(n)}, C_i), \tag{4.9}$$

where

$$C_i = \begin{pmatrix} \sigma_{v,i}^2 & 0 \\ 0 & \sigma_{s,i}^2 \end{pmatrix} \tag{4.10}$$

is the axis-aligned covariance matrix. Figure 9 shows an example of the volume-speed diagram implemented in the model. For efficiency, the joint $p(v_i, y_i)$ is pre-computed and replaced by a piecewise-constant approximation on a fine grid. This avoids iteration over all historical data points in the actual computation and the conditional is easily obtained from the joint as also shown in Figure 9.

An important feature of our model is that it allows us to relax the assumption that the traffic stream behaves the same way at all locations, which — in addition to determinism — characterizes most macroscopic

[1] In civil engineering literature, the term "fundamental diagram" often refers to volume-density relationship only. We use the term more loosely, meaning a volume-speed joint distribution throughout the thesis.

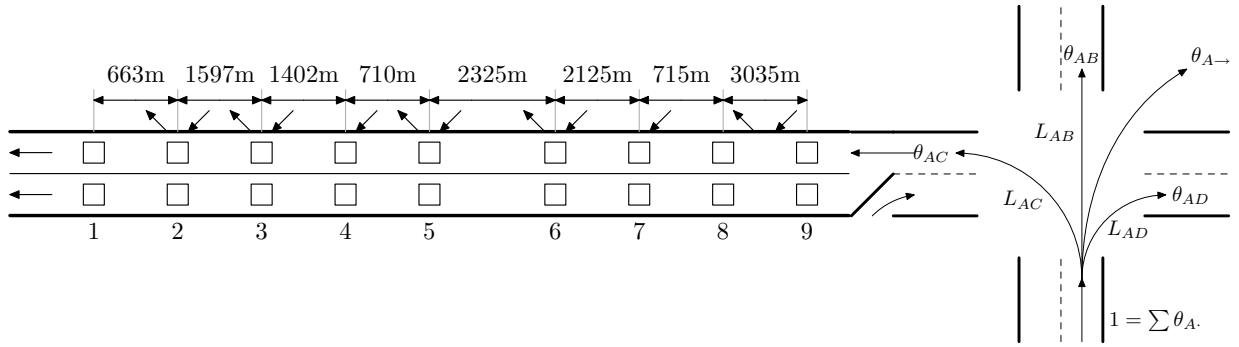


Figure 8: A schematic of a highway segment with its physical parameters obtained from the sensor database and the ArcGIS geographic information system. This highway has two lanes along most of its length, although some portions have three lanes (not shown). The squares represent loop sensor locations. There are no sensors on the entry and exit ramps and the flows on these must be estimated otherwise. Distances between sensors are noted at the top. Small diagonal arrows below the distances indicate presence of exits and onramps. Numbers on the bottom identify sensor stations and, at the same time, segments. Traffic moves from right to left, "westward". The right portion of the figure shows an intersection or interchange and how it connects to a road segment. The behavior of an interchange is governed by the O-D matrix (see text). While highway interchanges may be more complex than simple intersections, they are handled conceptually the same way.

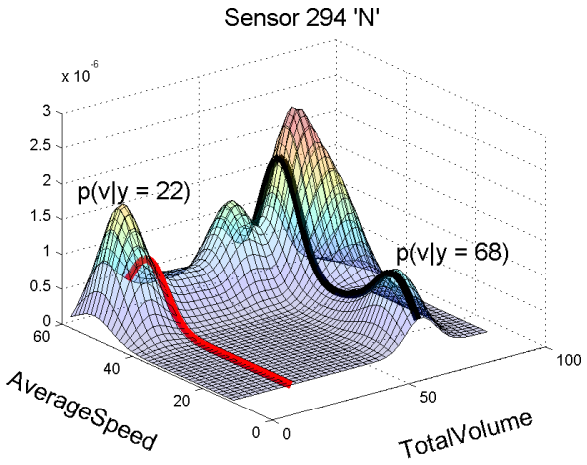


Figure 9: A fundamental diagram and the resulting conditional distributions (thick lines) for two values of flow volume.

models. In reality, the speed resulting from a certain number of vehicles attempting to pass is a stochastic quantity whose distribution changes from site to site. We extract a fundamental diagram for each sensor site. This is in contrast to work of Mihaylova et al who also proposed endowing macroscopic traffic flow model with a probabilistic component [115, 116] and ended up using a particle filtering solution for much the same reasons. However, their approach follows the civil engineering model more closely: instead of learning the volume-speed diagrams from data, they characterize speed as a parametric function of density ρ, which is linearly related to demand q (as in Equation 4.8). The parameters of this function are then tweaked until the resulting dependency resembles the observed data. We instead learn the volume-speed relationship directly. Details on the effect of choosing a different representation of the fundamental diagram can be found in Section 4.3.3.

4.3.3 Fundamental diagram representations

Other representations of the fundamental volume-speed relationship are possible. We set up an experiment to determine which of the candidate approaches performs the best. In all cases, we look at five locations selected so that they represent the breadth of variation in their shape. The alternative representations we considered were a piecewise linear model and a mixture-of-Gaussians model.

The piecewise linear model learns a stochastic functional relationship with speed v being the function of volume y. The stochastic function has two components, mean μ and standard deviation σ:

$$v = (\mu, \sigma)(y) \qquad \text{so that} \qquad p(v|y) = \mathcal{N}(\mu(y), \sigma(y))$$

The interval of volumes is represented by $M = 10$ points $y_1, \ldots, y_M$. It is not required that they be equidistant, but we make them so in the implementation. The mean and variance of a prediction at volume $y$is given by the linear interpolation between $\mu(y_i)$ and $\mu(y_{i+1})$

$$\mu(y) = \frac{(y_{i+1} - y)\mu(y_{i+1}) + (y - y_i)\mu(y_i)}{y_{i+1} - y_i} \qquad \sigma^2(y) = \frac{(y_{i+1} - y)\sigma^2(y_{i+1}) + (y - y_i)\sigma^2(y_i)}{y_{i+1} - y_i},$$

where y_i, y_{i+1} are the unique points such that $y_i \leq y \leq y_{i+1}$.

Conditioned on the volume point y_i, the mean speed and the variance is learned as the mean and variance of data points $\{(y_i, v_i)^{(n)}\}_{n=1}^{N}$ to which y_i is the closest volume point:

$$\mu(y_i) = E[v^{(n)} : |y^{(n)} - y_i| \leq \min_i |y^{(n)} - y_i|]$$

A 10-component Gaussian mixture was learned.

Lastly, we learned the non-parametric representation as described in Section 4.3.2. Conditional log-likelihood of withheld data points was chosen as the comparison metric:

$$LL = \sum_{n=1}^{N} \log p_*(v^{(n)}|y^{(n)}),$$

where $v^{(n)}$ and $y^{(n)}$ are the speed and volume components of the n-th data point from the testing set.

Site	270W	2557W	34W	36W	2775W
MG	-145.9 (21.6)	-137.1 (16.6)	-147.5 (17.0)	-149.3 (15.7)	-168.4 (25.7)
Lin	$-130.1(25.1)^{0.979}$	$-129.1(44.5)^{0.770}$	$-143.0(19.7)^{0.776}$	$-118.9(13.0)^{0.000}$	$-161.7(26.7)^{0.786}$
NP	$-131.1(15.0)^{0.440}$	$-121.5(12.3)^{0.765}$	$-128.3(12.2)^{0.995}$	$-120.0(11.1)^{0.359}$	$-138.6(12.0)^{0.999}$

Table 2: Experimental comparison of different FD representations: mixture of Gaussians (MG), Linear (Lin) and non-parametric (NP). The superscript is the value of cumulative t-distribution for significance. Values close to 1.0 indicate a highly significant improvement over the method above.

The data was split into training and testing sets in 70/30 ratio. The results are tabulated in the Table 2. The numbers are means and standard deviations over 20 train-test splits.

The results show that the most successful model of the fundamental diagram turned out to be the non-parametric representation. In all locations, it was either the best performer or statistically indistinguishable (in the sense of t-test significance) from the model achieving the best average score. Thus this is the representation we use.

We also observed that the result is quite robust to the choice of M for the "piecewise linear" FD model and the number of mixture components in the mixture-of-Gaussians (not tabulated).

The most appropriate model depends on the shape of the fundamental diagram. For instance, the sensor 270W experiences virtually no jams, no multimodal behavior and is therefore well even modeled by the linear model. On the other hand, sensor 34W exhibits a strongly multimodal behavior. It is unclear why this was not well picked up by the 10-component Gaussian mixture model, but it does demonstrate an advantage of the non-parametric model: it is not subject to random initialization of model parameters at the outset of EM learning and the resulting uncertainty of the learned representation.

4.3.4 Intersections and interchanges

The behavior of a intersection or interchange is characterized by an origin-destination (O-D) matrix D that specifies how an inflow to an intersection divides into the outflows. Moreover, the interchange will often have on- and/or off-ramps through which vehicles enter or leave the instrumented portion of the system. If there are roads A,B and C, an OD-matrix might look like this:

$$\begin{pmatrix} 0 & \theta_{AB} & \theta_{AC} & \theta_{A\rightarrow} \\ \theta_{BA} & 0 & \theta_{BC} & \theta_{B\rightarrow} \\ \theta_{CA} & \theta_{CB} & 0 & \theta_{C\rightarrow} \\ \theta_{\rightarrow A} & \theta_{\rightarrow B} & \theta_{\rightarrow C} & 0 \end{pmatrix}, \tag{4.11}$$

where θ_{AB} indicates the proportion of flow coming from A turning to B, while $\theta_{B\rightarrow}$ is the proportion of B's flow leaving the intersection. Intersections and interchanges may present an identifiability problem,

especially with unobservable in/out-flows. This problem is usually solved in civil engineering literature by least-squares estimation methods for the O-D matrix [9, 100, 171].

However, we only need to estimate the "turning proportion". Given an intersection with inputs I and outputs O, the goal is to find the proportion of flow coming in on link $i \in I$ that goes to $o \in O$. The difference is that there are no intermediate links involved. All links in I connect directly to all links in O, unless there are turning restrictions.

Under the assumption of approximate stationarity or smoothness of flow conditions, we can ignore the travel time through the intersection and treat the interchange as a "switch" with an immediate response.

Let $|I| = k$ and $|O| = \ell$. Let data be given in the form of count matrices with n data points, $G_{n\times k}$ and $H_{n\times \ell}$ respectively. Further, assume that the time of travel through the intersection is negligible. Then the turning matrix $T_{k\times\ell}$ transforms the input flow into the output flow as follows:

$$H = GT + N_{n\times\ell}, \tag{4.12}$$

where $N_{n\times\ell}$ is an unobserved noise matrix. Upon closer inspection, this equation simply expressed the dynamics of the intersection: the input flows are divided according to coefficients in the rows of matrix T and then recombined (summed) into the output flows. This is exactly what matrix product does.

To obtain the turning matrix, we need to solve the above equation, subject to the constraint that rows of T sum to 1. Thus, the constraint is enforced that T is a matrix of proportions and thus Equation 4.12 preserved total flow. Since N is unobserved, is is reasonable to view this as a minimization problem

$$\begin{aligned}
\underset{T}{\text{minimize}} \quad & ||N|| && && (4.13)\\
s.t. \quad & H && = GT + N && (4.14)\\
& T\mathbf{1}_{\ell\times 1} && = \mathbf{1}_{k\times 1} && (4.15)\\
& T && \geq \mathbf{0}_{k\times\ell}, && (4.16)
\end{aligned}$$

where the first constraint is just what Equation 4.12 does and the last two constraints say that T is a proper matrix of proportions. $||\cdot||$ is some measure defining what is a "small" matrix. The most popular such measure is the Frobenius norm

$$||N||_F = \sum_{i,j} n_{ij}^2 = \sqrt{\text{tr}(N^T N)}.$$

Minimizing the Frobenius norm minimizes the squared error. The problem then becomes a quadratic optimization problem with linear constraints

$$\begin{aligned}
\underset{T_{\cdot\cdot}}{\text{minimize}} \quad & \sum_{m,j} (H_{mj} - G_{m\cdot} \cdot T_{\cdot j})^2 && && (4.17)\\
s.t. \quad & \sum_j T_{ij} = 1 && \forall i && (4.18)\\
& 0 \leq T_{ij} \leq 1 && \forall i, j && (4.19)
\end{aligned}$$

This is easily solved using Matlab's `quadprog` function after expanding the objective and collecting the coefficients of T_{ij}.

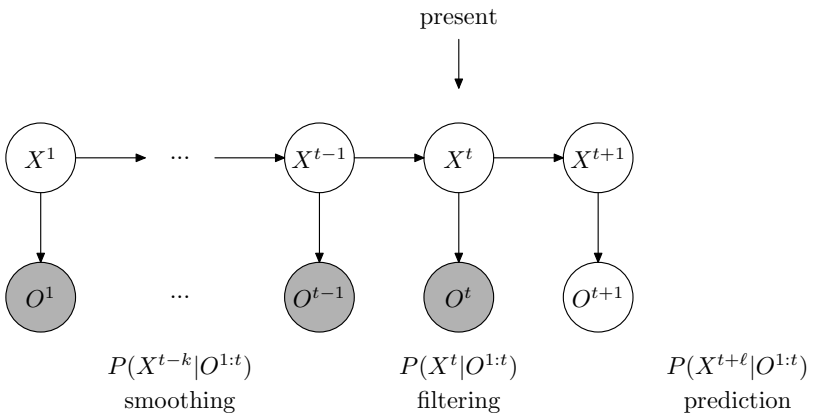


Figure 10: The three basic inference tasks in probabilistic dynamic models: smoothing, state tracking and prediction. Filtering and state tracking are synonymous. X represents the unobserved variables, O the observed ones and superscript indexes time.

4.3.5 Inference

Our main objective is to use the above dynamic model to support traffic predictions. The main inferential challenge to this task is presented by the types of probability distributions defined by the dynamic model. The flow balance equation uses an empirical distribution of segment inflows and the fundamental diagrams bring in essentially arbitrary distributions of speeds. Because of this, distributions of traffic quantities for any future time do not warrant the use of "easy" parametric distributions.

We apply a Monte Carlo scheme [46] to deal with these complex distributions. Let us denote the time of last observation by T. The model of dynamics can be equated to a dynamic graphical model [119], a small piece of which is shown in Figure 11. Our prediction goal corresponds to the probabilistic query

$$p(\mathbf{v}^{T+1:T+H}, \mathbf{y}^{T+1:T+H} | \mathbf{v}^{1:T}, \mathbf{y}^{1:T}), \tag{4.20}$$

where H is the prediction horizon. With dynamic probabilistic models, there are three basic inference tasks: filtering, state tracking and prediction. See Figure 10 for their description.

We need to track the hidden state (demand, number of cars on a segment) $\mathbf{q}$ until time T (with $\mathbf{v}$ and $\mathbf{y}$ observed) and predict from there on, with $\mathbf{v}$ and $\mathbf{y}$ unobserved.

We keep a set of P particles of the form $p^{(k)} = (\mathbf{q}^{(k)}, \mathbf{v}^{(k)}, \mathbf{y}^{(k)}, w^{(k)})$, where $w^{(i)}$ is the particle's weight. In the first step, we initialize all $q_i^{(k)}$ to[2] $CL_i y_i^1 / v_i^1$ and all weights $w^{(k)}$ to $1/P$. $\mathbf{y}$ and $\mathbf{v}$ are initialized to their appropriately scaled observed values.

In subsequent time steps, we

[2] This is a "maximum likelihood" value of q, cf Equation 4.8.

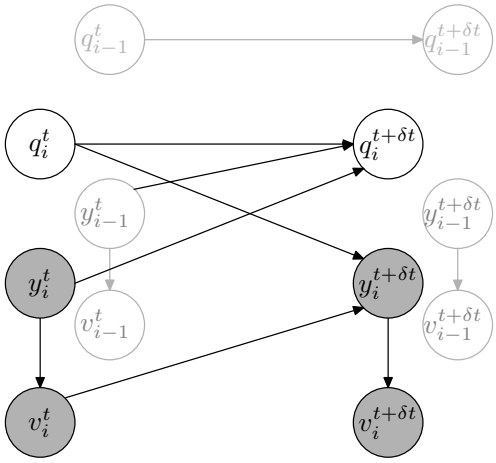


Figure 11: Traffic evolution at i-th segment as a dynamic Bayesian network. An identical "plate" associated with the neighboring $i-1$-st segment is shown in gray, with some of its arcs omitted for clarity. Also omitted are the inflow random variables io_i^t, which would be additional parents to their respective q_i^t. Variables $\mathbf{y}$ and $\mathbf{v}$ will be observed only when the model is used for state tracking. $\mathbf{q}$ is always unobserved.

- take each particle $p^{(k)}$, and create a new particle $p'^{(k)}$ as follows. To get the q-component, we apply Equation 4.7 to the constituent parts of $p^{(k)}$. The inflow/outflow is represented by a sample from the empirical inflow distribution io_i.
- When state tracking, $\mathbf{v}$ and $\mathbf{y}$ are observed and we can directly proceed to weighting the new particle. When predicting, the speed and volume are unavailable and we obtain a complete particle as follows. To get the y-component, we use Equation 4.8, making y and q independent given their previous values. Because of this, it is important that the model step is relatively fine, so that correlations between time steps are tight, expressing the demand/volume dependency.
- The speed component v is the most interesting. It is obtained as a sample from the conditional obtained by slicing through the joint distribution (fundamental diagram) of volume and speed for the particular location, as illustrated by Figure 9:

$$v'^{(k)}_i \sim p(v_i | y_i = y_i^{(k)}). \tag{4.21}$$

- Finally, in filtering the weight of the new particle $p'^{(k)}$ is updated with the likelihood of the observation $(\mathbf{y}_{obs}, \mathbf{v}_{obs})$:

$$w'^{(k)} = w^{(k)} \prod_{i=1}^{S} p(y'_i = y_{obs} | q_i^{(k)}, y_i) p(v'_i = v_{obs} | y_i^{(k)}) \tag{4.22}$$

The property that $\sum_k w^{(k)} = 1$ is maintained by renormalization in each step. When no observation is available (during prediction and in the many model steps that occur between consecutive observations), the particles are updated by simple forward sampling.

4.4 EXPERIMENTAL EVALUATION

In the following experiments, we are interested in how well can the dynamic model predict future state of traffic.

4.4.1 Data

We consider I-376 westbound, a major metropolitan highway in Pittsburgh, USA, whose schematic is in Figure 8 (page 34). This highway has 9 sensors installed that collect the volume (number of vehicles) and the average speed of travel in 5-minute intervals. Data from adjacent lanes are aggregated. Our data consists of 165 sequences of length $M = 18$ – this corresponds to 1.5 hours as each element of the sequence corresponds to one 5-minute measurement. One sequence is extracted from the morning peak portion of each workday and sequences that have missing data from any of the sensors are discarded.

In experiments, the 165 sequences are randomly divided into a training and a testing set. In each "fold", the training set consists of randomly selected 70% of the sequences. The remaining 30% form the testing set.

4.4.2 Experimental setup and baselines

We will compare three methods of predicting future traffic volume. First, we try a very simple model that always predicts the previous state. We add a Gaussian noise term $\epsilon \sim N(0, \sigma^2)$ fitted to training data to get a probability distribution on future outcomes:

$$y^{t+1} = y^t + \epsilon \tag{4.23}$$

Since traffic state usually does not change very abruptly, this is often a very reasonable predictor.

Secondly, we will use the $ARMA(p, q)$ model [18]: The first sum expresses the Markovian dependence on previously observed values. The second sum carries the effect of noise into subsequent time points. The parameterization $(\boldsymbol{\varphi}, \boldsymbol{\theta})$ is learned from the training set using standard libraries available with MATLAB. Since the order of an optimal ARMA model varies with location, we try all combinations of p, q for $p = 1, \ldots, 4$, $q = 0, \ldots, 2$ and compare our dynamic model to the best-performing ARMA variant to establish a stronger baseline.

Lastly, we will use our dynamic model with 8 segments. The model is run at resolution $\delta t = 20s$. Inference in this model uses $P = 100$ particles. The number was chosen so as to allow faster than real-time prediction on an older desktop machine and could be increased as needed and permitted by quality of implementation.

4.4.3 Evaluation metrics

We are interested in predicting the flow in time horizons of $\Delta t = 5, 10, \ldots 50$ minutes (next 10 observations) given the most recent state of the traffic flow y^t. The three models respectively give the probability distributions:

$$p_0(\mathbf{y}^{t+\Delta t}|\mathbf{y}^t), \quad p_{AR}(\mathbf{y}^{t+\Delta t}|\mathbf{s}^{t-p:t}) \text{ and } p_{dyn}(\mathbf{y}^{t+\Delta t}|\mathbf{s}^{t-p:t}),$$

where $\mathbf{s}^t$ is the most recently observed traffic state and $\mathbf{s}^{t-p:t}$ denotes the sequence of the p most recent states. Each testing sequence (18 consecutive observations) is entirely used in experiments; when prediction horizon is H, the remaining $18 - H$ points is used for conditioning, i.e. the distribution we actually compute is $p(\mathbf{y}^{H+1:18}|\mathbf{s}^{1:(18-H)})$. The length of initialization (burn-in) makes a difference in a hidden state model such as our dynamic predictor.

We measure and report the performance separately for each sensor location. Two metrics are used, the root mean square error (RMSE) and the likelihood of actually observed data. The RMSE is computed as the difference between the actually observed value $y^{t+\Delta t}$ and the mean of the predictive distribution $p(\hat{y}^{t+\Delta t}|\mathbf{s}^t)$, averaged over the testing examples (indexed by the subscript $_{(j)}$):

$$e_\ell^{\Delta t} = \left(\frac{1}{N_{test}} \sum_{j=1}^{N_{test}} (E_p[\hat{y}_{(j)}^{t+\Delta t}] - y_{(j)}^{t+\Delta t})^2 \right)^{1/2} \tag{4.24}$$

To avoid cluttering the notation, we noted the location ℓ for which we predict in $e_\ell^{\Delta t}$, but not in $y_{(j)}^t$. The values $y_{(j)}^t$ in the above equation should be understood to be those associated with the location ℓ.

The likelihood metric is defined as

$$L_\ell^{\Delta t} = -\frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \log \hat{p}(y_{(i)}^{t+\Delta t}|\mathbf{s}_{(i)}^{1:t}) \tag{4.25}$$

and measures how well the predictive distribution fits the actually observed values.

Since the quantity to predict is continuous, a distribution represented by a finite number of weighted particles will almost surely assign the probability of 0 to any particular value. We obviate the problem by smoothing the particle set into a non-parametric distribution $p(y)$. The respective parameters of the non-parametric kernels are identical to those we used to model the volume-speed relationship.

4.4.4 Experimental results

Speed. A C++ implementation of the dynamic model runs at about 10 times real time with approximately 75 segments in the networks, $\delta = 20$ sec simulation step and $P = 100$ particles on a 6-year old desktop machine. That is, the model will take about 20 seconds to predict three minutes into the future in this setting. The model would be very natural to parallelize if needed. Off-line learning of the model (fundamental diagrams, inflow/outflow distributions) takes approximately 30 seconds per segment in Matlab connecting to the MySQL server.

ARMA off-line learning takes a few seconds per sensor as implemented in Matlab on the same hardware; prediction is nearly instantaneous.

RMS error. First, we have tabulated the RMS error for volume prediction by the best model in Table 3. Overall, ARMA(2,0) was most frequently the best time-series model. As one would expect, prediction tends to deteriorate with longer prediction horizons for both the ARMA model and the naive baseline. The ARMA results are rather disappointing for this medium-term (tens of minutes) prediction. Previous applications extended mostly into the short-term only (at most minutes).

The results obtained by the dynamic model are tabulated in Table 4. In the short term, our dynamic model mostly loses to the best time-series approach. In longer term, the dynamic model's prediction improves over that of ARMA.

Also, the downstream segments are generally better predicted by our model. This is intuitively understandable: when the downstream segments receive the vehicular flow, much information has already been collected upstream about the approximately same batch of vehicles. Upstream segments benefit much less from this information.

Likelihood metrics. The corresponding likelihood metric results are tabulated in Table 5 and Table 6. They show that ARMA provides a much better estimate of the uncertainty of the prediction than the naive model even if the latter predicts the mean almost as well.

The results in Table 6 suggest that the dynamic model provides a much better probabilistic description of traffic than ARMA models. Figure 12 shows some cases where the dynamic model assigns a likelihood that differs significantly from that obtained from the ARMA model. In general, volume distributions tend to be skewed to the right. This biases the Gaussian time series approach to underestimate volume. It is plausible that a generalized ARMA model using a skewed distribution (e.g. Poisson) could remedy this problem.

There is also evidence of multiple modes in the predictive distribution. In such a distribution, many samples can be actually quite far from the mean value. The location and time of day examined here is prone to bifurcation phenomena (free flow/jam), so this seems a likely explanation.

Overall, two regions of successful prediction by the dynamic model can be identified in the results: that of longer-term downstream prediction and that of the two upstream sensors. The first is caused by better information while the explanation for the second appears to lie in the physical topology. The dynamic model is also "less surprised" by the future data which points to better modeling of traffic quantity variation.

4.5 CONCLUSIONS

Adding knowledge about the physical process of traffic flow resulted in more accurate predictions of traffic speeds in situations that are more difficult to handle with time-series approach, especially longer prediction horizons. Also, traffic at sites located further downstream is better predicted by the dynamic model as it

takes advantage of more observable data: downstream sites are seeing quantities of cars that the model has been observing for some time.

The model connects several linear highway segments by interchanges, whose "mixing function" is well described as multiplication by a turning proportion matrix. Estimation of the turning proportion matrix itself from data is a quadratic estimation problem.

Our results offer a better alternative to ARMA prediction in medium term, but further investigation into the effects of the physical arrangement of the infrastructure is necessary as predictability is clearly correlated with measuring site location. We still have little understanding of how physical roadway features affect prediction. Clearly, the presence of an unmeasured on/off-ramp disturbs prediction. Also, a change in the number of lanes brings about effects that may not be fully captured by the fundamental diagram extracted from a sensor that may be removed several hundred meters from such feature. Other roadway features such as quality of the surface, curvature and grade may have observable effects. Such effects should be learnable given enough sensors and data on the physical infrastructure in which they are embedded.

The question of segmenting the roadway more finely than we did here also needs to be explored.

Given the requirement of a fine "simulation" step, an approach that dispenses with the step altogether in favor of a continuous Markov network [48] formulation seems worth looking into. The challenge of that approach is the incorporation of non-Gaussian fundamental diagram and inflow/outflow distributions.

Currently our model assumes that the input parameters, mainly the in/out-flows, are predetermined. We have no model of how they would change given a control action, with the exception of ramp metering. For instance, how they would react to a "Congestion ahead" sign, is a function of aggregate driver behavior which we do not cover here. Such behavior would be better evaluated in a microscopic simulator.

Sensor	mean	$\Delta t = 1$	$\Delta t = 2$	$\Delta t = 5$	$\Delta t = 7$	$\Delta t = 10$
1	193.5	$23.6(2.3)^{0.712}$	$28.2(5.8)^{0.943}$	$42.6(3.0)^{1.000}$	$52.7(2.5)^{1.000}$	$61.1(3.5)^{1.000}$
2	326.0	$44.6(3.6)^{0.967}$	$34.0(4.7)^{0.958}$	$78.8(6.1)^{1.000}$	$94.4(7.3)^{1.000}$	$92.6(7.8)^{1.000}$
3	357.0	$37.4(2.6)^{0.216}$	$40.6(5.8)^{0.597}$	$67.9(2.2)^{1.000}$	$79.5(1.9)^{1.000}$	$88.6(2.7)^{1.000}$
4	255.7	$29.5(3.1)^{0.766}$	$39.0(5.5)^{0.991}$	$39.7(8.1)^{0.999}$	$39.1(11.0)^{1.000}$	$38.4(7.4)^{1.000}$
5	280.7	$28.1(6.7)^{0.741}$	$30.2(7.7)^{0.965}$	$33.7(8.1)^{0.999}$	$29.8(5.3)^{0.999}$	$33.4(9.8)^{1.000}$
6	276.2	$20.7(1.2)^{0.341}$	$32.4(13.1)^{0.668}$	$29.9(7.1)^{0.954}$	$36.8(4.0)^{0.999}$	$39.2(6.7)^{0.997}$
7	263.2	$40.6(5.8)^{0.009}$	$40.4(2.9)^{0.000}$	$45.9(4.0)^{0.000}$	$47.9(6.3)^{0.000}$	$48.7(3.1)^{0.000}$
8	206.1	$31.1(2.8)^{0.544}$	$38.7(4.7)^{0.000}$	$40.1(4.4)^{0.000}$	$40.6(3.7)^{0.003}$	$42.1(7.4)^{0.487}$

Table 3: Volume RMS error results for best ARMA models. Rows correspond to segments, downstream segments have the lower numbers. Columns represent selected prediction horizons. Prediction horizons are multiplies of the observation period which is 5 minutes. The numbers in parentheses represent standards deviations computed over 20 experimental folds. Superscripts close to 1.0 indicate significantly better performance of ARMA than the naive model (numbers not shown in interest of space) under an unpaired heteroscedastic t-test. Highlighted cells are those where the ARMA model is better at $\alpha = 0.01$. To help assess the relative magnitude of the error, each segment's average 5-minute flow is given in the first column. Note the anomalous sensor 7, ARMA learning probably diverged there.

Sensor	$\Delta t = 1$	$\Delta t = 2$	$\Delta t = 5$	$\Delta t = 7$	$\Delta t = 10$
1	$44.4(3.3)^{0.000}$	$29.3(8.1)^{0.314}$	$27.3(7.0)^{1.000}$	$31.3(8.7)^{1.000}$	$34.7(6.3)^{1.000}$
2	$73.8(1.4)^{0.000}$	$58.9(6.9)^{0.000}$	$62.5(11.2)^{1.000}$	$57.8(12.2)^{1.000}$	$59.6(15.0)^{1.000}$
3	$50.7(12.2)^{0.000}$	$44.3(23.3)^{0.249}$	$47.3(11.8)^{1.000}$	$48.9(8.1)^{1.000}$	$51.6(24.4)^{1.000}$
4	$47.5(11.9)^{0.000}$	$40.6(17.7)^{1.000}$	$49.5(8.0)^{0.000}$	$38.2(13.9)^{1.000}$	$30.7(10.9)^{0.992}$
5	$38.1(18.5)^{0.018}$	$42.2(18.2)^{1.000}$	$40.1(11.6)^{0.029}$	$32.5(4.3)^{1.000}$	$32.0(16.3)^{0.627}$
6	$40.1(11.2)^{0.000}$	$47.2(17.7)^{1.000}$	$40.8(11.9)^{0.001}$	$41.4(5.9)^{0.005}$	$42.1(12.7)^{0.189}$
7	$53.5(9.7)^{0.000}$	$51.8(18.2)^{1.000}$	$51.0(11.7)^{0.040}$	$48.0(12.1)^{0.487}$	$46.1(8.5)^{0.893}$
8	$43.3(4.5)^{0.000}$	$28.3(4.1)^{1.000}$	$25.2(7.9)^{1.000}$	$23.8(7.3)^{1.000}$	$26.5(3.1)^{1.000}$

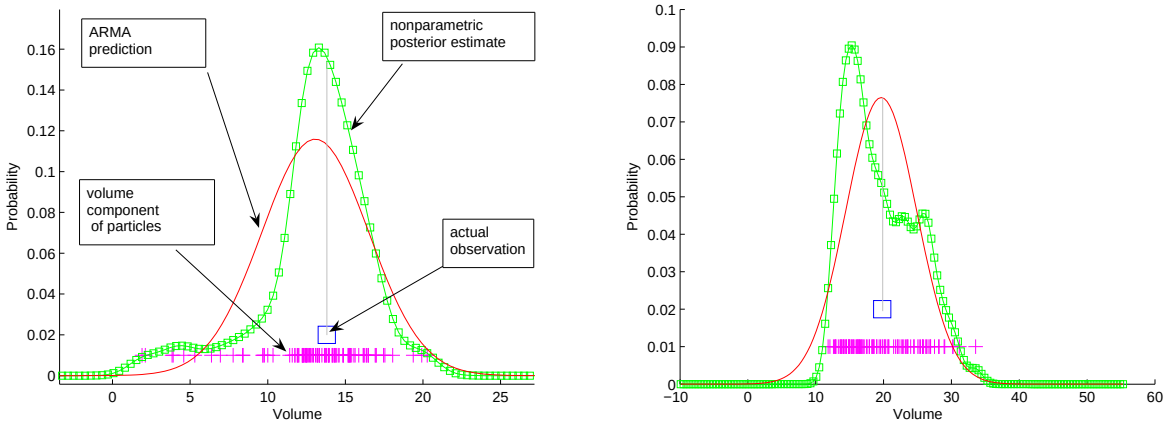
Table 4: Volume RMS error results for the dynamic model, segments in columns, prediction horizon in rows. The superscripts are the values of Student t cumulative distribution for significance of difference. Values near 0.0 indicate ARMA is better, values near 1.0 mean the dynamic model is better. Highlighted are the instances where the dynamic model is better. Values smaller than 0.001 or larger than 0.999 are reported as 0.000 and 1.000, respectively.

Sensor	$\Delta t = 1$	$\Delta t = 2$	$\Delta t = 5$	$\Delta t = 7$	$\Delta t = 10$
1	$4.68(0.04)^{0.000}$	$4.73(0.08)^{0.000}$	$5.18(0.05)^{0.000}$	$5.49(0.08)^{1.000}$	$5.61(0.11)^{1.000}$
2	$5.19(0.06)^{0.000}$	$5.13(0.04)^{0.000}$	$5.67(0.08)^{1.000}$	$5.85(0.02)^{1.000}$	$5.91(0.11)^{1.000}$
3	$5.20(0.05)^{0.000}$	$5.23(0.09)^{1.000}$	$5.64(0.04)^{1.000}$	$5.79(0.02)^{1.000}$	$5.83(0.11)^{1.000}$
4	$4.98(0.15)^{0.322}$	$5.08(0.19)^{0.001}$	$5.34(0.11)^{0.001}$	$5.39(0.16)^{0.000}$	$5.40(0.10)^{0.000}$
5	$4.91(0.08)^{0.116}$	$4.85(0.09)^{0.002}$	$5.24(0.08)^{0.000}$	$5.13(0.14)^{0.000}$	$5.13(0.16)^{0.000}$
6	$4.72(0.07)^{0.033}$	$4.88(0.18)^{0.118}$	$5.06(0.12)^{0.000}$	$5.12(0.09)^{0.000}$	$5.18(0.17)^{0.007}$
7	$5.21(0.10)^{0.995}$	$5.08(0.12)^{1.000}$	$5.27(0.07)^{1.000}$	$5.42(0.11)^{1.000}$	$5.38(0.11)^{1.000}$
8	$4.90(0.05)^{1.000}$	$5.30(0.40)^{1.000}$	$5.26(0.10)^{1.000}$	$5.20(0.11)^{1.000}$	$5.19(0.05)^{1.000}$

Table 5: Likelihood metric results, ARMA/naive model. Larger values are worse. A highlighted cell means the ARMA model is significantly better at $\alpha = 0.01$.

Sensor	$\Delta t = 1$	$\Delta t = 2$	$\Delta t = 5$	$\Delta t = 7$	$\Delta t = 10$
1	$3.62(0.12)^{+}$	$3.63(0.04)^{+}$	$3.66(0.11)^{+}$	$3.72(0.16)^{+}$	$3.73(0.14)^{+}$
2	$3.11(0.07)^{+}$	$3.09(0.02)^{+}$	$3.12(0.02)^{+}$	$3.11(0.05)^{+}$	$3.13(0.03)^{+}$
3	$3.07(0.08)^{+}$	$3.08(0.04)^{+}$	$3.10(0.07)^{+}$	$3.04(0.02)^{+}$	$3.07(0.01)^{+}$
4	$3.08(0.04)^{+}$	$3.11(0.01)^{+}$	$3.11(0.10)^{+}$	$3.06(0.04)^{+}$	$3.05(0.02)^{+}$
5	$2.94(0.02)^{+}$	$2.95(0.02)^{+}$	$2.92(0.01)^{+}$	$2.89(0.02)^{+}$	$2.90(0.01)^{+}$
6	$3.22(0.08)^{+}$	$3.21(0.11)^{+}$	$3.21(0.09)^{+}$	$3.18(0.13)^{+}$	$3.26(0.14)^{+}$
7	$3.67(0.17)^{+}$	$3.53(0.17)^{+}$	$3.56(0.09)^{+}$	$3.49(0.21)^{+}$	$3.36(0.10)^{+}$
8	$3.05(0.01)^{+}$	$3.09(0.03)^{+}$	$2.99(0.02)^{+}$	$3.01(0.03)^{+}$	$3.00(0.04)^{+}$

Table 6: Likelihood metric results, dynamic model. The dynamic model is significantly better than ARMA in all cases, with negligible p-values, all smaller than 10^{-15}.



(a) Example of dynamic model posterior

(b) Local flow properties

Figure 12: **a)** A frequent case of a volume posterior distribution skewed to the right, with a descriptive legend. **b)** A rarer case of a left-skewed posterior. Accidentally, the ARMA prediction is actually slightly better here. The volume is so small because it is scaled into the $\delta t = 20s$ model step interval.

5.0 ROUTING PROBLEMS IN TRAFFIC NETWORKS

Summary. *We consider the problem of finding a minimum expected-time path through a road network. This is an environment-navigation problem that can be decomposed by assuming that the actions of the single agent do not influence the overall evolution of the surrounding environment. Then we are faced with two distinct problems: how to model the environment and how to make agent decisions given that the model is in hand. Our answer to the former is in the previous chapter. This chapter addresses the latter question. General MDP-solving approaches must view the state as unstructured. The first proposed algorithm follows this structure, searching through the traffic state subspace and the location subspace simultaneously and addressing the challenge of continuous state space by sampling. Then, we explore a complementary approach in a new planning algorithm called k-robust planning, which explores through the traffic state subspace first and only then acts in the location subspace. k-robust is rooted in the intuition that a plan should be "robust" with respect to likely future states of the environment. We show that the proposed k-robust algorithm is good for environments with high volatility.*

5.1 INTRODUCTION

Our objective is to develop methods for planning shortest paths through a road network that take advantage of the up-to-minute sensor data now available. With more data available, classical formulations of the shortest path problem are no longer the best we can do. For instance, traffic flows can abruptly transition into a different mode with different costs. If we can predict or learn of a congestion forming, then choosing an alternate route may be the optimal decision that would not have been possible without online data.

Shortest path problems are among the oldest and most thoroughly studied algorithmic questions. The simplest form of the shortest path problem is deterministic shortest path, solved with graph algorithms such as A^* [136]. In graphs the size of a country's road network, hierarchical search algorithms [124] work much faster, since road networks have a naturally hierarchical structure[1]. Shortest path

The on-line availability of sensor information changes the problem into an instance of a sequential decision making problem that can be modeled as a Markov decision process [13, 14]. Since the time it takes MDP and SMDP

[1] Taking a freeway as opposed to a side road would be called a macro-step in planning jargon.

to execute a driving decision varies greatly, we find it more advantageous to formulate our problem as a Semi-Markov decision process [70, 131], which represents time explicitly.

Use of dynamic model

Most work concerning sequential decision making begins with assuming that the transition probabilities and cost distributions are given. But where do they come from? This is one of the main motivations of the new model of flow dynamics given in the previous chapter. A dynamic model that enables probabilistic traffic state prediction is necessary for planning that considers the cost of future actions.

Both random and time dependent

Two aspects of cost structure make optimization difficult in the context of real-world transportation networks. First, the action costs are stochastic. By itself, this is not a problem if we seek minimization of expected time. Let $\langle x_1, x_2, \ldots, x_N \rangle$ be a route, i.e. a sequence of locations to travel through. The time cost of such a route is

$$cost(\langle x_1, x_2, \ldots, x_N \rangle) = \sum_{n=1}^{N} cost(x_n). \tag{5.1}$$

The total expected cost is simply a sum of expected link costs, by linearity of expectation:

$$E[cost(\langle x_1, x_2, \ldots, x_N \rangle] = \sum_{n=1}^{N} E[cost(x_n)]. \tag{5.2}$$

Thus, we can simply use expected edge costs in search and the problem is solved.

Secondly, the costs depend on the time at which the action is executed. Now each expectation E is over a different distribution. Over which distribution do we take the expectation of x_N? That distribution is $p(t_1 + t_2 + \cdots + t_{N-1})$, where no two time costs $t_i = cost(x_i)$ and $t_j = cost(x_j)$ are independent. They are tied together by the evolution of traffic state.

Hall [60] shows that having both randomness and time dependence changes the problem much more fundamentally than either of the two properties alone and he gives a branch-and-bound algorithm for the fastest expected time path. It is not surprising that the algorithm is exponential, seeing as the problem of finding a minimum expected time route when cost distributions are arbitrary and time-dependent is NP-complete [52].

Contribution

In this chapter we develop two classes of stochastic shortest path algorithms that build on the flow dynamics model. First, we build algorithms that take into account the initial traffic state information as observed by the sensors. Two algorithms are proposed in this work: SampleSearch and k-robust. Second, we develop on-line planning algorithms that assume continuous sensor information feedback to aid the routing procedure.

SampleSearch

The traffic routing problem can be formalized as a Markov decision process (MDP). At the heart of an MDP is the so called Bellman equation [11] that describes the structure of an optimal solution. In our problem, however, the expectation in the Bellman equation is over a complex continuous distribution of traffic state variables. The intractable computation can be approximated by sampling the successor state. In the first algorithm presented here, SampleSearch, we adapt A^* to search through the state space using a small number of samples. Combinatorial explosion is avoided by collapsing identical or very similar samples when possible and culling the population of samples back to constant size by sub-sampling.

The k-robust algorithm first obtains from the dynamic model a set of possible state trajectories of the environment. It uses a fast deterministic planner to get the plans that are optimal with respect to each of these trajectories. Finally, it evaluates the candidate paths and selects the one optimizing the planning objective. This relies on the assumption that the driver actions will not have an observable effect on the evolution of the environment. k-ROBUST

One of the advantages of the k-ROBUST algorithm — and, to a smaller degree, SAMPLESEARCH — is that to select the best, most robust plan, any optimization criterion can be applied, even though in the present work we apply it to problems where minimal expected time is the criterion. In the evaluation phase of k-ROBUST, we need only to actually compute the value of the chosen criterion. For SAMPLESEARCH, we could, for instance, order the search according $u(f(s))$, where $u(\cdot)$ is the utility function, and f the estimated node cost distribution as defined by the search algorithm. The properties of such search algorithm are not immediately clear and will depend on the choice of u. For the algorithms, the ability to work with non-linear utilities opens many interesting application domains where expected cost is not the dominant optimization criterion.

The algorithms are compared to other approaches on the task of planning a minimal expected-time route through a transportation network. The results show that injecting up-to-date information indeed improves route choices, as expected. Also, they suggest that while our algorithms are not the best method for the rather predictable Pittsburgh road network, the k-ROBUST algorithm is better in highly volatile environments.

5.2 STATE OF THE ART

Let us briefly review the applicable formalisms and methods pertinent to the navigation problem, addressing in sequence our various challenges, such as a continuous state space, non-stationarity and complexity considerations.

5.2.1 Shortest path

Standard deterministic fixed-cost shortest path is a problem long considered solved. The problem is defined by a graph $G = (V, E)$ with a distinguished node t (from $target$) and a function $c : E \to \mathbb{R}$ defining the cost to traverse an edge. The solution is a successor function $s : V \to V$ such that $(v, s(v)) \in E$ and the sum of edge costs on path defined by s from any node v to t is minimal. The solution is given by a class of dynamic programming algorithms containing Dijkstra's shortest path [45] and its heuristically pruned variant, A^* [136].

5.2.2 Markov Decision processes

The states of traffic are generated by a stochastic process. Decision making of an agent under the resulting uncertainty may be modeled by a MDP.

Definition 2 *A Markov decision process (MDP) is a tuple* (S, A, P_T, R) *where*

- S *a set of states*
- A *is a set of actions. Only a subset* $A(s)$ *may be available in a state* s*.*
- P_T *is a transition kernel specifying the distribution of the next state* $P_T(s'|s, a)$*.*
- $R : S \to \mathbb{R}$ *is a reward function.*

The agent is in a state s and can take one of the actions A. Upon choosing the action a, the universe transitions to a state s' according to the probability distribution $P_T(s'|s, a)$ and the agent receives a reward $R(s')$. It is useful to imagine that the agent's choice of actions is guided by a *policy*:

Definition 3 *A policy associated with a MDP* (S, A, P_T, R) *is a function* $\pi : S \to A$*. Following a policy means that the agent always chooses action* $\pi(s)$ *in state* s*.*

Solving an MDP means finding a policy π that maximizes the reward

$$\sum_{t=0}^{H} E[R_\pi(s^t)\gamma^t], \tag{5.3}$$

where H is the planning horizon (maximal number of steps), $0 < \gamma \leq 1$ is a discount factor, R_π is the reward received when following policy π and the expectation is taken over the transition kernel distribution.

Solving MDPs

Traditionally, this is accomplished by one of the techniques based on either value or policy iteration. Both are based on the fundamental Bellman equation [10]:

$$\pi(s) = \operatorname*{argmax}_a \sum_{s'} P_T(s'|s, a)V(s') \tag{5.4}$$

$$V(s) = R(s) + \sum_{s'} P_T(s'|s, a)V(s') \tag{5.5}$$

These equations can be viewed operationally as assignments to the term on the left side. Value and policy iteration differ in how the updates are ordered. Policy iteration performs the search in the policy space. Thus, every time the policy changes (policy improvement), the value function updates are propagated until (near) convergence (policy evaluation). Value iteration, on the other hand, does not explicitly represent the policy, but directly maximizes over all possible actions:

$$V(s) = R(s) + \max_a \sum_{s'} P_a(s'|s, a)V(s'). \tag{5.6}$$

For convergence with infinite horizon and to model temporal aspects of utility, a discount factor γ is often introduced just before the sum over successor states s'.

Forward search algorithms essentially perform a tree roll-out [66] of the MDP which can be solved with the expectimax algorithm. However, it is not clear how to roll out an MDP with a continuous state space if the transition kernel is not tractable.

5.2.3 Stochastic shortest path

Heving defined MDPs, we are now ready to discuss a natural generalization of the shortest path problem in the direction of randomness, the stochastic shortest path (SSP) problem. [2] [13, 14] The stochastic shortest path problem is formally equivalent to a Markov decision process, with the reward structure and objective set up so that it models the problem of reaching a goal state t. The optimal (stationary) policy in SSP is such that the expected cost of reaching t is minimal and following it results in eventually reaching it with probability 1.

5.2.4 Semi-Markov models

One way to introduce time dependencies into the MDP model is to use the semi-Markov decision process (S-MDP) [73, 131]. Semi-Markov decision processes generalize MDPs in that the system evolution is modeled in continuous time. [3]

Definition 4 *A Semi-Markov decision process is a tuple* (S, A, F, P_T, R) *where* S *is the state space as before;* A *is the set of actions and* $R : S \times T \to \mathbf{R}$ *is the reward function. Moreover,*

- F *is the set of conditional probability distributions specifying the probability of next transition time:* $F(t'|s, a, t)$*, where* $s \in S$ *and* $a \in A$*.*
- P_T *is the transition function conditioned on the time of transition:* $P_T(s'|t, s, a)$*.*

Under a Semi-Markov decision process, the execution of an action takes a variable amount of time. The agent finds itself in state $s \in S$ at time t and chooses an action $a \in A(s)$. The action execution is atomic and takes until t', sampled from the distribution F. After the transition, the world is found in a state s', the time is t' and the agent receives a reward $R(s', t')$. Our solution described in the next section builds on the semi-Markov decision process model.

5.2.5 Other approaches to non-stationary cost distributions

Sometimes, more problem-specific ways of incorporating time into the decision process can yield more efficient solutions. Boyan et al [19] proposed an exact algorithm for solving special time-dependent MDP by folding the time into the state and representing the value function as a piecewise linear function. In such a case, dynamic programming may be applied to obtain the sequence of piecewise-linear value function approximations. More recently, Meuleau has extended this approach to the rover planning problem [20]. Unfortunately, our semi-Markov model of the traffic network dynamics does not appear to warrant such

[2] While we have the transportation network in the back of our head, it is important to avoid a mixed metaphor here: the states in the SSP problem are not merely the positions of the agent in the network, but also involve the much larger state space of the environment.

[3] Conventionally, the seemingly more natural term "continuous time MDP" requires that the state space be discrete and the transition times are distributed exponentially: $F(T \leq t|s, a) = 1 - \exp(-\beta(s, a)t)$.

simplifications. Thus, it is unlikely we can ever calculate the optimal value function or the optimal policy exactly for all possible states.

With specific regard to transportation networks, [124] reviews the state of the art (as of 1997), describing time-dependent shortest path algorithms for a limited deterministic version of the problem with discrete costs that connects up time-slice replicas of the original network, not unlike a DBN [41]. A more recent survey [51] avoids the topic of time-dependent stochastic routing altogether.

Wellman et al [167, 105] consider path search in time-dependent networks with stochastic costs, where the criterion is expected arrival time. They show that standard priority-queue search remains optimal with respect to expected cost under mild conditions of stochastic consistency. Efficiency is helped by a stochastic dominance pruning rule [157] which is the natural generalization of comparison of deterministic path costs. Liu [104] also presents an anytime algorithm, which is achieved by planning with an increasingly fine discretization of time. Unfortunately, they fail to demonstrate practicality experimentally.

Also based on discrete time and discretized costs is the approach to SSP by [117]. They take a two-step approach, where they first reduce, in an offline phase, the set of possible paths to one target by determining which paths have a positive probability of being the fastest. Then the on-line phase has a smaller decision space to work with.

5.2.6 Practical and complexity issues

Most practical route search algorithms are variants of the A^* algorithm [8]. They need not search the entire state space and therefore often exhibit better efficiency than dynamic programming. In case of stochastic costs, the corresponding algorithm is called AO^* [121, 22, 64]. The AO^* algorithm searches decision trees with chance nodes, but relies on static, time-independent action costs. Various strategies can be used to bound the value function and prune the search space [85].

Anytime search

Because the state space can be large and practical planners have to operate within fixed time limits, it is important to consider anytime algorithms, that is algorithms that have a (possibly suboptimal) solution ready quickly and gradually improve it when allowed additional computation time. Variants of the A^* possessing the anytime (online) property were proposed in [63, 170] and recently [62]. The anytime property is achieved by adapting the weighted A^* algorithm [39]. An admissible heuristic is weighted heavily at first, allowing the algorithm to quickly find a suboptimal solution. As the heuristic weight is lowered, solutions converge to optimality.

Learning search

Much work exists on planning with derivatives of A^* for agents that must act in real-time in imperfectly known environments. Typically, the imperfectly known environment is modeled by keeping an estimate of the value function, initialized according to a heuristic [93, 90, 88, 132]. Some work recognizes the need for these algorithms to deal with stochasticity of action execution outcome [90]. Learning real time A^* ($LRTA^*$) represents an example where the agent might plan with a different goal than expected cost. Inspired by game-play search, they treat nature as an intelligent adversary and plan to minimize the worst-case cost. However,

they tend to emphasize the value-function learning aspect in environments where only very local observations are available—the agent must be present at x to learn something about $V(x)$. In our application, the state of the entire network can plausibly be communicated to the planner. This reduces the utility to this effort of the strengths that $LRTA^*$ and its offspring possess.

The D^* algorithm [89, 154] for dynamic environments achieves speedups by incrementally reusing information learned in solving previous replanning runs. It can be combined with anytime search ideas [102] to obtain an anytime algorithm [101].

5.2.7 Continuous state space

The space of traffic conditions S is continuous. Consequently, the optimal value (the optimal expected time) and the optimal policy may not have finite support. Compact parameterizations of corresponding value functions are likely to exist only under severe restrictions placed on the form of the transition model and model costs.

An interesting approach to continuous-state MDPs is to express the value function approximately as a linear combination of basis functions [95]. The weight space can be again continuous, with a simpler structure and/or a lower dimension [96]. The problem of finding the optimal value function is thus reduced to a large (in the number of constraints) linear program. While large linear programs are still intractable, techniques exist that can be used to quickly get an approximate solution, such as constraint sampling [40].

In transportation applications, the question of continuous state space is sometimes sidestepped by relying on the existence of two modes of traffic: free-flowing and jammed. This is the approach in [85], who solve an MDP where the link state has only two values – congested or not and each link state forms a detached Markov chain. The disadvantage is that they need to define a fixed state (speed) for each of the binary states. This may be difficult to substantiate. They also consider discrete time decisions. Their model lacks the generality of our approach where every link state is characterized by a vector of continuous variables.

5.3 THE TRAFFIC ROUTE NAVIGATION PROBLEM

The road network can be viewed as a directed graph (X, E), where X is the set of intersections and $E \subset X \times X$ is the set of graph edges, in our case the road segments. The navigation problem we consider is to find a minimal-expected time route from an origin o to target t. A route is a function $r_t : X \to X$ defining the successor location when the routing goal is location t. Thus a route differs from an MDP policy in that it only depends on the location component of the state. Since we will have a definite origin o and assume that a routing decision can always be deterministically executed, a route can be reduced to a simple sequence of locations $\langle o, x_1, \ldots, x_n, t \rangle$ through which the driver should pass.

5.3.1 SMDP formulation of the navigation problem

Let us now describe our navigation problem as a MDP, assuming for simplicity discrete time. First, the state space must contain the traffic state variable vector $\mathbf{S}$, which includes, at a minimum, the flow speeds at all locations. In addition to the traffic variables, we extend the state with the current position of the car we plan to route. The position is represented using variable X. X is assumed to be observed at any decision point. Not coincidentally, this is the same X as in the graph representation of the road network. As a result, the full state space of the Markov decision process is the cross-product of traffic states and a car location.

The actions to consider correspond to route choices at the current location. Formally, $A(x) = \{x' | (x, x') \in E\}$. The traffic situation component of the state does not limit the driver's choice. Therefore A is only a function of X and not $\mathbf{S}$.

Since the optimization criterion is time, it is not appropriate to discount the cost, so $\gamma = 1$. This would cause convergence problems in the infinite planning horizon, but navigation can be safely considered a finite-horizon problem.

So far the MDP formulation was straightforward, but the discrete-time assumption is crude as some road segments are much longer than others. Let us make time explicit by casting the navigation task as a Semi-Markov Decision problem (S-MDP) [70, 73, 131] and take advantage of some independencies specific to the application. The transition dynamics are fully described by the conditional density:

$$p(s', x', t | s, x, a) \tag{5.7}$$

where s and s' represent the current and the next traffic state, x and x' are the current and the next locations of the vehicle, a is the routing action taken at x. t is the time of the transition, that is, the time it takes to get from x to x' under traffic condition s.

Assuming that the travel time t is defined by the current[4] traffic conditions $\mathbf{s}$, the current location x and action taken at x, the density function $p(s', x', t | s, x, a)$ can be decomposed as

$$p(\mathbf{s}', x', t | \mathbf{s}, x, a) = p(t | \mathbf{s}, x, a) p(\mathbf{s}', x' | \mathbf{s}, x, a, t). \tag{5.8}$$

$p(t | x, \mathbf{s}, a)$

Let us analyze the first term of the product. The time to traverse a link only depends on the (speed component of the) traffic state s and not on the moment when the transition is made. For simplicity, the travel time t over a segment i is computed deterministically as $t = L_i / v_i(\mathbf{s})$, where L_i is the segment length (obtained from infrastructure data) and v_i is the component of the state $\mathbf{s}$ that represents speed at the i-th segment. This formulation is contingent on three assumptions. First, we assume the driver will realize the same speed as the overall traffic flow. Second, we are making the homogeneity assumption – that the speed is uniform along the segment. Lastly, the execution time should be short enough so that the speeds remain nearly constant.

[4] This assumption is warranted if action execution (link traversal) time is much shorter than the time scale on which significant state changes occur in the underlying process. For our traffic application, the median road segment traversal time is far below one minute and one can safely assume that the traffic state remains approximately constant during such a short period.

Now, let's look at the second term of Equation 5.8. A special feature of the navigation problem is that routing actions, the choices made by the driver of a single vehicle, do not affect overall traffic behavior.[5] Under this assumption, s' becomes independent of x, x' and a; and the distribution $p(\mathbf{s}', x'|\mathbf{s}, x, a, t)$ decomposes into $s' \perp x'$

$$p(\mathbf{s}', x'|\mathbf{s}, x, a, t) = p(s'|s, t)p(x'|x, a) \tag{5.9}$$

Further, it is safe to assume that the next location x' is a deterministic function of the current location x and the road segment a chosen. In other words, $p(x'|x, a)$ is a degenerate distribution with mass 1 at a single location x'. This assumption reflects the domain-specific fact that once we take a road a in location x we eventually reach the segment's end at x' and no other X-states in between. $p(x'|x, a)$

As a result of these independencies, the transition dynamics of the MDP factorizes as

$$p(s', x', t|s, x, a) = p(s'|s, t)\underbrace{p(x'|x, a)}_{determ.}\underbrace{p(t|s, x, a)}_{determ.} \tag{5.10}$$

Thus, the only component of the next state that is subject to random behavior is s'. If we consider planning in fixed-time increments, $p(s'|s)$ is sufficient to describe traffic state evolution. However in general, the next decision point at which s' is observed again depends on the traversal time of the link identified by x and $a \in A(x)$.

We need a good representation for $p(s'|s, t)$ to predict future traffic and solve the S-MDP. One might use a general traffic flow model such as METANET [113], frequently used to simulate and study traffic flows. The METANET model is a system of difference equations relating speed, volumes and buffering effects of traffic in an interconnected road network. However, the METANET model is deterministic and does not reflect possible variations among traffic variates (speeds, volumes) and variations among speeds of individual vehicles. A closer match is a probabilistic predictive system like JamBayes [69], which models traffic behavior using Bayesian network. This model is unfortunately unsuitable since it reduces the problem by focusing on pre-identified traffic bottlenecks only, while we consider the network in its full generality. Also, it is less appealing intuitively, since it optimizes a prediction metric only and employs structural learning. As a result, its learned Bayes network models have no common-sense interpretation.

We will compare a range of different representations of $p(s'|s, t)$ from straw men of increasing complexity to the dynamic model of Chapter 4.

[5] If such an algorithm is widely adopted and recommends to each driver the same route, interesting effects arise and the assumption is no longer valid. In fact, many envision that traffic management systems of the future will be centralized to a large degree, benefiting from the origin-destination information collected from most vehicles prior to their departure. While they might take the road utilization to a new level of optimality, systems so pervasive—and intrusive—will definitely not be implemented in the foreseeable future (and there are good reasons to hope for "never"). Therefore this work will not be concerned with such problems of collective decision making.

5.4 SAMPLESEARCH NETWORK PLANNING ALGORITHM

Planning problems are now usually formulated as a Markov decision process (MDP). However, the road network in a metropolitan area involves thousands of interconnected road components, each with traffic state measurements. This leads to an MDP with a high-dimensional state space. In addition, all quantities are continuous which makes the exact solution of the full MDP intractable. Therefore most shortest-path algorithms are search-based.

In this section, we propose SAMPLESEARCH, a modified A^* algorithm to operate on a continuous state space with intractable dynamics and stochasticity. We will us first address the intractability by means of approximating the Bellman update. Then, we discuss how to prune the search tree when costs are time-dependent.

We consider two settings, offline and online. In the offline setting, the agent simply executes the initial route computed at the start. This corresponds to a vehicle without communication equipment. In the online setting, at every decision point the current traffic condition may be observed and taken into account when selecting the next route choice. The travel must start at the time of planning.

5.4.1 Approximate Bellman updates

The solution (under expected-time) of the decision problem can be described by the Bellman equation [11]:

$$V(s,x) = \min_{a \in A(x)} E[t|s,x,a] + \int_{S'} p(s'|s,t)V(s',x^a) \tag{5.11}$$

where $V(s,x)$ is the optimal expected time for the route starting in location x at traffic condition s and x^a denotes the intersection reached from x by taking the road segment a. Briefly, the optimal expected time for the current traffic condition s and location x is obtained by optimizing the sum of the expected time for the first section of the road corresponding to the action choice a, and the optimal expected time for the remainder of the route. The expectation is taken over the distribution of probable next traffic state.

Equation 5.11 involves an intractable integral over a very complex state space with hundreds of state variables. In general, the integral in the Bellman equation can be estimated using Monte Carlo sampling:

$$V(s,x) = \min_{a \in A(x)} \frac{1}{n} \sum_{i=1}^{n} [t_i(x,s,a) + V(s'_i, x^a)], \tag{5.12}$$

where $t_i(s,x,a)$ is the i-th sample of execution time of a (segment traversal time). This is an instance of the general MCMC approach where expectation over p is replaced by an average of samples drawn from p. A sample of size n of the next state s' and travel time t is obtained from the stochastic dynamic model. Because the number of states is so large, we wish to do this dynamic programming update in a forward fashion. That implies that the next state value function does not yet exists and needs to be computed in the same manner. The same problem repeats itself on the next level, where $V(s'_i, x^a)$ for each sample i must be estimated. This leads to a sample-based decision tree, where the depth of the tree corresponds to the

number of segments traveled from the initial state. Assuming that the number of action choices at every decision step is b (branching) and samples of size n are used, the complexity of optimization for the tree of depth d is $O((bn)^d)$, exponential in the length of the route. For any but the smallest routing problems, such a method is still impractical and requires further approximation.

5.4.2 Time-dependent A^* pruning

We can directly extend A^* to the case with time-dependent deterministic costs by keeping the "total elapsed time" T in the search tree nodes together with the node g-value. In our case of planning to minimize time to target, T and the g-value are identical.

FIFO networks

A minor computational hurdle is that pruning of already expanded states in general cannot be used with arbitrary non-fixed costs. The traffic problem has the following monotonicity property. Assume a fixed route with the target location t and consider an arbitrary point x along the route. Then

$$t + cost(x \to y, t) < t' + cost(x \to y, t') \quad \text{for all} \quad t < t'. \tag{5.13}$$

In other words, if the agent reaches any intermediate node x earlier, he will not arrive later. The driver must act and cannot wait at an intersection. Networks with such a property are called FIFO networks. This property allows us to partially prune the search space with respect to the location component of the state.

Stochastic dominance pruning

The pruning rule is based on stochastic-dominance [157]. Let x be reached by the search for the second time. Let $P_1(t)$ be the probability distribution of time of reaching x through a parent p_1. Correspondingly, let $P_2(t)$ be the distribution conditioned on reaching x through p_2. Then, the search branch $p_2 \to x$ can be pruned iff the cumulative distribution P_1 dominates P_2:

$$\forall t : P_1(T \leq t) \geq P_2(T \leq t). \tag{5.14}$$

This conditions says: "Whatever time t (to get to x) you are trying to beat, your chances are better going through p_1. Together with the monotonicity property, this implies that the search branch reaching x through p_2 can be pruned.

5.4.3 Algorithm description

Putting these tricks together, we propose an algorithm called SampleSearch that can search in the space of the network graph (X-space) only, referring to a dynamic model for the full traffic state. To begin the description of SampleSearch, first note that all we need to store in the search tree is the time (cost) of each node z, which is the sum of edge costs sampled on the way to z. Having the predictive model handy, we can always obtain a sample from the predictive distribution given a time in the future. Thus, the search tree node only needs to store the location x and time t.

SampleSearch holds p samples $\{T_i^q\}_{q=1}^p$ in each node x_i in the form of a histogram. A histogram simply represents an empirical distribution as a set of pairs $\langle T^z, w^z \rangle$ such that $\sum_z w^z = 1$.

The nodes to be expanded are held in a priority queue ordered by f-function that consists of the expected time of reaching the node and an admissible heuristic $h(x_i)$:

$$f(x_i) = g(x_i) + h(x_i) = \left[\frac{1}{p}\sum_{q=1}^{p} T_i^q\right] + h(x_i).$$

We will discuss some heuristics below.

To expand a decision node $[T_i, x_i]$ with weight w, we take p samples $t^1, \ldots, t^p$ from the link traversal time distribution $p(t|\mathbf{s}(t_i), x_i, a_i)$. The p children $[T_{i+1}^z, x_{i+1}]$ are created with $T_{i+1}^z = T_i + t^z$ and weight $\frac{w}{p}$. Thus we have p^2 samples of time of arrival for each of the b successor nodes. We can now use the stochastic dominance test to determine if the successor node can be pruned from search.

However, we cannot keep p^2 samples per node. The first contribution to reducing this number is to collate the samples that resulted in the same total time of arrival at the successor. In practice, we will consolidate into one time all samples that differ less than the error of measurement, say 1 second. If there are still more than p samples left, we sample p survivors according to the associated weights and discard the rest to restore the invariant that each node (corresponding to a location) holds at most p samples. The process is illustrated in Figure 13.

Given this construction, the space complexity of SampleSearch is at most $O(\max(p|X|, p^2 b))$. The time complexity is determined by the size of the network space $|X|$ and quality of the heuristic. The worst case time complexity is $O(|X|p^2 b)$.

The SampleSearch algorithm is similar to the sparse sampling approach of Kearns, Mansour and Ng [79]. Their approach also creating a sampling tree from a generative model of the MDP (equivalent to or $p(s', x'|s, x, t)$) and evaluating this tree under expectimax to get the value function of the root of the tree. The differences are several. They place a zero value on the leaves of the tree, while we use a more informed heuristic. Accuracy of their estimate is guaranteed by discounting – in fact, any bounded value at the leaves would in principle work as well. Secondly, they cut the tree at a depth when the desired precision can be guaranteed by discounting. [79] is a general method that does not assume any state space structure and is designed for discrete-time MDPs. SampleSearch unfolds the tree over the X-subspace of the state space and implicitly stores a sample from the S-subspace in the search nodes. SampleSearch does not discount, searches all the way to the goal and instead collapses the population of samples to guarantee a constant per-step running time. The methods presented here do not come with performance guarantees such as [79], nor does it seem possible to give such guarantees without introducing discounting.

5.4.4 Online search

The setup so far called for formulating a policy at the start of the search and following it all the way to the goal. The commitment to a single route may lead to a suboptimal behavior.

To incorporate current network state into the decision making, we rely on the greedy online decision strategy. The basic greedy strategy expands the decision tree for just one step and the value of the optimal

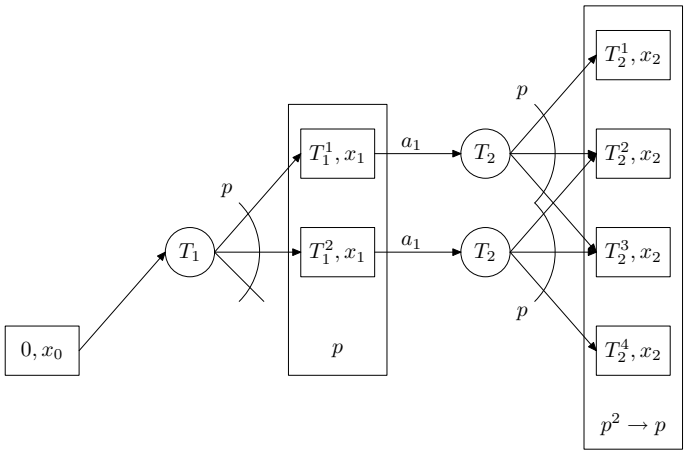


Figure 13: The sample-based tree roll-out of an MDP. Decision nodes are drawn as rectangles, while chance nodes are the circles. Note that the actions a are fixed in each decision step and the vehicle always gets to the same x_i reliably. The chance nodes represent the random amount of time that it will take to reach the next decision point. The states s_i^j represent the traffic situation when decision point i is reached at time t_j. We take n samples from the distribution of link travel time. On the next level, those decision paths j and ℓ that had $T_i^j + T_{i-1}^j = T_i^\ell + T_{i-1}^\ell$ will collapse together. Typically, more than p samples will still be left. In that case, we subsample the cumulative travel time distribution down to p samples to keep the representation constant in size.

strategy for the next step state is replaced with its approximation. The value for the current state is calculated as:

$$V(s,x) = \min_a \left[\frac{1}{n} \sum_{j=1}^{n} [t_j(s,x,a) + \hat{V}(s'_j, x_a)] \right],$$

where $\hat{V}$ is obtained via some more efficient approximation. For instance, $\hat{V}$ can be obtained by solving the shortest-path problem in the X-space, without considering traffic state. The difference in efficiency is that in Equation 5.12 we obtain $V(s'_j, x_a)$ by the same algorithm, here we fall back on a faster heuristic.

The optimal greedy decision is then

$$\pi(s,x) = \operatorname*{argmin}_a \left[\frac{1}{n} \sum_{j=1}^{n} [t_j(s,x,a) + \hat{V}(s'_j, x_a)] \right].$$

The d-greedy strategy expands d levels of the decision tree before relying on the approximation.

There are many choices for $\hat{V}$. For instance, solution to a relaxed shortest path problem with fixed costs without regard to traffic state qualifies as a possible approximation. It is more efficient to perform backwards search in this case, as the cost-to-go is immediately collected for every node reached and the result can be cached for subsequent calls to the heuristic.

5.5 THE K-ROBUST ALGORITHM

We will now introduce a new planning algorithm designed for planning in environments characterized by stochastic evolution and negligible influence of the agent on the environment. The algorithm tries to mirror a natural decision making process in a driver faced with several choices of route to her destination. The driver estimates the traffic environment she is likely to encounter and considers a route that appears reasonable with respect to that environment. Then she considers alternative states of the environment, especially those that would make her chosen route a bad choice. She winds up considering several candidate routes and evaluating them against possible future environments states. Finally, she selects the one that is "best" under whatever preference criterion exists implicitly in her mind.

Similarly, our algorithm considers a future trajectory of traffic conditions $\bar{\mathbf{s}} : T \to \mathbf{S}$, a function that associates with each each time a traffic flow state. Such a trajectory can be obtained from the model of traffic dynamics $p(\mathbf{s}'|\mathbf{s})$, starting in the initial traffic state, by successively sampling $s_1^{(j)}$ from $p(s_1^{(j)}|s_0^{(j)})$, $s_2^{(j)}$ from $p(s_2^{(j)}|s_1^{(j)})$, etc. using a fixed time period between $s_i^{(j)}$ and $s_{i+1}^{(j)}$. The traffic state at times between samples is approximated by linear interpolation between the two samples closest in time. The sequence needs to be long enough to cover the maximum possible time that execution of the plan can take. In our implementation, we interleave the dynamic model with planning. The planner requests the samples from the dynamic model, which uses lazy evaluation for efficiency.

We can then use the evolution of traffic states represented by $\overline{\mathbf{s}}$ to supply fixed cost functions for network links. Given $\overline{\mathbf{s}}$, a deterministic time-dependent search algorithm can be run very quickly by considering only the travel times (edge costs) based on $\overline{\mathbf{s}}$. The deterministic search produces a candidate route r.

An advantage of such a sampled-future shortest path approach is that it takes into account the initial traffic condition and that dependencies among speeds, volumes and travel times in neighboring road segments are reflected in the sample trajectory. But of course, the route is built for a just one state trajectory and this may not represent very well the population of trajectories that can arise from the initial state $\mathbf{s}_0$. To alleviate this issue we propose the k-robust strategy. The idea of the method is simple: we sample k state trajectories $\overline{\mathbf{s}}^{(1)}, \ldots, \overline{\mathbf{s}}^{(k)}$ according to the model of dynamics and for each sample $\overline{\mathbf{s}}^{(i)}$ we generate the shortest path $r^{(i)}$, obtaining k proposed routes to the target.

Each route is then evaluated with respect to a set of trajectories that it was *not* generated from. The route that optimizes the optimization criterion is selected. This is reminiscent of the PEGASUS approach to solving POMDPs [120], with the twist that we fix the future state trajectories to identify the family of policies (routes), while PEGASUS uses it for evaluation of routes only. Additionally, the route is a restricted kind of policy as it depends only on X (location) and not on S (traffic state) component of the MDP state.

The general scheme of the k-robust planner is in Figure 14. An advantage of the algorithm is that any criterion we may wish to optimize can be accommodated in the evaluation part. Thus, the applicability of the procedure is not limited to expected-time optimization.

There are three limitations to k-robust planning. First, it is only applicable in environments where the assumption of negligibility of agent actions holds. Second, it is not clear how to allow for future sensing actions. The agent is committed to one of the routes and cannot change it without full replanning. Consider for instance a decision problem where the agent arrives at some point to an intersection where the light is mostly green. Since the agent cannot condition his decision on observing the light, the most robust policy under many reasonable decision criteria is to drive through, which carries a high cost if the light is actually red. Third, as it considers deterministic future state trajectories, it is difficult to take variance of the predicted cost into account.

Let us now address the evaluation of a route in more detail. Assume a fixed route $\mathbf{x} = \langle x_1, x_2, \ldots, x_\ell \rangle$. The route also implicitly fixes the routing decisions a_i at each location x_i. We would like to obtain an estimate of $\mathbf{x}$'s expected time starting in state $\mathbf{s}_0$ as efficiently as possible. Under expected time, the cost of a route $\mathbf{x}$ is:

$$V^{\mathbf{x}}(s, x_i) = E[T|s, x_i, a(x_i)] + \int_S p(s'|s, t(x_i, a(x_i)))V^{\mathbf{x}}(s', x_{i+1})\mathrm{d}s' \tag{5.15}$$

Existence of a closed form of the equation is unlikely in all but very special cases. Similarly to direct route optimization, we have a complexity problem with the integrals. To avoid the problem of exponential growth we use the same trick again: We first sample m state trajectories $\{\overline{\mathbf{q}}^{(1)}, \cdots \overline{\mathbf{q}}^{(j)}, \cdots, \overline{\mathbf{q}}^{(m)}\}$. Then, we obtain an empirical distribution (histogram) of total travel times for each of the candidate paths $r^{(i)}$ by evaluating it with respect to $\overline{\mathbf{q}}^{(j)}$ for all j. At the end of the path, the histogram represents the cost distribution $p(t|\mathbf{x})$

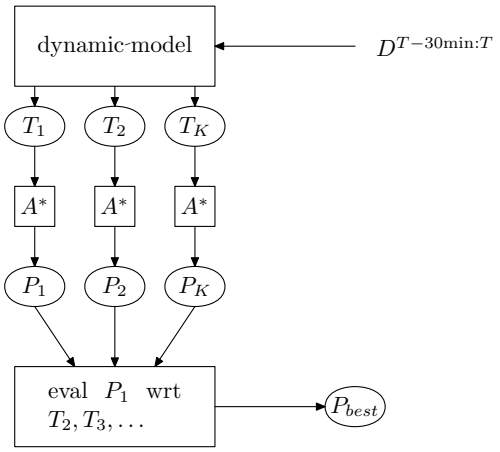


Figure 14: The block scheme of the k-robust planning algorithm

for the path $\mathbf{x}$ under state trajectories $\{\overline{\mathbf{q}}^{(j)}\}_{j=1}^{m}$. We obtain a representation of the distribution of time to execute the route $r^{(i)}$, consisting of m samples. Now, we can apply any utility function $u(t)$ to the outcomes represented by the particles of the histogram and obtain a "score" for the proposed path $\mathbf{x}$. In our example, we just take the expected value of t as this is the optimization criterion. The best-scoring path is the result of the k-robust algorithm.

Importantly, note that the complexity of the one path evaluation is only $O(k\ell)$. Therefore, the overall complexity of the k-robust technique is dominated by executing k instances of A^*.

5.5.1 An online version of k-robust

Upon seeing an observation, the straightforward online variant of k-robust planning would estimate values of the successor states by resetting the dynamic model to the observed successor traffic flow state and sampling the state trajectories from the new predictive distribution. Due to the similarity with beam search [136], we would call the online version the k-BEAM algorithm. However, it is disputable whether such an expensive operation that looks far ahead into the future is a good match to the near-sighted greedy algorithm whose principal motivation comes from performance constraints.

There is, however, a natural generalization of k-robust to the online setting that does not have to reset the dynamic model with every observation. When we sampled the k scenarios, we implicitly associated equal weights with them. There is a direct one-to-one correspondence between particles in the dynamic model and the trajectories we sampled. The particles must have equal weights in prediction since no observation is available. Thus a new algorithm that does not generate new scenarios during the online phase is possible.[6]

[6]Proposed by Dr. Gregory Cooper.

First, in the offline phase, we proceed as k-robust and obtain an initial plan. However, during the execution of the plan, we keep updating the dynamic model using the flow equations. When an observation is received, we reweight the particles and the associated state trajectories. Given a set of scenarios, we can easily compute the value $V_i(x)$ of being in any location-state x for each trajectory $\mathbf{s}_i \in \{\overline{\mathbf{s}}^{(1)}, \ldots, \overline{\mathbf{s}}^{(k)}\}$ by backward search since the time of arrival is known and costs are fixed. Anytime the driver is to make a driving choice in x, the re-planning is trivial – the driver should go to x^*:

$$x^* = \underset{(x,x')\in E}{\operatorname{argmin}}\, u(\{(w^{(i)}, V_i(x'))|(x,x') \in E\}) = \underset{(x,x')\in E}{\operatorname{argmin}} \sum_{i=1}^{k} w^{(i)} V_i(x'), \tag{5.16}$$

where we are minimizing because V represents cost-to-go. The weights $w^{(1)}, \ldots, w^{(k)}$ are those of the k particles in the model. The weights are of course proportional to the posterior probability of the scenario. u is the utility function we are applying; in this case we just expanded it to the expected value of successor states. Thus this algorithm involves k forward searches, k backward searches (to establish $V_k(x)$ whenever necessary) and running the dynamic model. We will call it the ONLINE k-ROBUST.

5.6 EXPERIMENTAL COMPARISON

We evaluate the proposed algorithms and baselines on the task of finding the minimum expected time route through Pittsburgh road network.

5.6.1 Experimental setup

The metropolitan traffic network we use in this evaluation has 15,975 road segments connecting 4,937 intersections extracted from the ArcGIS [www], commercial geographic information system.

For many origin-destination pairs, there is only one sensible route alternative. In other words, the optimal route is the same under any traffic situation. That is not very useful for comparison of planning methods. Therefore, several origin-destination pairs were chosen so that both ends lie near a major, instrumented road and so that at least two alternative routes between them exist that have (to the point of data precision) equal expected time of travel. Where the road is not instrumented, we use the posted speed limit as the estimate of travel speed.

The time at which the planning occurs varies with the choice of the origin-destination pair, but is always a fixed time of a *week*. We plan a route using each method 20 times, using the same time point in different weeks of 2004 as evaluation data and collect the running-time statistics. The generated route is then deterministically evaluated using the values actually measured at the time and location of the supposed route plan execution.

The training and testing sets are respectively represented by data from two different years, 2003 and 2004. The dynamic model parameters were estimated from 2003 data, as were all the parameters for the

baselines that use historical data. All algorithms are designed to effectively simulate an agent that plans her route in 2004. The agent's knowledge of 2004 data is limited to the instantaneous snapshot of the traffic situation at the moment of planning, and, in case of online planners, the situation updates during execution of the plan.

5.6.2 Evaluation metric

Depending on the origin-destination pair and the margin by which the optimal policy outperforms the second-to-optimal one, even wildly differing predictions of traffic flow state may lead to identical plans. Conversely, a small change in prediction can change the optimal policy. Therefore, as opposed to the measures used in Chapter 4, to evaluate quality of algorithms and model for predicting the edge costs, it is better to consider the actual cost of the eventually computed routes.

5.6.3 Baseline heuristics

The problem of time-dependent stochastic shortest path can be approximated if instead of travel time distributions we assume fixed travel times on the links. This has been the traditional approach [125, 124, 51]. Two reasonable transformations that can yield fixed travel time costs are the posted speed limit and the mean speed heuristics. MaxSpeed determines the travel time of a link from the speed limit and the length of the segment. MeanSpeed divides the length of the segment by the average speed observed on the road segment at the time-of-day when planning occurs. Use of an admissible heuristic ensures optimality of the search solution. The heuristic we use is straight-line-distance divided by 70 mph and is admissible since the straight-line distance is a lower bound on the actual distance and 70 mph upper-bounds the flow speed. The maximum speed limit in the region is 55mph and measured average speeds over 70mph are anomalies occurring at times of low traffic.

One limitation of the speed-limit and the mean-speed approximations is that they disregard the current traffic situation s_0 and its probable evolution. The simplest way to utilize this information is to take the current snapshot of traffic in the network at the time of planning, define the fixed link costs and run A^* with them. This Snapshot algorithm considers the current situation but not its future evolution and time dependencies that may exist among traffic variables.

5.6.4 Experimental results

The results are summarized in Tables 7 and 5.6.4. Let us comment on Table 7 first. In Table 5.6.4, we report results of experiments on additional sites with slightly different parameters.

Offline planning. The offline planning method observe the traffic state at the time of planning, commit to a route and execute it. MaxSpeed represents our lower performance bound, as it does not take advantage of any congestion information. The MeanSpeed algorithm that defines the edge traversal costs to be the

route	method →	MaxSpd	MeanSpd	SmpSearch	10-robust	Snapshot	Omniscient
$H \to M^{ab}$	cost (h)	0.499	**0.430**	**0.403**	**0.399**	**0.3790**	0.3788
17 : 00	± (h)	0.063	$0.057^{1.000}$	$0.060^{1.000}$	$0.059^{0.973}$	$0.051^{0.997}$	$0.049^{0.665}$
	CPU (s)	15.4	504	102	569	112	83.2
	± (s)	6.23	48.0	39.5	286	41.8	27.0

Table 7: Performance metrics for the "one-shot" planning algorithms on real network data. The algorithms are sorted by mean performance. Numbers in bold indicate a statistically significant ($\alpha = 0.95$) difference from the method on the left, using a paired t-test. Superscripts of standard deviations indicate the cumulative t-distribution. Values close to 1 are strongly significant. The path cost is in hours, the runtime in seconds. The runtime is strongly affected by computational issues and should be considered a measurement with only order-of-magnitude precision guarantees.

[a] H = Churchill ($-79°51'53.60'', 40°25'51.28''$)
[b] M = Millvale ($-79°59'31.60'', 40°29'26.29''$)

mean cost observed at the corresponding time-of-day fares significantly better. This underlines the value of historical traffic records in route planning.

SampleSearch opens a large gap between the previous methods which do not consider current traffic situation and the more informed planning methods.

The 10-robust method outperforms the SampleSearch by a narrow margin, which is nevertheless statistically significant in a paired t-test, as it almost always at least matches SampleSearch for path cost. 10-robust does that because one of the trajectories it considers is exactly that used in SampleSearch. [7]

The simple Snapshot algorithm performs surprisingly well in this real-world setting; we do not have evidence to reject the hypothesis that it actually matches the optimal solution. It appears that the momentary snapshot of traffic situation is a better predictor of traffic state in the approximately half-hour horizon of our evaluation than samples of previously realized state trajectories beginning in states most similar to s_0. In other words, the temporal autocorrelation of traffic state is tighter than the correlation of state trajectories with a common initial state.

The upper bound on performance is set by the (unrealizable) Omniscient method, which plans using edge traversal costs that the path would actually be evaluated on and is therefore always optimal.

The runtimes are given for orientation and should be compared on the orders of magnitude only as every method offers different implementation opportunities and tradeoffs, such as optimization and caching of database queries. Database performance is the bottleneck here; the search algorithms wait for the DB about 90% of the running time. Search tree expansion is highly parallelizable; a more sophisticated multi-threaded implementation balancing the DB load between several servers would surely achieve significant speedups.

[7] The slowdown from SampleSearch is less than 10-fold. This is an implementation artifact as the 10 successive evaluations are able to reuse cached DB query results.

route	costs: →	MeanSpeed	MaxSpeed	Online k-robust	Snapshot
$H \to M$	mean cost (h)	0.425	**0.392**	0.388	0.380 [a]
17 : 00	st dev (h)	0.075	$0.048^{0.981}$	$0.061^{0.702}$	$0.051^{0.919}$
	E[runtime] (s)	1373	66.2	1.76	8150
	st dev (s)	353	11.3	1.12	5500

Table 8: Performance metrics for the online planning algorithms using different edge cost predictors. The Online k-robust uses posted speed limit, the same as MaxSpeed. The legend is the same as in Table 7.

[a] Significant over MaxSpeed at 0.964

Online variants. Let us now consider the online-planning methods. In all cases, we used the greedy 1-step look-ahead algorithm. We report the results with three different choices for the heuristic $\hat{V}$, which is used to evaluate nodes beyond the first-level expansion of the decision tree. The result of a simpler search problem can be used as $\hat{V}$. In the first two cases, we use search with the MaxSpeed and MeanSpeed approximations of the link costs. In the third case, we use the Online k-robust algorithm from section 5.5.1. It performs quite well, beating the simpler heuristics, even though in the reported setup it uses the same edge cost distribution as MaxSpeed (the posted speed limit). Even if the statistical significance levels are low, the improvement points to the benefit of considering longer-term scenarios in planning. When Online k-robust uses the currently observed traffic state ("snapshot"), its decisions are identical to the next algorithm that does the same without considering the k scenarios. Lastly, we evaluate the well-performing Online Snapshot algorithm. Its performance is also statistically indistinguishable from the optimal solution and Snapshot by itself.

The MaxSpeed heuristic is independent of time and its value function can be pre-computed by straightforward dynamic programming prior to execution of the online search. Then the evaluation of $\hat{V}$ becomes a simple table lookup. This explains why the computational requirements are significantly lower for this method than for the Snapshot version. The Online k-robust is reported using the posted speed as cost estimator. Having it use the observed situation makes it indistinguishable from the Snapshot and Optimal solutions. Its time is reported per-step; however, there is a substantial off-line set-up time when the initial k-robust algorithm is executed and value functions are subsequently computed. Most of its per-step time is due to running the dynamic model, which in turn spends most of its time querying the traffic database. The planning time for Snapshot implementation is about 5 times the plan execution time, but as argued above, a higher-quality implementation could easily attain real-time performance. The other methods become prohibitively expensive to evaluate in the online setting, as replanning to occur at each step would have to reinitialize the dynamic model.

The improvements that we obtain in the real application are comparable to other studies that considered leveraging historical and online data in similar settings, e.g. [8]. Much higher gains are probably impossible. This is because traffic networks are highly optimized in their design process and do not offer many opportu-

route	method →	μSPEED	SMPLSEARCH	5-robust	20-robust	SNAPSHOT	OMNISCIENT
$P \to C$[a]	mean time (h)	0.558	**0.494**	**0.487**	0.481	**0.458**	**0.445**
06 : 30	st dev (h)	0.008	$0.029^{1.000}$	$0.030^{0.974}$	$0.027^{0.9237}$	$0.028^{1.000}$	$0.011^{0.997}$
$C \to P$[b]	mean time (h)	0.558	**0.479**	**0.470**	**0.466**	**0.442**	0.440
06 : 30	st dev (h)	0.007	$0.012^{1.000}$	$0.016^{0.999}$	$0.017^{0.985}$	$0.011^{1.000}$	$0.010^{0.923}$
$D \to R$[c]	mean time (h)	0.401	**0.348**	**0.334**	**0.324**	**0.279**	0.279
17 : 00	st dev (h)	0.042	$0.013^{1.000}$	$0.028^{0.996}$	$0.025^{0.996}$	$0.012^{1.000}$	0.012^{NA}
$D \to C$[d]	mean time (h)	0.215	**0.167**	0.160	0.157[e]	**0.150**	0.150
17 : 00	st dev (h)	0.003	$0.017^{0.989}$	$0.008^{0.891}$	$0.004^{0.909}$	$0.009^{0.951}$	0.009^{NA}

Table 9: Travel times between origin-destination pairs; and the time of day of route execution. Mean and standard deviation of travel time are taken over 20 plan executions. Boldface indicates the entry is significantly better than the method to the its left under a paired t-test at $\alpha = 0.95$.
Superscripts of standard deviations indicate the cumulative t-distribution. Values close to 1 are strongly significant. Where NA appears, the methods performed identically.

[a] C – Carnegie ($-80°4'59.54'', 40°24'27.32''$)
[b] P – Penn Hills ($-79°50'53.52'', 40°28'42.24''$)
[c] R – Rodi Road ($-79°49'49.44'', 40°26'17.52''$)
[d] D – Downtown ($-79°4'47.04'', 40°26'23.28''$)
[e] the difference is significant over SAMPLESEARCH

nities for taking sensible alternate routes in face of periodic (not caused by incident) congestion. Moreover, any strictly preferable routes would quickly come to bear increased usage degrading their level of service to a Pareto equilibrium where no participant can improve his travel time by switching to the preferebale route.

Let us now discuss further experimental results from Table 5.6.4. We followed a similar setup, extending the evaluation to more origin-destination pairs. Also, different k-variants of the k-robust versions are used to see the direction of k's effect on performance. The clearly inferior MAXSPEED heuristic was dropped from the evaluation.

In all cases, using the dynamic model to predict travel time and planning with that prediction yields significantly shorter travel time (as much as 5 minutes on a half-hour trip) than the baseline approach, which uses observed historical average speed to determine travel time.

The proposed k-robust algorithm also results in an improvement, albeit smaller, for $k = 5$. A further improvement is observed for $k = 20$. This suggests that inter-temporal correlations of travel cost indeed play a role, even though a weaker one compared to the importance of predicting the evolution of traffic better on average.

While a gap remains to the (unachievable) optimal solution, we have shown that vehicular traffic prediction can be exploited in order to plan faster routes. There is of course a catch for practical applications: we have assumed that the driver's actions do not influence the evolution of traffic. The predictions are no longer valid if a non-negligible fraction of the traffic network users execute plans based on such predictions. This is a much more complex topic that we deliberately avoided.

5.6.5 Volatility experiment

Surprised by the near-optimal performance of the SNAPSHOT planning algorithm on the road network data, we investigate what the conditions need to look like so that the expected benefit of k-robust is observed. To this end, we built a simple synthetic network model where we have more control of the traffic behavior.

5.6.5.1 Link failure model We basically run the model from Chapter 4, but induce "blockades" randomly by zeroing the outflow from a segment to $y_i = 0$. Speeds on the blocked segment quickly drop near 0. Each segment's blocked state is controlled by a Markov chain with transition matrix

$$\begin{pmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{pmatrix} \quad \text{with} \quad p_{00} > p_{01} > p_{10} > p_{11},$$

where p_{00} is the probability that the road stays unblocked in the simulation step, p_{01} is the probability that a free-flowing road jams. Obviously, it must be that $p_{00} = 1 - p_{01}$. Analogously, p_{10} and p_{11} control the behavior of a jammed segment. This is a fairly plausible model of incident dynamics. By controlling the values of $p_{..}$, we can control the predictability of traffic conditions, frequency of simulated jam occurrence and their duration: the larger the diagonal elements, the more predictable the traffic state is. A similar approach has been used by Kim et al [85, 84].

This model implies exponential durations for both the free-flowing and jammed states, but we do not consider this a serious limitation for the purpose of demonstrating effectiveness of our algorithm in a volatile environment.

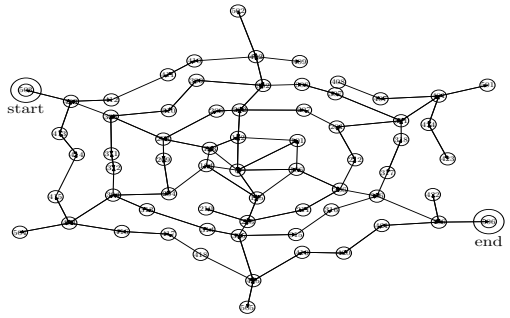
5.6.5.2 Evaluation setup The simulated traffic network with 61 nodes and 144 links is shown in Table 10. The flow–distribution parameters θ_{ij} were chosen so that the volume distributes evenly into successor links: $\theta_{ij} = 1/|Succ(i)|$. The network state was initialized in a randomly generated state and allowed to burn-in for a significant time before sampling. We chose sampling time $\delta t = 0.01$ for the traffic state model. p_{01} is set up so that the time to traverse a link is about the expected duration of both congested and uncongested state, which is about 1 unit of model time. This is a very volatile environment. The volume-speed relationship is modeled by a fundamental diagram shown in Figure 9a.

The start and end states are fixed throughout the evaluation. The experiment consists of 20 independent rounds, each with a different initial traffic state. The dynamic model maps one graph edge to one segment. For each initial traffic state a set of test traffic-state trajectories is generated according to the dynamic model. These trajectories are used to evaluate the performance of each routing algorithm under the conditions defined by the testing state trajectory.

5.6.5.3 Evaluation results The results of our experiments are tabulated in Table 10.

Offline planners All planning methods compare well against the MAXSPEED method that does not take advantage of the traffic information and uses the simple Dijkstra's shortest path algorithm to obtain

Method	Cost ± stdev	Runtime
MaxSpeed	7.02 ± 0.635	0 ± 0
Snapshot	7.09 ± 1.14	1.6 ± 4.92
MeanSpeed	6.85 ± 0.662	2.35 ± 5.74
k-robust	6.30 ± 0.646	30.6 ± 11.8
Greedy + Snapshot	6.47 ± 0.531	0 ± 0
Greedy + MeanSpd	6.38 ± 0.443	0 ± 0
K-Beam greedy	6.28 ± 0.693	9.7 ± 2.51
Omniscient	6.00 ± 0.331	0.8 ± 3.58



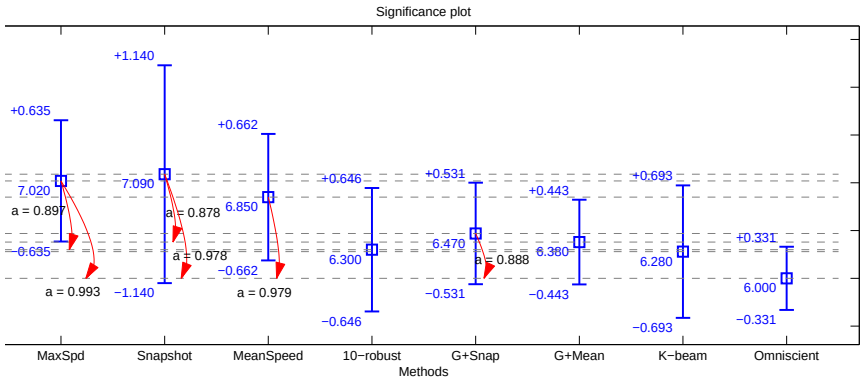


Table 10: Results on the simulated traffic road network, finding path from *start* to *end*. (opposing peripheral positions). The cost (travel time) is in minutes. Running time is in milliseconds. For the fixed strategies, the time is that of the execution of the algorithm; for the information feedback strategies, it is the average time to reach a decision at an intersection. Where time measurements are 0, the method was too fast to measure time accurately on the millisecond scale. A unpaired t-test significance plot is included.

the route. MAXSPEED fails to take note of frequent congestion patterns occurring on some roads and leads the agent onto congested roads. The SNAPSHOT algorithm freezes the initial traffic situation to build a plan. It is plagued by failures to account for change in the traffic state and in our experiments performed even slightly worse than the MAXSPEED method, because of the amount of volatility in the network. MEANSPEED method is better, but its performance, based on average traffic flow behavior, is limited by the variance of actual traffic costs. The k-ROBUST method, emerges a winner among the fixed strategies because it is able to predict future possible contingencies better. It ran with $k = 10$ planning and $M = 20$ internal evaluation trajectories. The OMNISCIENT method represents the unattainable lower bound on path cost by planning with foreknowledge of the actual testing situation.

In terms of runtime, the MAXSPEED algorithm was the best, as expected. MAXSPEED has no need to access the current state data. The SNAPSHOT method only needs to retrieve a single vector of speeds, while MEANSPEED is slowed down by trajectory sampling and the mean computations. k-ROBUST spends over 10 times more computation as it has to both plan 10 paths and evaluate them.

Greedy 1-step look-ahead The greedy methods are tested with a 1-step look-ahead. As expected, the performance of the greedy look-ahead methods depends on the heuristic $\hat{V}$. The online search equipped with the MEANSPEED search heuristic had better estimates of the true value than the search using the snapshot heuristic. That mean is a better predictor than current state suggests there is anticorrelation between traffic states on the time scales comparable with the route execution time. This parallels the standalone performance difference between the two methods embedded into the greedy policy as heuristics.

Both variants achieved negligible average time to make a decision. This is achieved by caching of heuristic evaluation results – if a state is reached in heuristic evaluation that is identical to a state seen in previous searches, the evaluation is terminated and the previous result is returned.

K-BEAM-GREEDY is the straightforward online version of the k-robust method, feasible in this small network. The performance of the K-BEAM-GREEDY is similar to that of k-ROBUST method in cost. A slight improvement is observed due to the availability of additional information during plan execution, allowing for course corrections. However, the method appears to be driven heavily by the underlying cost estimator. In terms of planning time, the method is faster than k-ROBUST, because it only evaluates the resulting paths if they disagree on the first step and the cost of running the dynamic model is negligible in the toy example.

5.7 CONCLUSIONS

We have experimented with several strategies for route planning with stochastic, time-dependent costs in transportation networks. The results show that exploiting real-time congestion data in the route planning task can reduce the expected travel time by as much as 20 percent.

Two new planning algorithms were proposed. Both take stochasticity into account, but then actually

search in the network graph, which lends them efficiency. Both algorithms' intermediate result in the last step are empirical distributions of travel times over a route. This is more complete and desirable information than just a point estimate of the same.

The SampleSearch algorithm approximates the full expansion according to the Bellman equation with sampling from the transition kernel and its search proceeds through the spatial X-component and the traffic S-component of the state space simultaneously. As such, it is better suited for modification into an online version.

The second algorithm is k-robust planning, which proceeds through S-space first. It relies on the intuition that while traffic evolution is stochastic, there is a relatively small number of possible general scenarios (modes) that differ significantly in the pattern of edge traversal costs. It samples these scenarios and selects routes that perform most robustly with respect to the set of possible futures. Both proposed algorithms outperformed uninformed methods in route quality. However, a simple informed method is very close to optimal in this traffic network and we could not demonstrate superiority of our algorithms in this setting. We do have evidence that the k-robust algorithm performs well in very volatile environments. Clearly, a better quantification of how much unpredictability it takes for it to win is desirable. However, how to measure the "volatility" of an environment remains an open problem.

This chapter focused on the expected time to destination as the metric of route quality. Other metrics may reflect actual user preferences better. In fact, there appears to be a premium on low variance and predictability of travel time [49]. The sample-based decision tree and k-robust planning algorithm can easily accommodate a wide range of optimization criteria, such as those that take variance into account. For instance, minimizing the probability of arriving late is a reasonable criterion for which we have no good algorithms in general situations with covarying costs of actions.

6.0 LEARNING TO DETECT INCIDENTS

Summary. *Many deployed traffic incident detection systems employ algorithms that require significant manual tuning. I demonstrate here that machine learning incident detection solutions can reduce the need for manual adjustments by taking advantage of massive databases of traffic sensor network measurements. First, I examine which features might be most suitable for the construction of a learnable detection algorithm. Next, I show that a rather straightforward supervised learner based on the SVM model outperforms the fixed detection model employed in the state-of-the-art traffic incident detectors. Lastly, we seek further detection performance improvements by correcting misaligned incident times in the training data. The noise in incident labels arises from an imperfect incident logging procedure. I propose a label realignment model based on a dynamic Bayesian network to re-estimate the correct position (time) of the incident in the data. Experiments show that the detector trained on the automatically realigned data consistently tops the detection performance of the model derived from the original data while maintaining a low false positive rate.*

6.1 INTRODUCTION

Operation of real-world infrastructure systems is often interrupted with events that interfere with their expected function. Prompt detection of these events is critical for recovery of the system and for mitigation of further costs associated with the incidents. With advances in sensor instrumentation, many of these systems can now be better monitored and occurrence of the incidents automatically detected. Our work focuses on methods for early and accurate detection of road traffic incidents.

The most widely deployed traffic incident detection models are fixed-structure models that combine and threshold a set of signals such as volumes, speed and speed derivatives [107]. Tuning of these thresholds requires extensive involvement of traffic experts. What is worse, as the settings extracted for one site typically do not transfer to a new site, the tuning costs are multiplied by the number of sites in the network. Models that can be automatically tuned from data can reduce or eliminate costly recalibrations and improve detection performance [44, 134, 133, 127]. Inter-site transferability of detection algorithms is a central concern in automatic incident detection [155]. Machine learning offers a recipe for transferable detection algorithms: re-learn the detector on data from the new site, possibly borrowing power from the old site in form of prior

knowledge and we are presenting just such a solution.

If incident labels are available, the detection models can be trained in supervised mode and the detection task is one of classification. Alternatively, incident detection can operate by detecting anomalies in the behavior of the traffic system. In such a case, incident labels are not needed and any anomalies are assumed to be incidents. However, not every anomaly is in fact an incident. This mismatch leads to reduction of detection accuracy. Thus it is reasonable to expect supervised approaches to outperform unsupervised anomaly detection. Since we do have access to incident logs, we adopt the supervised learning view.

First, we gauge the informativeness of particular traffic flow features by constructing simple detectors using a single feature at a time. Then, we attempt to combine features in a probabilistic detector. A dynamic naive Bayes detector seems to capture the known phases of congestion onset and clearance, but the resulting detection performance is disappointing. We are much more successful with a SVM-based detection and show it easily outperforms the optimally calibrated standard model (the "California 2" algorithm [128]).

Real-world data are often subject to bias and noise coming from many different sources. Acknowledging and explicitly modeling the biases may translate to further detection improvements. In our case, we have noticed that incident logs are imperfect; the initial time of incidents is logged with a variable delay. Consequently, the incident label may be misaligned with the onset of the observed changes in traffic flow caused by the incident. Training a learner with such temporally misaligned class labels leads to suboptimal detector performance.

To work around the incident logging problem, we propose a new dynamic Bayesian network [41] that models the misalignment problem. It relates traffic flow quantities and the observed label position to the actual time of incident manifestation in data. We train the model on the manually realigned data from a *single* highway segment. Once learned, the model can be transferred to other highway segments to correct the incident labeling — a preprocessing stage. The transfer procedure is a variant of EM, starting with priors obtained from the hand-aligned data and combining them with data from the target site to adapt to the new location. The realignment model generates new incident labels in temporal data that are then used to train a supervised classifier such as a SVM to obtain the detection algorithm. This approach allows us to learn, with a *limited* amount of human effort, a more reliable detection model. We demonstrate the improvement in detector quality on traffic flow and incident data collected in the Pittsburgh highway network.

6.2 STATE OF THE ART

Incident detection has received intense attention of civil engineers. Several families of incident detection algorithms have been developed [107]. The California class of algorithms [128] are essentially simple decision trees, where the tests are typically of the form $traffic\_quantity \leq T_i$. T_i is a threshold whose value is a parameter that needs to be calibrated.

Practically any general anomaly detection method can be applied to incident detection. A simple T-test for increase in mean of 10 recent intervals as opposed to 10 preceding intervals was used successfully in [5] to detect with sparse detector stations. In double exponential smoothing [21], the first exponential smoothing generates an intermediate sequence which is the input of a second smoothing equation. A jump in the doubly smoothed sequence is indicative of an incident. Moving averages are another general model used to detect sudden deviations from mean. They have been applied to incident detection by [156].

The McMaster algorithm [129] is a single-station algorithm which fits a quadratic polynomial decision boundary in the volume-occupancy space. It works well for extraurban roads that do not experience frequent congestion and no seasonal changes in the fundamental diagram [166]. However, it suffers from high false positive rates in urban environments.

Time-series algorithms such as ARIMA [18] have been used to signal an incident when an observed value lies in the tail of the predictive distribution given previous k values [4]. They have mostly fallen out of favor because they tend to underestimate the variance of traffic measurements which leads to unacceptable false alarm rates [107]. Wavelet transforms are another time series tool to detect incidents [78].

Previous approaches to *learning* incident detection from loop detector data used almost exclusively neural networks [44, 134]. Recently, [169] presented a straightforward Bayesian network approach with good results, but strong likelihood of overfitting caused by the way they handled their very small set of incidents.

Frequently, a detector working well in the lab simulation settings disappoints in the actual deployment [127]. The detectors were often tested on data sets that might not be properly calibrated. That is, the proportion of incident instances in test set was much higher than could be expected in real deployment. This can severely skew the evaluation metrics. To avoid the pitfall, our testing procedure carefully preserves data calibration.

Another widely traveled computational avenue of incident detection is processing of video from traffic cameras [114]. This body of work is less relevant to our purposes as we do not have video surveillance data.

6.3 SUPERVISED LEARNING FOR INCIDENT DETECTION

Major roads in the metropolitan network are monitored by a sensor network that generates a stream of data about the vehicular traffic. The detection task is to continuously observe the data stream and raise an alarm when the readings seem to indicate an incident[1].

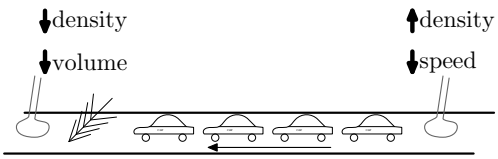


Figure 15: How a pair of sensors registers an incident (e.g. a downed tree). The two loop sensors detect the current induced in a wire loop by a passing vehicle, aggregate the counts and send them to the traffic management centre.

6.3.1 Traffic incident data

Incidents that the metropolitan Traffic Management Center (TMC) was aware of are noted in the data: their approximate location, time of accident and time of clearing by emergency responders (Figure 16). Short free-text accident descriptions are also available.

How does an incident manifest in the data? Figure 15 sketches the physical situation. The incident restricts the capacity of the roadway by blocking one or more lanes, forcing drivers to slow down to navigate around it. This will result at a temporary drop in the number and density of vehicles passing the downstream sensor. Upstream of the accident, a jam forms. When the tail end of the jam approaches the nearest sensor, it will cause a speed drop and vehicle density increase. The time when the sensors detect the anomaly depends on the utilization of the highway, distance to the sensors and severity of the incident.

6.3.2 Performance metrics

In the following text, when we say "detector", we mean the detection algorithm; "sensors" refer to the physical devices.

A false alarm occurs when the system raises an alarm, but no accident is present. The false alarm rate (FAR) is the number of false alarms divided by the number of detector invocations; thus a lower FAR is more desirable. The detection rate (DR) is the number of accidents actually detected divided by the number of accidents that occurred; thus a higher DR is more desirable. Traditional receiver operating characteristic (ROC) curves [130] are a metric designed to quantify detection of one-off binary events. Because accidents affect the traffic for a longer span of time and the detections are not equally valuable around the beginning and the end of the span, we instead prefer the activity monitor operating characteristic (AMOC) curve as the primary performance metric. AMOC curves are used for evaluation of rare event detection performance, such as detection of disease outbreaks [35]. AMOC curves relate false alarm rate (FAR) to time-to-detection (TTD). TTD is defined here as the difference between the time of the start of the first data interval that

[1] The term *incident* includes vehicular accidents as well as unscheduled emergency roadwork, debris on the road and many other hazards. Most incidents are accidents and we will use the terms interchangeably.

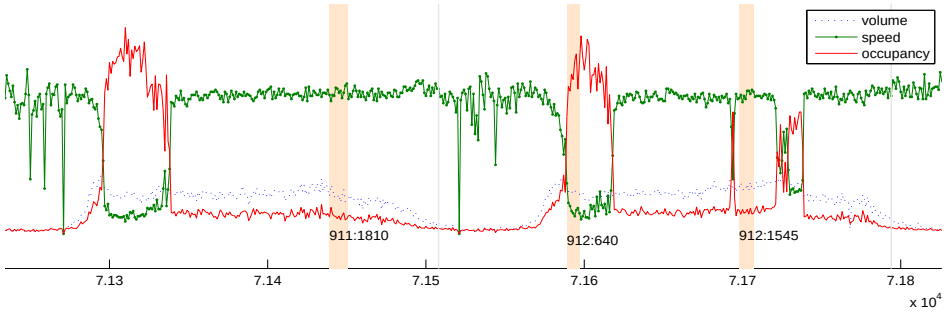


Figure 16: A section of the raw data. The red (solid in black and white printouts), green (solid with markers) and blue (dotted) lines represent average occupancy, average speed and total volume, respectively. Time is on the horizontal axis. The vertical (orange) stripes represent the reported accidents durations. A thin grey vertical line is drawn at midnight of each day. The numbers at the bottom encode accident time as recorded by TMC. Some accidents square with congestions perfectly (912:640 – September 12^{th}, 6:40am), some are slightly shifted (912:1545) and some even have no observable effect on traffic (911:1810). The incident at 912:640 is mostly obscured by morning peak traffic – compare to the morning traffic on the previous day.

was labeled as "accident" and the reported incident time. Note that this number can be negative because of the delayed incident recording. As we cannot guarantee to detect all accidents, we introduce a two-hour time-to-detection limit for accidents that remain undetected. When a scalar metric is desired, we compare detectors on $AUC_{1\%}$, the area under the curve restricted to the $(0, 0.01)$ sub-interval of FAR. This is a better indicator of detector performance in the usable low-false-positive region than the straightforwardly defined AUC.

The target performance at which a system is considered useful depends chiefly on its users. A study [133] surveying traffic managers found that they would seriously consider using an algorithm that achieves a DR over 88% and FAR under 2%. For any rare event detection system, a low FAR is absolutely essential. A system with a high FAR subjects its users to "alarm fatigue," causing them to ignore it.

6.3.3 The California 2 model

The algorithm known as "California #2" is a popular benchmark against which new detection algorithms are often compared [155]. California #2 (abbreviated CA2) is a fixed-structure decision tree model that proceeds as follows:

- Let $Occ(s_{up})$ denote occupancy at the upstream sensor s_{up} and $Occ(s_{down})$ the same at the downstream sensor. If $Occ(s_{up}) - Occ(s_{down}) > T_1$, proceed to the next step.
- If $(Occ(s_{up}) - Occ(s_{down}))/Occ(s_{up}) > T_2$, proceed to the next step. The rationale behind this step is

while a capacity-reducing accident will always produce large absolute differences in occupancy, these may also be produced under almost stalled traffic conditions.

- If $(Occ(s_{up}) - Occ(s_{down}))/Occ(s_{down}) > T_3$, wait until the next reading. If T_3 is still exceeded, flag an alarm. The wait is introduced to cut down on false alarms.

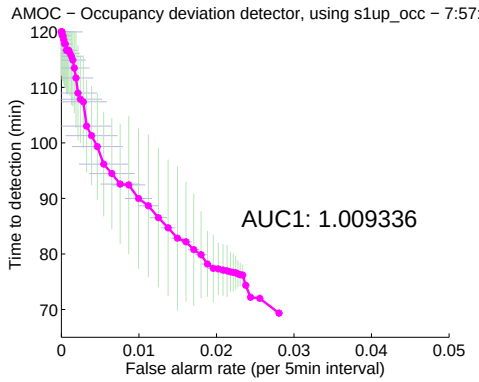
Thresholds T_1, T_2, T_3 need to be calibrated manually for each sensor site. The CA2 algorithm is normally tuned by experts who choose the three thresholds. We did not have access to such expert. However, we do have plenty of data to help eliminate the expert labor. Therefore we found the best parameterization by an exhaustive procedure trying all possible settings of the three parameters on a discrete grid covering a wide range of parameter values. The "best performance" for the purpose of parameterization was defined as the best DR at a fixed FAR of 1%. This was an extremely time-consuming procedure that is impractical for a metropolitan network with hundreds of sensors and changing traffic patterns, not to mention an uninteresting one from the learning perspective.

6.3.4 Evaluation framework

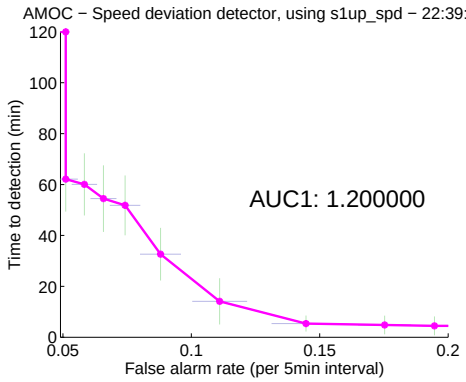
We evaluate the models under the cross-validation framework. Some adaptations of the standard train/test split concept were necessary to accommodate the specifics of traffic data. The dataset consists of three long sequences per sensor, one for each of the three traffic variates. We divide the data into train/test splits by incidents, making sure an entire incident sequence makes it into one and only one of the sets. To create the training set, we first select I_{train} "incident" sequences of preset length L so that the reported time of the incident falls in the middle of the incident sequence. C "control" sequences without an incident are selected so that no incident is recorded within additional $L/2$ data points before and after the control sequence. This safeguards against the imprecise accident recording. By choosing I_{train} and C, the class prior in the training set can be biased towards incident occurrences. The testing set consists of the $I_{test} = I_{all} - I_{train}$ incident sequences that were not selected for the training set. Additional sequences without accidents are added so that the testing set has class prior equal to that in the entire dataset.

To obtain the experimental statistics, we use 10 different train/test splits using the above method, with $I_{train} : I_{test} \approx 70 : 30$, sequence length $L = 100$ and $C = 50$ for training. For testing, instead of choosing a fixed C, we make sure the proportion of positive (incident) instance approximates the proportion observed in the entire dataset. All statistics reported are averages and standard deviations across these 10 splits, even where error bars were dropped for sake of readability. Error bars in all graphs in this section represent one standard deviation.

In each cross-validation fold, only the positive (incident) training sequences are realigned and the quality of the detection is evaluated on a withheld testing set, *using the original incident labeling*. While testing on the original labeling will result in a measurement that is somewhat off, the skew will be consistent across detectors and relative comparisons remain valid. If we evaluated on the realigned data, we would run the



(a) Upstream occupancy threshold detector



(b) Upstream speed threshold detector

Figure 17: AMOC curves for two simple detectors: $\mathbf{Occ}(s_{up}, t_0)$ and $\mathbf{Spd}(s_{up}, t_0)$. The speed detector has no practical value here, but the occupancy detector begins to detect incidents even with a very low sensitivity setting, translating to low false positive rates. The curve is plotted by varying the threshold from the minimal to the maximal value of the reading found in data. For each threshold, TTD and FAR are noted. This evaluation is repeated with 10 random sub-samplings of the incident database. The curve points and the bars plotted are the average and standard deviation respectively.

risk of having both the realignment algorithm and the detector make the same mistake in lockstep, losing touch with the data.

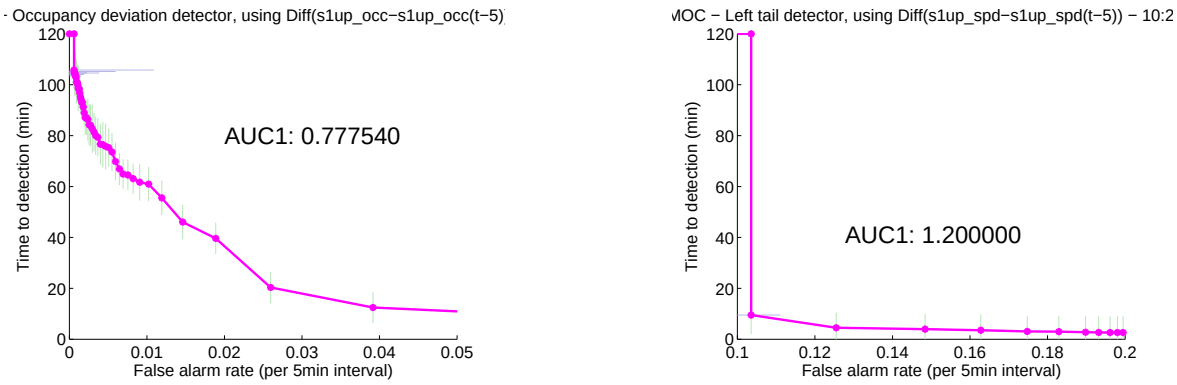
6.3.5 Feature selection

The CA2 algorithm uses a surprisingly limited set of features. Could a better detection performance be achieved if the detector takes advantage of multiple features? And which features?

The information value of data-derived features depends strongly on the temporal aggregation of the available data. In this section, we examine the detection power of simple distribution-tail detectors on our data, which has five-minute aggregation.

In Figure 17 we see the performance of the most basic detectors. Detector $\mathbf{Occ}(s_{up}, t_0)$ detects an accident whenever the occupancy reading at the upstream sensor s_{up} exceeds a threshold. The second argument in parentheses, t_0, is to denote that the algorithm uses measurement most recent at the time of the detector invocation. Detector $\mathbf{Spd}(s_{up}, t_0)$ outputs 1 if the speed falls below a detection threshold. While occupancy is clearly a more powerful feature, speed measurements can, as we show later, contribute important information in combination with other features.

A fundamental expectation for incident detection is that an incident changes the traffic flow abruptly. The abruptness should be best captured by features that are temporal derivatives of the sensor measurements.



(a) Occupancy spike detector

(b) Speed dip detector

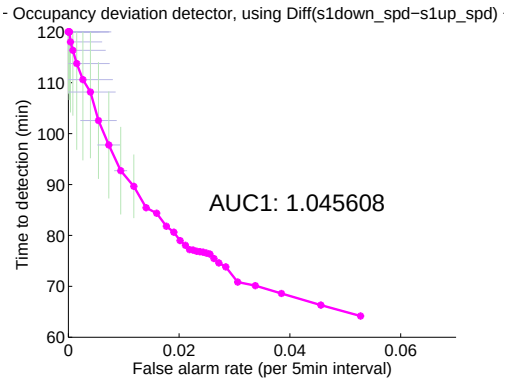
Figure 18: AMOC curves of temporal derivative detectors **a)** occcupancy spike detector $\mathbf{Occ}(s_{up}, t_0 - t_1)$, **b)** speed dip detector $\mathbf{Spd}(s_{up}, t_0 - t_1)$. Threshold varied from μ to $\mu + 5\sigma$. Again, the speed detector is not very useful.

The performance of detectors based solely on thresholding temporal derivatives of speed and occupancy is graphed out in Figure 18. Again, occupancy derivative seems to be the more informative feature.

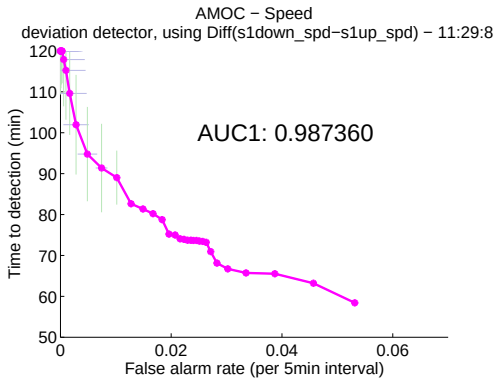
The intuition behind including, as an input to a detector, the reading of the neighboring sensor is that accidents and benign congestions can be distinguished by the discontinuity of flow characteristics between the upstream and downstream sensors. When an accident constricts the roadway capacity, we observe a congestion upstream of the accident. Unlike a benign congestion, an accident should cause a *drop* in the downstream sensor volume and occupancy measurements. The power of the difference detectors alone is shown in Figure 19.

The diagrams above inform us about the value of a feature for incident detection. However, the simple thresholding fails to take into account that measurements may vary in time with normal traffic and no incident involved. We may assume that such variation is smooth under normal condition. Thus a natural learning approach would be a form of t-test. Suppose we need to decide whether to raise an alarm on measurement x^t given previous N measurements $x^{t-1}, \ldots, x^{t-N}$. Under the assumption of normal distribution, we can check whether x^t lies in the (right or left, depending on the feature) tail of the maximum-likelihood normal distribution given the observations $x^{t-1}, \ldots, x^{t-N}$. We call this the t-test detector. Even better, we could attempt to follow the trend with linear regression. The linear regression detector learns from the N most recent data points and predicts the value x^t. Again, the value is anomalous if it lies in a tail of the predictive distribution.

See Figure 20 for the corresponding graphs. The lack of performance boost with a learning method can be explained by the fact that the difference features are already designed to capture any abrupt jumps and

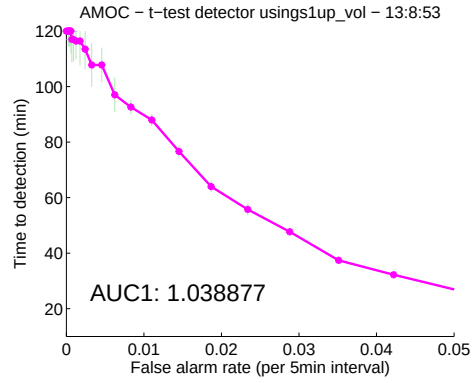


(a) Occupancy spatial discontinuity detector

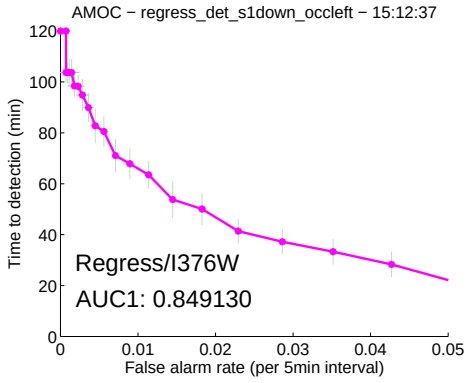


(b) Speed spatial discontinuity detector

Figure 19: AMOC curves for two spatial difference detectors: **a) Occ**$(s_{up} - s_{down}, t_0)$ and **b) Spd**$(s_{up} - s_{down}, t_0)$ Threshold is varied in the range $(\mu, \mu + 5\sigma)$. In contrast to the two previous cases, the spatial difference in speed is even more indicative of an incident than a difference in occupancies

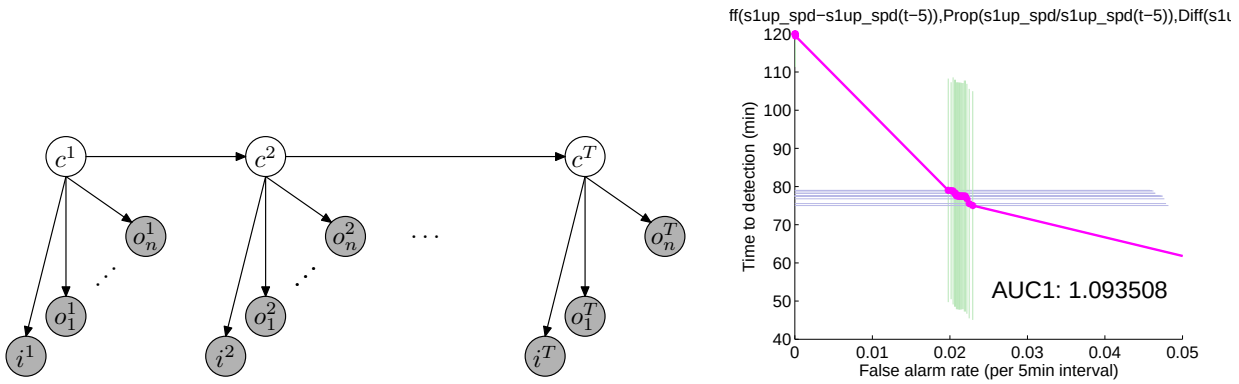


(a) t-test detector



(b) regression detector

Figure 20: AMOC curves for **a)** a t-test detector (see text), working with upstream volume. **b)** a regression detector registering drop in density downstream of the incident. These are the respectively best-performing features for these models and $N = 10$. The location is the same as for other detectors evaluated in this section, a portion of Parkway East in Pittsburgh, denoted "I376W".



(a) The graphical model

(b) The AMOC curve

Figure 21: The Dynamic Naive Bayes graphical model for detection and its performance characteristic. Note how the performance points cluster around FAR 0.2. This is a sign that there are few posteriors with moderate values and the posteriors form a strongly bimodal distribution. Sensitivity of such a detector will be difficult to adjust.

thus the regression detector tends to pick up the same spikes or drops in feature values.

6.3.6 Dynamic Naive Bayes

We define a dynamic Bayesian network model, with a single discrete hidden state variable s and a number of conditionally independent univariate Gaussian observation nodes $o_1, \ldots, o_n$. There is also a distinguished binary observation i, which is the incident state as observed by the TMC (Figure 21a).

Inspired by traffic flow theory [80, 137], which describes analogous flow states, we attribute the following semantics to the values of the hidden state variable: normal steady state ns is expected to undergo slow changes. The accident effect buildup ae captures the first minutes after an accident, characterized by a rapid spike in occupancy at upstream sensor, volume drop at the downstream sensor and a drop in speed at the upstream sensor. In the accident steady state as, capacity remains impaired, the upstream occupancy remains at the saturation limit, speed and throughput are lowered. The recovery phase rp is characterized by increasing speed at the upstream sensor and volume hovering near capacity at the downstream sensor.

The conditional distribution $p(i|s)$ is set by hand (Table 11) and serves to anchor the rows and columns of the inter-slice transition matrix $p(s^n|s^{n-1})$ to their intended interpretation.[2] The inter-slice transition matrix as well as the conditional probabilities $p(o_n|s^n)$ are learned from data.

An alarm threshold θ_a is set and alarm is triggered at time n if $p(s^n = ae|\mathbf{o}^n) + p(s^n = as|\mathbf{o}^n) \geq \theta_a$.

Unfortunately, the performance of the simple Dynamic NB models falls short of extracting the gains we

[2]The probabilities are of course somewhat arbitrary, but what is important is their relative magnitude.

	$p(i = v \mid s =$	$ns)$	$ae)$	$as)$	$rp)$
$v = 0$		0.01	0.30	0.90	0.10
$v = 1$		0.99	0.70	0.10	0.90

Table 11: The "anchoring" conditional probability table for DBN model of incidents

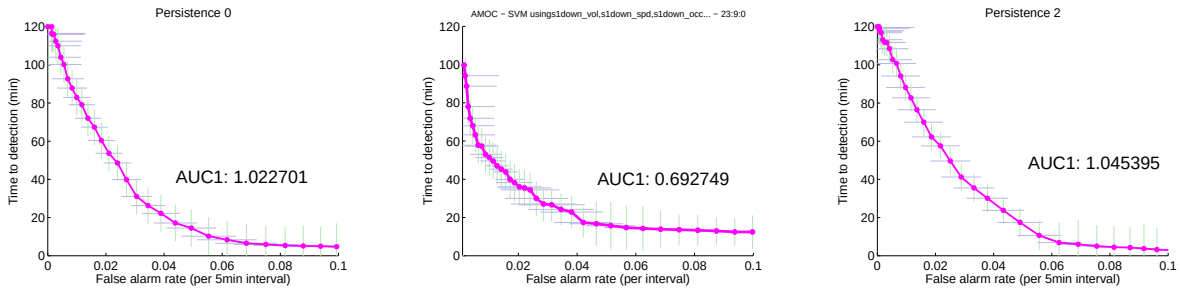
expect from a detector combining multiple features. This is most likely so because the Naive Bayes structural assumptions are a poor fit for the data. While it may also be the case that the coarse grain of the data does not warrant such fine distinction between traffic flow states, a similarly weak performance was seen in a model with only 2 states. Also, the distribution of posterior is unfavorable, resulting in a detector whose sensitivity is difficult to adjust.

On the other hand, it is unclear how to assign a probabilistic interpretation to the SVM detector described below or how it may be integrated into a framework that considers time. Herein lies a challenge for future work: a well performing classifier that can be "dynamized", such as a Bayesian network. We will combine the advantages of both SVMs and dynamic Bayesian networks in our integrated framework presented in Section 6.4.

6.3.7 The SVM detector

Let us now try the support vector machine (SVM) as the detection model. We had two reasons for choosing the SVM [106]. First, in preliminary experiments it outperformed logistic regression and several variations of dynamic Bayesian network detectors. Second, the SVM is fairly robust to irrelevant features, allowing us to include features that are weakly informative individually, but perhaps become strong predictors in aggregate. The SVM was learned in the straightforward way. Data points falling into the intervals labeled as "incident" in the data were assigned class 1, the remaining data points class -1. Misclassification cost was selected as to balance for unequal class sizes: if there are N instances of class 1 and M instance of class -1, then the misclassification of "-1" as "1" costs N/M and 1 vice versa.

The performance of the SVM detector using different feature sets can be seen in curves and the associated $AUC_{1\%}$ values in Figure 23. It appears that for our data, the direct sensor readings (speed, volume, occupancy) provide most of the detection leverage. Addition of the spatial difference (and proportion) features affects the performance minimally. The temporal difference features do bring a small improvement, albeit one that fails to confirm the perception of temporal difference as an important feature [107]. This could be in part explained by the fact that our data are 5 minute averages and the sharp temporal effects important for detection are somewhat averaged out. A detector using the features of the CA2 algorithm is included for comparison. The results confirm our intuition: the SVM detectors using multiple features outperform that



(a) $k = 0$ (b) $k = 1$ (c) $k = 2$

Figure 22: Performance with persistence checks: **a)** the SVM detector without the persistence check, **(b)** the SVM detector with 1-persistence check — incident must be detected in two consecutive data points for an alarm to be raised. **(c)** the SVM detector with 2-persistence check

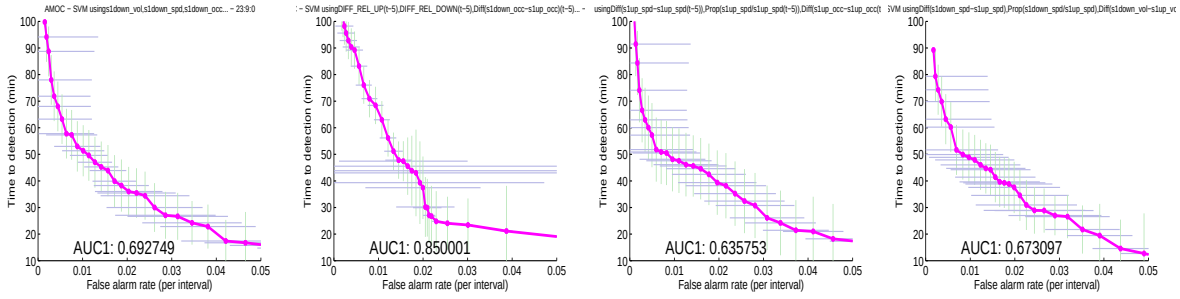
using only CA2 features (the comparison to CA2 itself follows later).

6.3.7.1 Persistence checks False alarm rate can be traded for detection time with the alarm signal post-processing technique known as persistence check. k-persistence check requires that the alarm condition persist for k time periods before the alarm is raised. Note that CA2 already has a built-in persistence check in its last step. We experimented with the optimal choice of k for the SVM detector with the basic measurement features (on the training site data). The results are shown in Figure 22. Since the area under curve is minimized at $k = 1$, all of the reported SVM experiments use that persistence value. Conveniently, this also matches the persistence check step of the CA2 algorithm, putting both algorithms on an even footing.

6.4 OVERCOMING NOISE AND BIAS IN LABELS

The major obstacle to further improvement appears to be the data itself. A systematic bias exists in the data with respect to incident times. Incidents are necessarily reported after they happen. Because the actual incident time is unknown, the time recorded it that of the report.

Our solution relies on human-aligned data from a *single* site. The model is a Dynamic Bayesian network that learns the amount of offset between manifestation and recording of an incident together with the effect of incident on sensor measurements. The model is trained on the manually re-aligned data, but once learned it can be used to initialize learning of the accident impact to other sites. The rationale is that while the differences between sensors are enough to impair the performance of an detector simply replanted from one



(a) A (b) B (c) C (d) D

Figure 23: Performance of the SVM model, for different feature sets. The features are: (A) All readings for the sensor. (B) California 2 features (the occupancy ratios). (C) All of current and previous step measurements. (D) All current measurements together with differences and proportions of the corresponding readings at the upstream and downstream sensors. For drawing the curves, the intercept of the SVM hyperplane is varied in the (-1,1) range, giving a lower estimate on the true performance [7]. For each value the detection threshold is set to, we compute the average FAR and TTD over the 10 train/test splits and draw the graph point as well as both of the corresponding error bars. The area under the portion of the curve up to 1% FAR is reported.

location to another without calibration, the underlying physical processes are qualitatively the same. The newly realigned labels are then combined with a supervised classifier such as a SVM to obtain the detector. The classifier uses features such as occupancy or drop in volume. This approach allows us to learn, with a *limited* amount of human effort, a more reliable detection model that is transferable to other sensor sites. It is conceivable that no further labeling needs to take place if we incorporate the knowledge gained here into an initial relabeling model for, say, a different city.

Our objective is to detect the accident as soon as its impact *manifests* in sensor readings. This time will always lag the time the accident actually *happened*. The lag amount depends, among other factors, on the capacity utilization of the roadway and the relative position of the sensor and accident locations. The time that the incident is *reported* to the TMC and logged in the database may precede or follow the time of manifestation. Differences between these times lead to label misalignment.

There are two states that the detector can learn to recognize: the short period when the accident's impact builds up (e.g. speed falls) around the sensor, and the longer steady state condition with lowered speeds or jammed traffic. To optimize detection time, we should focus the detector at the transient period. The transient period is very short and any misalignment will cause the accident start label to fall outside of it. It is therefore crucial for supervised learning that the label is precisely aligned with the observed impact of the accident. The end-of-incident labels are less important: by the time the incident is cleared, the emergency

response has already taken place. We do not attempt to align incident clearing times.

By definition, misalignment can only occur in positive instances, i.e. those sequences that contain an incident. We need a method to correct the alignment of incident labels in the training set so that the learned model accuracy may improve.

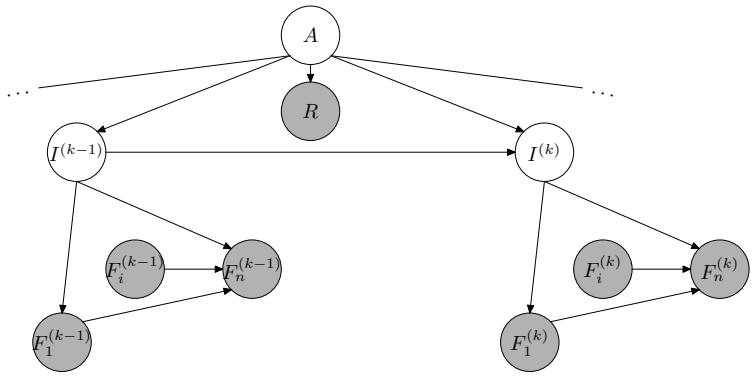


Figure 24: Two slices of the temporal probabilistic model for realignment. As usual, shaded nodes represent observed random variables; unshaded nodes correspond to latent variables. There are a total of L slices; the superscript (k) denotes the variable's instantiation in the k-the time slice.

6.4.1 A model for label realignment

Consider a single positive sequence S of traffic feature vectors. An incident start label r denotes the point in sequence S where the single incident is reported to occur. The label realignment task is to output the label ℓ pointing where the incident truly began to manifest in S. For label realignment, we model the sequence of feature vectors with a special dynamic Bayesian network model, shown in Figure 24. In the model, A represents the true accident time and takes on a value from $\{1, \ldots, L\}$, where L is the sequence length. Each *impact* variable $I^{(k)}$ is a binary indicator of incident impacting the traffic flow at time k. Each I is a part of the intra-slice Bayesian network that models the interaction between the traffic measurement features $F_1, \ldots, F_n$. We place no restrictions on the within-slice network in general. In order to keep the model presentation simple, we do not draw arrows between the features in subsequent slices. However, features may depend on values at other nearby time points; for instance the "occupancy derivative" $F(t) = Occ(t) - Occ(t-1)$ depends on the previous measurement.

The variables A and $I^{(k)}$ have a special relationship, expressed in their probability distributions. First, we express the lack of prior knowledge about A by defining the prior $P(A)$ to be the uniform distribution on the set $\{1, \ldots, L\}$. Second, the conditional distribution is also fixed, expressing that once an incident starts

impacting traffic, it continues to do so:

$$\begin{aligned} P(I^{(k)} = 1 | A = k', I^{(k-1)} = 1) &= 1 \\ P(I^{(k)} = 1 | A = k', I^{(k-1)} = 0) &= 1 \text{ if } k = k', \\ & \quad 0 \text{ otherwise.} \end{aligned} \tag{6.1}$$

We can afford this simplification because we only want to model the accident onset and disregard the accident clearing event.

The report time R depends on the true accident time and is assumed to obey a conditional Gaussian distribution: $P(R|A = k) \sim N(k + \mu, \sigma^2)$, with μ, σ identical for all k. Equivalently, the amount of misalignment has a stationary Gaussian distribution: $R - A \sim N(\mu, \sigma^2)$.

6.4.2 Inference for realignment

Using the probability distribution p defined by the above model, the label alignment task can be formulated as a posterior mode problem. Given that the data places the incident start label at r, we reassign the label to ℓ, so that

$$\ell = \operatorname*{argmax}_k p(A = k | R = r, \mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(L)}), \tag{6.2}$$

where $\mathbf{F}^{(t)} = (F_1^{(t)}, \ldots, F_n^{(t)})$ is the t-th observation vector.

Assume for now that we would like to evaluate this inference query with variable elimination. Note that the special form of the inter-slice probability distribution simplifies the inference task – there are only L indicator sequences with $p(I_1, \ldots, I_L) > 0$ to sum over. This is an instance of a *persistent* dynamic Bayesian network. Persistent DBNs contain chains of binary variables with a sink state, reducing the number of configurations from exponential to linear in the number of time slices as in this network.

An inference algorithm for persistent DBNs is developed in our paper [143]. In this particular model, the sequence $I_1, \ldots, I_L$ can be replaced by a single node $\tilde{I}$ with the meaning of "the time slice when the incident happened", also known as the *changepoint*. In [143], we generalize this transformation for a wider class of DBNs. A fast exact inference algorithm exists whenever a DBN has a low-treewidth one-slice structure and contains "many" persistent variables. The algorithm is equivalent to belief propagation, but takes advantage of a regular CPT structure induced by the changepoint transformation. The regular structure enables us to define a set of recurrences to compute belief propagation messages in time that is linear, instead of quadratic, in the variable domain size. In effect, an L-times (number of time slices) speedup versus straight belief propagation is achieved.

6.4.3 Learning and transfer to new locations

To transfer the detection model to a new location, we must parameterize a separate model for each sensor pair defining a highway segment (site). Let A denote the single calibration (training) site and let B_j, $j = 1, \ldots, S$

be the test sites. While one could learn the model in a fully unsupervised manner with the EM algorithm [42], there is little guarantee that the learning would converge to the intended interpretation. Instead, we first learn p^A, the sequence model for A, from manually aligned data. By assuming that the impact of an incident on traffic flow will be qualitatively similar at the remaining sites B_j, we can initialize the EM algorithm well with θ_A. Formally, the manual alignment gives us a fully observed dataset $\mathbf{X}^A = (\mathbf{F}^A, R^A, I^A, A^A)$ and maximum likelihood learning becomes trivial:

$$\Theta^A_{ML} = \underset{\Theta}{\operatorname{argmax}}\, p(\mathbf{X}^A|\Theta) \tag{6.3}$$

Then, inference in the model parameterized with Θ^A_{ML} can be applied to realign the labels for the B-sites where the manual annotation is unavailable. Of course, accident impact at each site B_j differs from that of the site A. The resulting labeling will be imperfect, but it still provides a good initial estimate. The EM-algorithm for estimation of Θ^{B_j} can proceed from there with a smaller risk of converging to an undesirable local optimum. Additionally, the sufficient statistics obtained in the estimation of Θ^A are stored and used to define the conjugate prior over Θ^{B_j}. Thus the resulting parameterization of a testing site model is a maximum a posteriori (MAP) estimate:

$$\Theta^{B_j}_{MAP} = \underset{\Theta}{\operatorname{argmax}}\, p(\mathbf{X}^{B_j}|\Theta)p(\Theta|\Theta^A_{ML}). \tag{6.4}$$

In the EM algorithm that estimates $\Theta^{B_j}_{MAP}$, the expectation step corresponds to inference of the unobserved labels A^{B_j} and I^{B_j}. The maximization step re-estimates the parameters of the conditional distributions $p(R|A)$ and $p(F_i|I)$. We consider the EM converged if the labeling (the posterior modes, see Equation 6.2) does not change in two consecutive iterations. For our dataset, the EM always converges in less than 5 iterations.

6.5 EVALUATION

We now test the performance of the realignment procedure, using as proxy the performance of the learned models on the task of detecting incidents.

6.5.1 Data

We selected 6 sites to evaluate our model on (Table 12). They are the six sites with the highest numbers of accident reports. The incident reports at S_{Train} were manually aligned to the incident manifestations in the data. The manual realignment was also aided by the free-text incident descriptions from the TMC database.

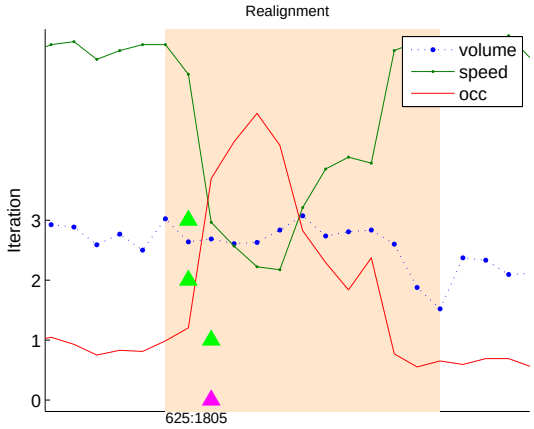


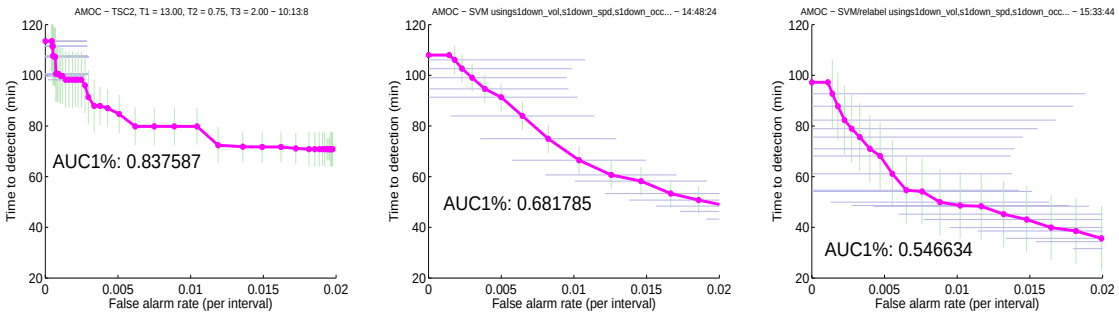
Figure 25: Convergence of accident position labeling. Accident labels are marked by upward-pointing triangles. The bottom (magenta) triangle marks the label assigned by the model p^A learned on the hand-labeled data; the green triangles (going upwards) represent successive iterations of the EM. The shaded area corresponds to the incident duration as recorded in the data. In this instance, p^A "overcorrected", but the following iterations of EM-estimation of p^B converged to a very satisfactory estimate.

6.5.2 Experimental setup and model parameters

We use the realignment framework in two variants that differ in how many dependencies between traffic features they model.

6.5.2.1 Naive Bayes slices To represent incident impact on traffic, we first use a Naive Bayes intra-slice model with binary indicator I and two features, F_1: the difference in occupancy at the upstream sensor in the previous and following interval and F_2: the same difference in speed. Both features are assumed to follow a conditional Gaussian distribution. This model makes do with very little information and strong independence assumptions.

6.5.2.2 TAN slices Secondly, relaxing the independence assumptions made by the NB model, we model the traffic impact of an incident with a Tree-Augmented Naive Bayes slice structure [50]. The Tree-Augmented Naive Bayes (TAN) model has been shown to improve, often dramatically, the modeling precision over the plain NB model; yet it retains favorable inferential complexity. In this arrangement, the class variable I again corresponds to the incident indicator. As features, we use all traffic quantities and their spatial and temporal differences and proportions. Since all these are continuous, we will again use a Gaussian random variable for each quantity.



(a) California 2 algorithm (b) SVM with original data (c) SVM on manually realigned data

Figure 26: Effect of data quality on SVM detector performance. Train site A with human-labeled data. Detection performance of (a) California 2 (b) SVM learned on original labeling, (c) SVM learned on the relabeled data.

TAN learning

The learning of a TAN model has two distinct phases: structure and parameter learning. The structure is learned so as to obtain the tree structure that has the maximum potential to capture dependencies present in the data. Chow and Liu [30] prove that this is achieved by a maximum-weight spanning tree, where edges are weighted by the mutual information. For a Gaussian, the mutual information is

$$I(x_u, x_v) = -\frac{1}{2}\log(|\hat{\Sigma}_k|/(\sigma_u^2\sigma_v^2)),$$

where (x_u, x_v) is an arbitrary pair of random variables, $\hat{\Sigma}_k$ is the maximum likelihood estimate of the 2×2-element covariance matrix of (x_u, x_v); and σ_u^2 and σ_v^2 are its diagonal elements.

Once the structure is fixed, standard expected loglikehood ascent (EM) can be used.[3] We use the Bayes Network Toolbox [118] to actually represent the network and implement the learning.

TAN realignment

To fit TAN in our framework, we first learn the structure and parameters on the hand-aligned data from the training site. The entire dataset is used, which ensures greater stability of the learned tree structure

[3] The prior $p(I)$ must be discarded as the learned network is plugged into the larger dynamic network, where $p(I^k|A)$ is fixed. It is anyway incorrect for the realignment task, where the proportion of positive instances is much higher.

Site	S_{Train}	S_{Test1}	S_{Test2}	S_{Test3}	S_{Test4}	S_{Test5}	S_{Test6}
Road	I376W	RT-28S	I279N	I279S	I376E 2	I376E 1	RT-28N
inc	145	100	97	92	107	84	100

Table 12: Sites included in the evaluation, with number of incidents and indication of the roadway. Some roadways have multiple incident sites.

Algorithm 2 Evaluation setup for model transfer to new site.

$M^A \leftarrow LearnStructure(AllData(OrigSite))$

$\Theta^A \leftarrow LearnMLParams(AllData(OrigSite))$

for $fold = 1$ **to** 10 **do**

$[TrainData_{fold}, TestData_{fold}] \leftarrow Split(AllData(NewSite))$

$\Theta^B \leftarrow LearnMAPParams(TrainData_{fold}, M^A, \Theta^A)$

$NewLabels \leftarrow \underset{A}{\operatorname{argmax}}\, p(A^B | TrainData_{fold}; \Theta^B, M^A)$

$Detector_{fold} \leftarrow LearnSVM(TrainData_{fold}, NewLabels)$

end for

$AMOC \leftarrow Evaluate(Detector_{:}, TestData_{:})$

M^A. The structure is fixed and parameters Θ^A are learned from the training data only. In the transfer phase for the new sites, the network structure is kept ($M^B = M^A$) and only parameters Θ^B are learned. In principle, the structure M^B can also be learned, starting from the training site model M^A. However, we found experimentally that the learned structures M^B exhibit too wide a variance if structure learning is allowed to take place in the transfer step, leading us to conclude there is simply not enough data for structure learning. Moreover, only positive – by definition anomalous – instances are included in the transfer step learning! For better clarity, the learning setup pseudocode is shown in Algorithm 2.

The TAN model contains all of the features mentioned above (the measurements, their spatial and temporal differences and proportions). It also makes as few independence assumptions as possible for it to remain tractable. Thus it lies on the "opposite end of the tractable spectrum" as Naive Bayes. If we find similar levels of performance in the NB and TAN models, we may conclude with a degree of confidence that realignment works because of the temporal dynamic Bayes net structure, as we expected.

TAN inference

Inference in TAN model can still be performed in time linear in L and n, the number of features. Recall we are after the posterior $p(A|\mathbf{F}, R)$. By Bayesian network probability decomposition,

$$p(A|\mathbf{F}, R) \propto \sum_{\mathbf{I}} p(A)p(R|A)p(\mathbf{I}|A) \prod_{k=1}^{L} p(\mathbf{F}^k|I^k).$$

Moreover, the term $p(\mathbf{I}|A)$ is deterministic and the prior on A is uniform:

$$p(A|\mathbf{F}, R) \propto p(R|A) \prod_{k=1}^{L} p(\mathbf{F}^k|I^k(A)).$$

(See also [143] for a more general view of such simplifications of inference.) The probabilities $p(\mathbf{F}^k|I^k)$ for $I^k = 0$ and $I^k = 1$ are easily obtained by definition of Bayesian network probability when observed, and using standard inference when missing. The TAN structure ensures that $p(\mathbf{F}^k|I^k)$ is tractable.

Area under first 1% of FAR range		Site						
		Train	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
	Road	376W	28S	279N	279S	376E 2	376E 1	28N
method	CA2	0.838	0.451	1.177	1.180	1.108	0.726	1.081
	SVM/orig	0.682	0.179	0.807	0.474	0.613	0.740	0.808
	SVM/NB	0.547	0.149	0.763	0.389	0.561	0.712	0.771
	SVM/TAN	0.553	0.152	0.783	0.294	0.567	0.698	0.720

Table 13: Summary of the $AUC_{1\%}$ performance statistics for the three detection algorithms and six evaluation sites. Some sites are more amenable to automatic detection, but consistent improvement is noted from CA2 to SVM on original data to SVM on realigned data.

6.5.3 Evaluation results

The quality of the resulting labels is most relevantly measured by the improvement in the $AUC_{1\%}$ performance metric of a classifier learned on the realigned data. The $AUC_{1\%}$ values for the three methods (CA2, SVM, SVM after relabeling) are summarized in Table 13. Note that the $AUC_{1\%}$ values do not have well defined standard deviations, as we obtain a single AMOC curve per testing location. The curves themselves, however, are 10-fold cross-validated averages and one can glean the amount of variance involved from the curves in Figure 26. The table shows that the SVM detector learned on the original data consistently improves over the CA2 method for every testing site. Similarly, the SVM detector learned on the label-realigned data realizes an improvement over the original SVM detector. The absolute performance varies significantly between testing sites as it depends on a number of site specifics: the distance between the accident site and the upstream sensor, volume of traffic, the presence of a shoulder lane where the vehicles may be removed from the flow of traffic, etc.

6.6 CONCLUSIONS

Our experiments showed that no single traffic feature may provide the most discriminative performance. Detectors that combine multiple features top out at a better performance.

All detectors (at least those discussed here) produce a "signal", whether it is the distance from mean measured in standard deviations, posterior probability of incident from a graphical model, or the distance of the point in measurement space from the separating hyperplane. A good detector will have a signal that is almost uniformly distributed in its respective range. A well spread-out distribution of signal allows for effective adjustment of the detector's sensitivity and specificity. Detectors that do not have this property, as we have seen in case of the dynamic Bayes net detector, will be unable to achieve some operating points, especially the desirable ones with low false alarm rates.

Supervised learning is a viable approach to construction of incident detection algorithms. We have shown it surpasses the performance of unsupervised anomaly detection techniques and easily leads to detectors that outperform traditional hand-crafted detectors. With sufficient data now available, it can do away with the problem of manual tuning and re-tuning of the detectors to adapt to new deployment locations and changing traffic patterns.

However, data obtained from such complex systems is inherently noisy. We proposed an algorithm that deals successfully with noise in event label timing and demonstrated that it improves the data quality to allow more successful learning of incident detectors.

Of course, a number of specific questions about our approach remain open. One could devise finer incident models and offset distributions; relax the assumption of independence of time-to-recording and incident impact severity – a more severe accident is perhaps more easily noticed. Explicitly modeling time-of-day and the expected traffic pattern looks especially promising as it permits the definition of an "unexpected" congestion, presumably more indicative of an accident. On the other hand, we advocate a cautious approach to more complex models. While data on traffic flow is plentiful, incidents and related data are (fortunately) relatively sparse, which makes more complex models vulnerable to overfitting, as we began to notice with the TAN realignment model, especially with regard to structure learning.

While the realignment algorithm was motivated by and presented in context of incident detection, it is generally applicable to situations where time of events is marked noisily in data streams. For instance, similar uncertainty in labeling alignment accompanies detection of intonation events in speech recognition [160].

7.0 CONCLUSION AND FUTURE CHALLENGES

7.1 CONTRIBUTIONS

The socio-economic importance of transportation and communication networks necessitates development of methods and models that enable thorough analysis and understanding of richly connected stochastic systems. Machine learning and statistical modeling have much to offer to this important application domain, from better ways to reason about dependencies among traffic quantities, to planning with predictive traffic flow models, to easier-to-calibrate incident detection algorithms. All of these contributions were enabled by abundance of traffic data.

This thesis [142] has presented four main contributions to the research that concerns the application of learning and probabilistic reasoning to transportation networks:

- Mixture of Gaussian trees – a new generative model for multivariate continuous densities, used to represent static dependencies among traffic measurements at various locations. The model has fewer parameters than a full-Gaussian model and therefore its parameter estimates should be more accurate given the same volume of data, allowing it to scale better to larger networks.
- A probabilistic model of traffic dynamics that combines a physical flow model with the ability to provide a predictive distribution, which we show to be a better match to the data than the distribution implied by ARMA, a time-series model. Moreover, the parameters in our model have a clear interpretation; and parameter fitting is easier and finer-grained than for traditional macroscopic flow models.
- Route optimization algorithms that build on the new model of traffic dynamics and take advantage of traffic data availability. While the proposed algorithms are not the best choice for the Pittsburgh traffic network, we show that the proposed k-ROBUST becomes the choice when volatility increases. Furthermore, the approximate planning algorithm is very general with respect to the optimization criteria it can use.
- An improved incident detection model driven by the new data re-alignment framework. We demonstrated that a learning approach to incident detection can outperform traditional decision tree based models deployed in many Intelligent Transportation Systems and argued that it requires less expert effort to recalibrate. Performance of detectors learned in a supervised manner suffers greatly if the labeling process is noisy. We gave a dynamic Bayesian network framework for rectifying label bias in temporal datasets that corrects the problem with minimal amounts of manual tagging.

These algorithms improve on the state of the art implemented in Intelligent Transportation Systems (ITS). As such, in the ideal scenario, they could eventually become parts of an integrated ITS that helps traffic managers in their daily duties. Incident detection and traffic prediction are among the core capabilities of a comprehensive ITS. In case of an inevitable multiple sensor outage, the proposed joint density model (mixture of Gaussian trees) is a viable option for completing the operational picture for traffic managers and users of ITS data. While routing problems in the context we considered rarely appear in the traffic management context, many consumers and transportation companies value routing services based on traffic prediction.

Given the costs of congestion and environmental damage, the transportation world is sure to turn to anyone offering solutions to squeeze more efficiency out of existing road networks. I believe I have shown that machine learning will not be left on the sidelines.

7.2 OPEN PROBLEMS

Many open problems and future challenges remain in the area of reasoning about complex systems such as transportation networks. Some of the challenges stem from complexity; others, such as sensing issues, come from its embedding in physical space.

7.2.1 Partial observability

In this thesis, we assumed in that the sensors give perfect measurements and made other assumptions that amounted to full observability. If the sensors are noisy, the road network is not fully covered by sensors or if some of the sensors fail, then the proper planning formalism is a partially observable (Semi-)Markov decision process (POMDP) [6, 76, 66] with a continuous state space. Unfortunately, a POMDP problem is very hard to solve exactly even for small discrete state problems [126]. It remains an open question if the special properties of the problems addressed here simplify the solution of partially observable MDPs as well and how exactly they can be leveraged.

7.2.2 New data

Just as data have changed the problems and solutions in the past, the field will continue to be challenged and transformed by advances in information technology. Instead of a set of fixed sensors placed in a few well thought-out strategic locations, the technology is moving rapidly towards massive numbers of cheap sensors. These can be either stationary or vehicle-based sensors that report wirelessly to a central location. However, due to GPS technology limitations, measurements from mobile sensors are not as precise as those of dedicated

fixed sensors. Also, communication is subject to stochastic network behavior of its own. Interruptions and congestion on the wireless data networks introduce variable delays and data loss.

As a result, we are faced with new data models that do not come in the neat form of a "matrix $\mathbf{X}$" and will require new formalisms to manipulate effectively. Nevertheless, they still suffer from the familiar problems of noise and missing data and probabilistic approaches are still our best angle of attack to deal with these.

7.2.3 Temporal noise in sensing and inference

A large issue we lightly grazed in the incident detection setting is what I call temporal noise, where we work with labeled sequence data, but the labels of events are placed only approximately correctly on the time axis. We have shown this kind of temporal noise makes a difference in performance of learned models and is usually a result of the information taking a variable amount of time to propagate to the right places. How do we efficiently reason with information that arrives out of temporal order? This might be the case with a vehicle transmitting its data after spending several minutes in a tunnel without signal. Incremental algorithms for temporal reasoning models such as Bayesian networks are ill-prepared to deal with out-of-order evidence patterns, preferring to assume a computationally convenient model where observations arrive synchronously and immediately.

7.2.4 Spatial vector fields

Moreover, spatial information may be subject to noise. For instance, GPS information in a moving vehicle may not be precise enough to distinguish between spatially close but physically separate lanes that go in different directions. This is of particular interest in complex interchanges. The data arriving from vehicles with imprecise position sensors looks more like a vector field. I believe we will see much more of continuous vector field models to capture these situations in the future. Tools such as Gaussian processes already exist to help model such objects. However, anchoring the observations and embedding prior knowledge (such as the exact position of the actual road) with good precision may require computationally unfavorable prior distributions. A new world of tradeoffs and research opportunities opens here.

7.2.5 From single vehicle to multi-vehicle route optimization

In our route planning exercise, we planned for a single vehicle moving in a large environment that is unaffected by it. This is clearly not the case if a sufficiently large number of users adopts the algorithm. One possibility to avoid costly congestions is to make the algorithm's recommendation randomized with the explicit goal of achieving some notion of equality. For this, however, the algorithm needs to have an estimate of how many vehicles are going where, because traffic can be seen as an instance of the minority game well known from game theory. There is no clear way around a central repository of origin-destination information.

If the serious privacy issues are resolved, the drivers might accept such a system. In essence, the route recommendation is an incentive for the user to provide information to build an up-to-date system-wide origin-destination matrix. Once the matrix is built, all vehicles need to be assigned a route, a problem called traffic assignment. The traffic assignment task is actually quite well studied, motivated by existing more tightly controlled transportation (rail, air) and communication network scenarios. However, the degree of "noise" will be substantially higher in road networks as drivers change their minds, make driving errors or deliberately misrepresent their destination. Game-theoretical approaches are sure to play a central role here.

APPENDIX

GLOSSARY

Congested flow is a mode of traffic behavior where demand exceeds capacity. Speeds fall, densities rise and a traffic jam forms.

Density A traffic flow quantity, the number of vehicles per one mile of travel lane (units "veh/mi").

Free flow A mode of traffic behavior when interactions between vehicles are negligible. As the number of vehicles increases, free flow transitions into synchronized flow.

Induced demand Consider a roadway with a poor level of service (LOS). It is often observed that adding capacity in response to a poor level of service results in an increase of demand [1]. Drivers not only alter their routing decisions to use the new capacity, but new trips may be generated to locations that were previously not worth accessing.

Level of service is defined in the HCM [164] and serves as general indication of driving conditions. Levels range from A — an essentially empty freeway — to F, stalled traffic.

Loop detector A loop of wire embedded in the pavement. Passing vehicles induce current in the loop and the voltage spikes are counted. Double-loop detectors consist of two closely spaced loops and, unlike single-loop detectors, can directly measure vehicle speed by measuring the time between detections.

Occupancy A dimensionless quantity, the proportion of time that a vehicle is over the detector loop. With single-loop detectors, occupancy and volume are used to impute speed [55] $s \propto \frac{volume}{occupancy}$.

Reversible lanes Some travel lanes are designed to function in both directions to adapt the link to diurnally variable demand. The access to these lanes is controlled by variable signs displaing $\times$ or $\downarrow$. For instance, the Liberty bridge in Pittsburgh has four lanes, three of which are open in the inbound direction in the morning. In the afternoon, the two center lanes reverse to serve oubound traffic, for a total of three outbound lanes and one inbound lane.

Synchronized flow is a mode of operation in which vehicles are restricted to match the speed the preceding vehicle by the vehicles occupying the adjacent lanes. However, the speed of the flow is close to maximum design roadway flow, unlike in congested traffic.

Ramp metering A traffic control device placed on an onramp of a freeway that only allows a limited number of vehicles to enter the road during a time interval. The device is remotely controlled, usually manually, so as to maintan optimal volume flow on the freeway mainline.

BIBLIOGRAPHY

[1] *Special Issue of* Transportation *on Induced Traffic*, volume 23, February 1996.

[2] M.S. Ahmed and A.R. Cook. Analysis of freeway traffic time-series data by using Box-Jenkins techniques. Technical Report 722, Transportation Research Board, 1979.

[3] C. Andrieu, A. Doucet, S. S. Singh, and V. B. Tadic. Particle methods for change detection, system identification, and control. *Proceedings of the IEEE*, 92(3):423–438, 2004.

[4] Guin Angshuman. *An Incident Detection Algorithm Based on a Discrete State Propagation Model of Traffic Flow*. PhD thesis, Georgia Institute of Technology, 2004.

[5] Charalambos N. Antoniades and Yorgos J. Stephanedes. Single-station incident detection algorithm (ssid) for sparsely instrumented freeway sites. *Transportation Engineering*, 1996.

[6] K.J. Astrom. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.

[7] Francis Bach, David Heckerman, and Eric Horvitz. On the path to an ideal ROC curve: Considering cost asymmetry in learning classifiers. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedins of AISTATS05*, pages 9–16, 2005.

[8] James L. Bander and Chelsea C. White. A heuristic search approach for a nonstationary stochastic shortest path problem with terminal cost. *Transportation Science*, 36(2):218–230, 2002.

[9] M.G.H. Bell. The estimation of an origin-destination matrix from traffic counts. *Transportation Science*, 17:198–217, May 1983.

[10] Richard E. Bellman. *Dynamic programming*. Princeton University Press, Princeton, NJ, 1957.

[11] Richard E. Bellman and S. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Princeton, 1962.

[12] D Belomestny, V Jentsch, and M Schreckenberg. Completion and continuation of nonlinear traffic time series: a probabilistic approach. *Journal of Physics A: Math. Gen.*, 36:11369–11383, 2003.

[13] Dimitri P. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific, 1995.

[14] Dimitri P. Bertsekas and John N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3), August 1991.

[15] J. Besag, J. York, and A. Mollie. Bayesian Image Restoration With Two Applications In Spatial Statistics. *Annals of the Institute of Statistical Mathematics*, 43(1):1–59, 1991.

[16] Christopher M. Bishop. Latent variable models. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 371–403. MIT Press, 1999.

[17] David Blei, Andrew Ng, and Michael Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, Jan 2003.

[18] George Box and F.M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, Oakland, CA, second edition, 1976.

[19] Justin A. Boyan and Michael L. Littman. Exact solutions to Time-Dependent MDPs. In *Advances in Neural Information Processing Systems*. MIT Press, 2001.

[20] John Bresina, Richard Dearden, Nicolas Meuleau, Sailesh Ramakrishnan, David Smith, and Rich Washington. Planning under continuous time and resource uncertainty. In *Proceedings of UAI'02*, pages 77–85, 2002.

[21] Fritz Busch and Martin Fellendorf. Automatic incident detection on motorways. *Traffic Engineering and Control*, 31(4):221–227, 1990.

[22] P. P. Chakrabarti, S. Ghose, and S. C. DeSarkar. Admissibility of ao^* when heuristics overestimate. *Artificial Intelligence*, 34(1):97–113, 1987.

[23] R. Chellappa and A. Jain, editors. *Markov Random Fields - Theory and Applications*. Academic Press, 1993.

[24] Chao Chen, Jaimyoung Kwon, John Rice, and Alexander Skabardonis. Detecting errors and imputing missing data for single loop surveillance systems. *Transportation Research Record*, 1855:160–167, 2003.

[25] D. Chen, J. Zhang, S. Tang, and J. Wang. Freeway traffic stream modeling based on principal curves and its analysis. *Intelligent Transportation Systems*, 5(4):246–258, December 2004.

[26] Haibo Chen, Mark S. Dougherty, and Howard R. Kirby. The effects of detector spacing on traffic forecasting performance using neural networks. *Computer-Aided Civil and Infrastructure Engineering*, 16:422 – 430, November 2001.

[27] Haibo Chen and Susan Grant-Muller. Use of sequential learning for short-term traffic flow forecasting. *Transportation Research Part C*, 9:319–336, 2001.

[28] Haibo Chen, Susan Grant-Muller, Lorenzo Mussone, and Frank Montgomery. A study of hybrid neural network approaches and the effects of missing data on traffic forecasting. *Neural Computing and Applications*, 10:277–286, 2001.

[29] David Maxwell Chickering and David Heckerman. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29(2-3):181–212, 1997.

[30] C. J. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Inf. Theory*, 14(3):462–467, 1968.

[31] Stepen D. Clark, M.S. Dougherty, and H.R. Kirby. The use of neural networks and time series models for short term traffic forecasting: a comparative study. In *Transportation Planning Methods: Proceedings of PTRC 21st Summer Annual Meeting*, 1993.

[32] Stephen Clark. Traffic prediction using multivariate nonparametric regression. *Journal of Transportation Engineering*, 129:161–168, 2003.

[33] Stephen D. Clark, H. Chen, and S.M. Grant-Muller. Artificial neural network and statistical modelling of traffic flows-the best of both worlds. In *Proceedings of the 8th World Conference on Transport Research*, pages 215–226. Elsevier Science, U.K., 1999.

[34] M. Collins, S. Dasgupta, and R. Schapire. A generalization of principal component analysis to the exponential family. In *Proceedings of NIPS 01*, 2001.

[35] Gregory Cooper, Denver Dash, John Levander, Weng-Keen Wong, William Hogan, and Michael Wagner. Bayesian biosurveillance of disease outbreaks. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pages 94–103, Arlington, Virginia, 2004. AUAI Press.

[36] Paul Dagum and Michael Luby. An optimal approximation algorithm for bayesian inference. *Artificial Intelligence*, 93:1–27, 1997.

[37] Daniel J. Dailey. Improved error detection for inductive loop sensors. Technical Report WA-RD 3001, Washington State Dept. of Transportation, May 1993.

[38] G.A. Davis, N.L. Niham, M.M. Hamed, and L.N. Jacobson. Adaptive forecasting of freeway traffic congestion. *Transportation Research Record*, 1287:29–33, 1991.

[39] H. Davis, A. Bramanti-Gregor, and J. Wang. The advantages of using depth and breadth components in heuristic search. In *Methodologies for Intelligent Systems*, volume 3, pages 19–28, 1988.

[40] Daniela Pucci de Farias and Benjamin Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Math. Oper. Res.*, 29(3):462–478, 2004.

[41] Thomas Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150, 1989.

[42] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood for incomplete data via the EM algorithm. *Journal of Royal Statistical Society*, 39:1–38, 1977.

[43] Hussein Dia. An object-oriented neural network approach to short-term trraffic forecasting. *European Journal of Operations Research*, 131:253–261, 2001.

[44] Hussein Dia, Geoff Rose, and Anthony Snell. Comparative performance of freeway automated incident detection algorithms, 1996.

[45] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[46] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, New York, 2001.

[47] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[48] Tal El-Hay, Nir Friedman, Daphne Koller, and Raz Kupferman. Continuous time Markov networks. In *UAI*, 2006.

[49] Federal Highway Administration. Traffic congestion and reliability: Trends and advanced strategies for congestion mitigation, 2005.

[50] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. In G. Provan, P. Langley, and P. Smyth, editors, *Machine Learning*, volume 29, pages 131–163, 1997.

[51] L. Fu, D. Sun, and L. R. Rilett. Heuristic shortest path algorithms for transportation applications: state of the art. *Computers and Operations Research*, 33(11):3324–3343, 2006.

[52] Liping Fu and L. R. Rilett. Expected shortest paths in dynamic and stochastic traffic networks. *Transportation Research Part B: Methodological*, 32(7):499–516, September 1998.

[53] D. Geiger and D. Heckerman. Beyond bayesian networks: Similarity networks and bayesian multinets. *Artificial Intelligence*, (82):45–74, 1996.

[54] S. Geman and D. Geman. Stochastic relaxations, Gibbs distributions and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[55] Todd L. Graves, Alan F. Karr, Nagui M. Rouphail, and Piyushimita Thakuriah. Real-time prediction of incipient congestion on freeways from detector data. *Submitted to Transportation Research B*, 1998.

[56] Jianhua Guo. *Adaptive Estimation and Prediction of Univariate Vehicular Traffic Condition Series.* PhD thesis, North Carolina State University, 2005.

[57] Srinivasa Ravi Chandra Haitham M. Al-Deek, Chilakamarri Venkata. New algorithms for filtering and imputation of real-time and archived dual-loop detector data in i-4 data warehouse. *Transportation Research Record: Journal of the Transportation Research Board*, 1867:116–126, 2004.

[58] H. Haj-Salem and N. Elloumi. Real data screening: Problem statement, methods and field results. In *Proceedings of IFAC Transportation Systems*, pages 1072–1077, June 1997.

[59] H. Haj-Salem and J. P. Lebacque. Reconstruction of false and missing data using a first order traffic flow model. *Transportation Research Record*, 1802:155–165, 2002.

[60] Randolph Hall. The fastest path through a network with random time-dependent travel time. *Transportation Science*, 20:182–188, 1986.

[61] M.M. Hamed, H.R. Al-Masaeid, and Z.M. Bani Said. Short-term prediction of traffic volume in urban arterials. *ASCE Journal of Transportation Engineering*, 121:249–254, 1995.

[62] Eric Hansen and Rong Zhou. Anytime a^* search. *Journal of Artificial Intelligence Research*, 28:267–297, 2007.

[63] Eric Hansen, Shlomo Zilberstein, and V. Danilchenko. Anytime heuristic search: First results. Technical Report 97-50, U of MA Amherst, Dept of Computer Science, 1997.

[64] Eric A. Hansen and Shlomo Zilberstein. lao^*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129:35–62, 2001.

[65] T. Hastie, R. Tibshirani, and J. Friedman. *Elements of Statistical Learning.* Springer, 2001.

[66] Milos Hauskrecht. Value function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.

[67] E. Hewitt and L. J. Savage. Symmetric measures on cartesian products. *Transactions of the American Mathematical Society*, 80:470–501, 1955.

[68] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*, Stockholm, 1999.

[69] E. Horvitz, J. Apacible, R. Sarin, and L. Liao. Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service. In *Twenty-First Conference on Uncertainty in Artificial Intelligence*, July 2005.

[70] Ronald A. Howard. Semi-Markovian decision processes. In *Proceedings of International Statistical Inst.*, Ottawa, Canada, 1963.

[71] Ling Huang, Xuanlong Nguyen, Minos Garofalakis, Michael Jordan, Anthony D. Joseph, and Nina Taft. Distributed PCA and network anomaly detection. Technical Report UCB/EECS-2006-99, EECS Department, University of California, Berkeley, July 14 2006.

[72] Finn V. Jensen. *An introduction to Bayesian networks.* Springer-Verlag, New York, 1996.

[73] William S. Jewell. Markov renewal programming: I. formulations, finite return models II. infinite return models example. *Operations Research*, 11:938–971, 1963.

[74] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Proc. International Symposium Aerospace/Defense Sensing, Simulation and Controls*, pages 182–193, 1997.

[75] Seungjae Lee Juyoung Kim. Dynamic travel demand estimation using real-time traffic data. Technical Report 06–2313, Transportation Research Board, 2006.

[76] L.P. Kaelbling, M. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1999.

[77] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45, 1960.

[78] Asim Karim and Hojjat Adeli. Incident detection algorithm using wavelet energy representation of traffic patterns. *Journal of Transportation Engineering*, 128(3):232–242, 2002.

[79] Michael Kearns, Yishay Mansour, and Andrew Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1324–1331. Morgan Kaufmann, 1999. Appeared in a special issue of the Machine Learning journal.

[80] Boris S. Kerner and Henry Lieu. *The Physics of Traffic : Empirical Freeway Pattern Features*. Springer, 2004.

[81] B.S. Kerner. Experimental features of self-organization in traffic flow. *Physical Review Letters*, 81:3797–3800, October 1998.

[82] B.S. Kerner. Dependence of empirical fundamental diagram on spatial-temporal traffic patterns features. *ArXiv Condensed Matter e-prints*, August 2003.

[83] Hyonungsoo Kim and David J. Lowell. Traffic information imputation using a linear model in vehicular ad hoc networks. In *Proceedings of the IEEE Intelligent Transportation Systems Conference*, 2006.

[84] Seongmoon Kim, Mark E. Lewis, and Chelsea C. White. Optimal vehicle routing with real-time traffic information. In *IEEE Transactions on Intelligent Transportation Systems*, volume 6, June 2005.

[85] Seongmoon Kim, Mark E. Lewis, and Chelsea C. White. State space reduction for nonstationary stochastic shortest path problems with real-time traffic information. *IEEE Transactions on Intelligent Transportation Systems*, 6, 2005.

[86] R. Kindermann and J.L. Snell. *Markov Random Fields and their Applications*, volume 1 of *Contemporary Mathematics*. AMS, 1980.

[87] Kitaoka, E. Teramoto, H. Oneyama, and M. Kuwahara. Development of the od estimation method for traffic condition prediction. In *Proceedings of the 12th World Congress on Intelligent Transport Systems*, 2005.

[88] Sven Koenig. A comparison of fast search methods for real-time situated agents. In *Proceedings of Third Int'l Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, 2004.

[89] Sven Koenig and Maxim Likhachev. D^* lite. In *Eighteenth national conference on Artificial intelligence*, pages 476–483, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[90] Sven Koenig and Maxim Likhachev. Real-time adaptive a^*. In *Proceedings of Fifth Int'l Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'06)*, 2006.

[91] Daphne Koller and Nir Friedman. *Bayesian Networks and Beyond*. Unpublished manuscript.

[92] Taehyung Kom, Hyoungsoo Kim, and David J. Lovell. Traffic flow forecasting: Overcoming memoryless property in nearest neighbor non-parametric regression. In *Proceedings of* 8^{th} *Int'l Conference on Intelligent Trasportation Systems*, 2005.

[93] Richard E. Korf. Real-time heuristic search. *Artificial Intelligence*, 41(1):41–78, 1990.

[94] A. Kotsialos, M. Papageorgiou, C. Diakaki, Y. Pavlis, and F. Middelham. Traffic flow modeling of large-scale motorway networks using the macroscopic modeling tool metanet. *IEEE Transactions on intelligent transportation system*, 3(4):282–292, 2002.

[95] Branislav Kveton. *Planning in Hybrid Structured Stochastic Domains.* PhD thesis, University of Pittsburgh, 2006.

[96] Branislav Kveton, Milos Hauskrecht, and Carlos Guestrin. Solving factored MDPs with hybrid state and action variables. *Journal of Artificial Intelligence Research*, 27:153–201, 2006.

[97] Jaimyoung Kwon and Kevin Murphy. Modeling freeway traffic with coupled HMMs, May 2000.

[98] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. Kolaczyk, and N. Taft. Structural analysis of network traffic flows. In *Proceedings of ACM SIGMETRICS*, 2004.

[99] Steffen L. Lauritzen. *Graphical Models.* Oxford University Press, 1996.

[100] Baibing Li and Bart De Moor. Dynamic identification of origin-destination matrices in the presence of incomplete observations. *Transportation Research Part B: Methodological*, 36:37–57, January 2002.

[101] Maxim Likhachev, David Ferguson, Geoffrey Gordon, Anthony Stentz, and Sebastian Thrun. Anytime dynamic A^*: An anytime, replanning algorithm. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2005.

[102] Maxim Likhachev, Geoff Gordon, and Sebastian Thrun. *ARA**: Anytime A^* with provable bounds on sub-optimality. In *In Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference (NIPS-03.* MIT Press, 2003.

[103] P-W. Lin and G.-L. Chang. Robust estimation of the dynamic origin-destination matrix for a freeway network. In *Proceedings of the 2004 IEEE International Conference on Networking, Sensing and Control*, 2004.

[104] C. Liu, T. Pai, C. Chang, and C. Hsieh. Path-planning algorithms for public transportation systems. pages 1061–1066, 2001.

[105] Chao-Lin Liu and Michael P. Wellman. Using stochastic-dominance relationships for bounding travel times in stochastic networks. In *proceedings of IEEE International Conference on intelligent Transportation Systems*, pages 55–60, 1999.

[106] O. L. Mangasarian and David R. Musicant. Lagrangian support vector machine classification. Technical Report 00-06, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, June 2000. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-06.ps.

[107] P.T. Martin, H.J. Perrin, and B.G. Hansen. Incident detection algorithm evaluation. Technical Report UTL-0700-31, Utah Traffic Laboratory, July 2000.

[108] I. Matschke, K. Heinig, and B. Friedrich. Data completion for improved od-estimation. In *12th IEEE International Conference on Road Transport Information and Control*, 2004.

[109] Adolf D. May. *Traffic Flow Fundamentals.* Prentice-Hall, Englewood Cliffs, N.J., 1990.

[110] G. McLachlan and D. Peel. *Finite Mixture Models.* Wiley, New York, 2000.

[111] M. Meilă and T. Jaakkola. Tractable Bayesian learning of tree belief networks. Technical Report CMU–RI–TR–00–15, Carnegie Mellon University Robotics Institute, 2000.

[112] Marina Meilă-Predoviciu. *Learning with mixtures of trees*. PhD thesis, MIT, 1999.

[113] A. Messmer and M. Papageorgiou. Metanet : A macroscopic simulation program for motorway networks. *Traffic Engineering and Control*, 31(8–9):466–470, Aug–Sep 1990.

[114] Panos G. Michalopoulos, Richard D. Jacobson, Craig A. Anderson, and Thomas B. DeBruycker. Automatic incident detection through video image processing. *Traffic Engineering and Control*, February 1993.

[115] Lyudmila Mihaylova and Rene Boel. A particle filter for freeway traffic estimation. In *IEEE Conference on Decision and Control*, 2004.

[116] Lyudmila Mihaylova, Rene Boel, and Andreas Hegyi. Freeway traffic estimation within particle filtering framework. *Automatica*, 43(2):290–300, 2007.

[117] Elise D. Miller-Hooks and Hani S. Mahmassani. Least expected time paths in stochastic, time-varying transportation networks. *Transportation Science*, 34(2):198–215, 2000.

[118] Kevin Murphy. The Bayes Net Toolbox for Matlab. *Computing Science and Statistics*, 33, 2001.

[119] Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, EECS, University of California, Berkeley, Berkeley, CA, July 2002.

[120] A. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of UAI-00*, 2000.

[121] Nils J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing, Palo Alto, CA, 1980.

[122] Iwao Okutani and Yorgost J. Stephanedes. Dynamic prediction of traffic volume through Kalman filtering theory. *transportation Research B*, 18(1):1–11, 1984.

[123] R.K. Oswald, William T. Scherer, and Brian L. Smith. Traffic flow forecasting using approximate nearest neighbor nonparameteric regression. Technical Report UVA-CE-ITS_01-4, University of Virginia, Center for Transportation Studies, 2000.

[124] Stefano Pallottino and Maria Grazia Scutellà. Shortest path algorithms in transportation models: classical and innovative aspects. Technical Report TR-97-06, Università di Pisa, 14, 1997.

[125] Stefano Pallottino and Maria Grazia Scutella. A new algorithm for reoptimizing shortest paths when the arc costs change. *Operations Research Letters*, 31(2):149–160, March 2003.

[126] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of optimal queueing network control. *Mathematics of Operations Research*, 24:293–305, 1999.

[127] Emily Parkanyi and Chi Xie. A complete review of incident detection algorithms and their deployment: What works and what doesn't. Technical Report NETCR37, New England Transportation Consortium, 2005.

[128] H.J. Payne and S. C. Tignor. Freeway incident-detection algorithms based on decision trees with states. *Transportation Research Record*, 682:30–37, 1978.

[129] B.N. Persaud, F.L. Hall, and L.M. Hall. Congestion identification aspects of the mcmaster incident detection algorithm. *Transportation Research Record*, 1287:167–175, 1990.

[130] Foster J. Provost and Tom Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Knowledge Discovery and Data Mining*, pages 43–48, 1997.

[131] Martin L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley, New York, 1994.

[132] D. Chris Rayner, Katherine Davison, Vadim Bulitko, Kenneth Anderson, and Jieshan Liu. Real-time heuristic search with a priority queue. In *Proceedings of IJCAI-07*, pages 2372–2377, 2007.

[133] Stephen G. Ritchie and Baher Abdulhai. Development, testing and evaluation of advanced techniques for freeway incident detection. Technical Report UCB-ITS-PWP-97-22, California Partners for Advanced Transit and Highways (PATH), 1997.

[134] G. Rose and H. Dia. Freeway automatic incident detection using artificial neural networks. In *Proceedings of the International Conference on Application of New Technology to Transport Systems*, volume 1, pages 123–140, 1995.

[135] M. Rudary, S. Singh, and D. Wingate. Predictive linear-gaussian models of stochastic dynamical systems. In *Uncertainty in Artificial Intelligence*, volume 21, pages 501–508, 2005.

[136] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

[137] Andreas Schadschneider, Thorsten Pöschel, Reinhart Kühne, Michael Schreckenberg, and Dietrich E. Wolf, editors. *Traffic and Granular Flow*, 2005.

[138] A. Schein, L. Saul, and L. Ungar. A generalized linear model for principal component analysis of binary data. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics AISTATS*, 2003.

[139] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[140] Ross Shachter and Robert Kenley. Gaussian influence diagrams. *Management Science*, 35(5):527–550, 1989.

[141] Sajid Siddiqi, Byron Boots, and Geoffrey Gordon. A constraint generation approach to learning stable linear dynamical systems. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1329–1336. MIT Press, Cambridge, MA, 2008.

[142] Tomáš Šingliar. *Machine Learning Tools for Transportation Networks*. PhD thesis, Department of Computer Science, University of Pittsburgh, 2008.

[143] Tomáš Šingliar and Denver H. Dash. Efficient inference in persistent dynamic Bayesian networks. In *Proceedings of the* 24^{th} *Conference on Uncertainty in Artificial Intelligence, UAI-2008*, pages 494–502, 2008.

[144] Tomáš Šingliar and Miloš Hauskrecht. Noisy-or component analysis and its application to link analysis. *Journal of Machine learning Research*, pages 2189–2213, Oct 2006.

[145] Tomáš Šingliar and Miloš Hauskrecht. Towards a learning incident detection system. In *ICML 2006 Workshop on Machine Learning Algorithms for Surveillance and Event Detection*, 2006.

[146] Tomáš Šingliar and Miloš Hauskrecht. Learning to detect adverse traffic events from noisily labeled data. In *Proceedings of Principles and Practice of Knowledge Discovery in Databases PKDD 2007*, number 4702 in LNCS, pages 236–247. Springer, 2007.

[147] Tomáš Šingliar and Miloš Hauskrecht. Modeling highway traffic volumes. In *Proceedings of European Conference on Machine Learning ECML 2007*, number 4701 in LNCS, pages 732–739. Springer, 2007.

[148] Tomáš Šingliar and Miloš Hauskrecht. Approximation strategies for routing in dynamic stochastic networks. In *Proceedings of the* 10^{th} *International Symposium on Artificial Intelligence and Mathematics—ISAIM 08*, page electronic collection, January 2008.

[149] WWW site. http://transims.tsasa.lanl.gov/.

[150] WWW site. http://www.aimsun.com.

[151] WWW site. http://www.paramics.com.

[152] B.L. Smith, B.M. Williams, and R.K. Oswald. Parametric and nonparametric traffic volume forecasting. In *Proceedings of Transportation Research Board Annual Meeting*, Washington, DC, 2000. Transportation Research Board.

[153] Alvaro Soto. Self adaptive particle filter. In *IJCAI*, pages 1398–1406, 2005.

[154] Anthony Stentz. The focussed D^* algorithm for real-time replanning. In *In Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1652–1659, 1995.

[155] Y.J. Stephanedes and J. Hourdakis. Transferability of freeway incident detection algorithms. Technical Report Transportation Research Record 1554, Transportation Research Board, National Research Council, 1996.

[156] Yorgos J. Stephanedes and A. P. Chassiakos. Application of filtering techniques for incident detection. *Journal of Transportation Engineering*, pages 13–26, 1993.

[157] D. Stoyan. *Comparison Methods for Queues and Other Stochastic Models*. John Wiley and Sons, 1983.

[158] Shiliang Sun, Changshui Zhang, and Guoqiang Yu. A Bayesian network approach to traffic flow forecasting. In *IEEE Transactions on Intelligent Transportation Systems*, volume 7, pages 124 – 132, March 2006.

[159] Shiliang Sun, Changshui Zhang, Guoqiang Yu, Naijiang Lu, and Fei Xiao. Bayesian network methods for traffic flow forecasting with incomplete data. In *Proceedings of ECML 2004*, Lecture Notes in Computer Science, pages 419–428, 2004.

[160] Paul A. Taylor. Analysis and synthesis of intonation using the Tilt model. *Journal of the Acoustical Society of America*, 107(3):1697–1714, 2000.

[161] Bo Thiesson, Christopher Meek, David Maxwell Chickering, and David Heckerman. Learning mixtures of Bayesian networks. Technical Report MSR-TR-97-30, Microsoft Research, 1998.

[162] M. Tipping and C. Bishop. Probabilistic principal component analysis. Technical Report NCRG/97/010, Neural Computing Research Group, Aston University, September 1997, 1997.

[163] Transportation Research Board. Highway capacity manual. Technical report, Transportation Research Board, 1997.

[164] Transportation Research Board. Highway capacity manual. Technical report, Transportation Research Board, 2000.

[165] Eleni I. Vlahogianni, John C. Golias, and Mathhew G. Karlaftis. Short-term traffic forecasting: Overview of objectives and methods. *Transport Reviews*, 24(5):533–557, September 2004.

[166] R. Weil, J. wooton, and A. Garcia-Ortiz. Traffic incident detection: Sensors and algorithms. *Mathematical and Computer Modelling*, 27(9-11):257–291, 1998.

[167] Michael P. Wellman, Matthew Ford, and Kenneth Larson. Path planning under time-dependent uncertainty. In *Proceedings of UAI-95*, pages 532–539, 1995.

[168] David Wingate and Satinder Singh. Kernel predictive linear gaussian models for nonlinear stochastic dynamical systems. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 1017–1024, New York, NY, USA, 2006. ACM.

[169] Kun Zhang and Michael A. P. Taylor. Towards universal freeway incident detection algorithms. *Transportation research. Part C, Emerging technologies*, 14(2):68–80, 2006.

[170] Rong Zhou and Eric Hansen. Multiple sequence alignment using Anytime A^*. In *Proceedings of AAAI-02*, pages 975–976, 2002.

[171] Henk J. Van Zuylen and Luis G. Willumsen. The most likely trip matrix estimated from traffic counts. *Transportation Research Part B: Methodological*, 14:281–293, September 1980.