technische universität
dortmund

Bachelor's Thesis

Dynamic Traffic Routing with Bandit Feedback Learning

Maurice Sotzny
October 2016

Reviewer:
Prof. Dr. Katharina Morik
Dr. Thomas Liebig

Technical University Dortmund
Faculty of Computer Science
Department of Artificial Intelligence
http://www-ai.cs.tu-dortmund.de

Contents

1 Introduction **1**

2 Related Work **3**

3 Principles **5**

3.1 Traffic Sensors 5

3.2 Traffic Flow Modeling 6

3.3 Road Network Modeling 8

3.4 Reinforcement Learning 10

4 Dynamic Traffic Routing **13**

4.1 Dynamic Nature of Road Networks 13

4.2 Dynamic Routing 14

4.3 Policy Optimizer for Exponential Models 15

4.4 Dynamic Routing with POEM 16

4.4.1 Prerequisites 16

4.4.2 Application 17

4.4.3 Logging 18

5 Evaluation **21**

5.1 Evaluation Environment 21

5.2 Evaluation Metrics 22

5.3 Experiment — Complete Information 23

5.4 Experiment — Reduced Sensor Locations 30

5.5 Experiment — Reduced Sensor Information 32

5.6 Experiment — Mobile Sensors 35

5.7 Experiment — Reapplication 38

6 Conclusion **41**

Chapter 1

Introduction

Traffic congestion is a globally occurring problem [2]. While more network capacity could potentially decrease congestion, it is also one of the most expensive countermeasures [3]. Additionally, increasing the network capacity is not always possible as metropolitan areas usually provide limited space for new infrastructure. Therefore, current research aims at decreasing congestion using more cost effective methods. Various approaches are currently being researched.

A common approach to reduce congestion is a modal shift policy [3]. It aims at decreasing driving demand by providing easy access to alternative means of transportation. The prime example is mass transit, which larger urban areas continue to develop [11, 21]. Additionally, bicycles have been advocated early on as a green, healthy alternative for short distances [3]. This initiative aims at making popular roads more accessible to cyclists as well as providing more cycling infrastructure in urban centers. More recently car sharing has also been gaining popularity [22]. It owes its success mostly to on-line car sharing communities, which make on-demand access to vehicles easily available.

In contrast, this thesis explores dynamic traffic routing as a less invasive method to decrease overall network congestion. The central idea is to use a reinforcement learning algorithm, namely *Policy Optimizer for Exponential Models* (POEM) [38], to detect congested areas in road networks. The results computed by POEM will then be used to provide live traffic updates. This should enable road users to bypass congestion at an early stage, which in turn should distribute traffic more evenly and reduce overall network congestion. As a result individual travel time should decrease for most road users. However, an advantage for some road users is expected to be at the expense of others.

Implementing such a system inherently requires exhaustive live road network data, collected by a multitude of sensors. For example, *Sydney Coordinated Adaptive Traffic System* (SCATS) is a proprietary system which integrates stationary sensors in existing road networks. The SCATS system was already implemented in many urban areas such as Hong Kong, Shanghai or Dublin [29]. However, widespread coverage using stationary sensors is

costly and rarely available. A more attractive, software-based solution could collect information using smartphone applications. Frequently cited examples are TomTom [1] and Google Maps [8]. Finally, information must be made available to the road users. Here, navigation devices without an Internet connection can use the *Traffic Messaging Channel* [23], which is a one-way radio interface.

Existing approaches and related work are presented in chapter 2. The principles regarding general reinforcement learning and traffic flow theory are outlined in chapter 3. Then, POEM is applied to congestion detection and vehicle routing in chapter 4. The proposed application is evaluated in chapter 5. Finally, a conclusion is drawn and prospective research is described in chapter 6.

Chapter 2

Related Work

The worldwide levels of congestion continue to rise every year, which is why research on its avoidance remains an active area of research. This chapter outlines a selection of relevant approaches.

The paper [28] introduces *r-Extreme Signaling*, an interval scheduling scheme which counters congestion by distributing vehicles more evenly. In *r-Extreme Signaling* road cost information is provided in intervals as opposed to scalars. Then vehicles individually chose scalar costs from the interval with respect to a predetermined policy. Each unique policy will provide a distinct view on the road network. This consequently leads to a more even vehicle distribution. In contrast to dynamic routing as presented in section 4.2 vehicles will not perform dynamic congestion avoidance, but rather remain on routes set at departure. This may be a caveat of *r-Extreme Signaling*, as drivers will occasionally be uncooperative and autonomously choose to bypass jams. However, a system without sudden reroutes may put less stress on drivers in real-world scenarios.

In [15] CHIMERA is presented as a novel *Intelligent Transportation System* (ITS) [17]. The proposed system installs multiple road side units (RSU) in a road network to monitor its current state. An RSU itself is not a sensor, but rather receives and processes measurements made by in-vehicle sensors currently located in its assigned area of influence. The measurements determine a road's density and mean speed, which are introduced in section 3.2. Those are used in a *k-Nearest-Neighbor* algorithm to predict congested roads. Finally, vehicles likely to be caught in a jam are selectively rerouted. The evaluation of CHIMERA uses *Simulation for Urban Mobility* (SUMO) [24]. This simulator is also used to evaluate dynamic routing with Policy Optimizer for Exponential Models (POEM) [38] in chapter 5. Although no direct comparison is made section 5.5 also evaluates dynamic routing with POEM using solely vehicle density and mean speed measurements.

The paper [27] presents Themis as a participatory smartphone navigation system. The users of Themis are routed evenly through a road network by using a proactive routing algorithm [33]. This counters the negative effects of selfish routing in which every vehicle

choses the shortest path [35]. Additionally, all users regularly send position updates to Themis. Those are used in a logistic regression model [5] which estimates the current number of vehicles on each road. This enables Themis to provide more accurate route suggestions to users which bypass congested areas. The main difference between Themis and dynamic routing with POEM is the employed learning method. Themis estimates current vehicle counts using a supervised learning algorithm. In contrast, POEM is a reinforcement learning algorithm and is used in section 4.4 to directly classify a road as either congested or uncongested.

The game-theoretic Nash equilibrium [31] can also be applied to vehicles in a road network. It is a local optimum in which no vehicle can gain a time-wise benefit by solely changing its own route. However, it is not necessarily a global optimum as changing routes of more than one vehicle may result in better overall network performance. The Nash equilibrium cannot be applied to real-world scenarios, as its calculation requires complete prior knowledge about driving demand. It is also unable to cope with unpredictable events such as accidents. Hence, it is only presented as an upper baseline during the evaluation in chapter 5.

Chapter 3

Principles

This chapter provides an overview of the necessary terminology and theories used throughout the thesis. Firstly, an overview of traffic sensors is given in section 3.1. Then, mathematical models of traffic flow and road networks are presented in sections 3.2 and 3.3. Lastly, a short introduction to reinforcement learning is given in section 3.4.

3.1 Traffic Sensors

Data representing the current state of a road network is collected using a wide range of sensors. They can usually be classified as being either single-point or point-to-point.

As the name suggests, a single-point sensor measures data in a single point of a road network. The measured data is usually limited to vehicle count, but more complex sensors are also able to detect vehicle speed or type. The most prevalent single-point sensor is the induction loop. Here, either a single or double wire loop installed inside the pavement is used to detect electromagnetic distortions caused by passing vehicles. The main advantage of the more expensive double loop variant is its capability of reliably measuring vehicle speeds. However, measurements by single loop sensors can be used to estimate vehicle speeds [32].

More recently, point-to-point sensors have been gaining relevance. One goal is to create source-target distributions by monitoring individual vehicles at two or more points in a road network. However, smartphones as point-to-point sensors allow more exhaustive data to be collected, as they natively provide positioning systems and routing applications [8, 10]. The measurements can then either be streamed live over a mobile network or uploaded later, when a Wi-Fi connection is available. Nevertheless, due to its accuracy, personalization and invasiveness, this information is without doubt very sensitive and must be collected defensively.

The sensor data in combination with static road network data is then used to create various empirical models describing the state of a road network.

3.2 Traffic Flow Modeling

In order to improve road networks and routing services, a mathematical model of real-world traffic is required. The complexity of this task is created by the large number of independent actors in a road network, such as vehicles, pedestrians or cyclists. Hence, all models provide merely an approximation by making various assumptions to simplify the relations between the actors.

Most models usually take either a micro- or macroscopic approach. A microscopic model is more intricate as it describes the state of a road network by modeling each individual vehicle and their interactions with others. In contrast, a macroscopic model only describes an aggregated state of road network components. Consequently, a sufficiently good microscopic model will always outperform any macroscopic model, albeit at much greater computational costs. Additionally, real-world microscopic datasets are very large in size. For example, microscopic in-vehicle sensors will continuously store vehicle positions and speeds. This is manageable when selected vehicles are monitored but does not scale well for larger sample sizes. Not only does processing of microscopic data require a lot of coordination but it also raises concerns with regard to privacy [25]. This is why microscopic data is usually aggregated to create a generalized, privacy preserving macroscopic model.

According to [18] a spatial model must satisfy two requirements. Firstly, it must be related to real-world scenarios. Secondly, it must be possible to move between micro- and macroscopic models without loss of critical information. The presented macroscopic model conforms with those requirements as it is based on aggregated, microscopic real-world sensor measurements.

Macroscopic Model

Macroscopic variables are used to represent an aggregated state of a road network [20]. Most commonly, a spatio-temporal average or sum is modeled, where a variable describes a cross section of road network over a discrete period of time. Here, cross section refers to a stretch of a road or lane, rather than a single point.

The flow or vehicle count $q := n * T^{-1}$ is defined as the number of vehicles n that pass a cross section over a discrete time period T. This variable can be measured using single-point sensors. Next density $k := n * X^{-1}$ is defined as the number of vehicles n per distance unit X. Similarly, occupancy is defined as the percentage of a cross section covered by vehicles. It will generally stay well below 100% as drivers will always keep a minimum distance to others. Another macroscopic variable is the mean vehicle speed u on a cross section. In order to reliably measure density, occupancy or vehicle mean speed, point-to-point sensors are required. However, measurements by single-point sensors can be extrapolated over a cross section with an inherent loss of accuracy. Finally, waiting time is defined as the absolute time all vehicles wait on a cross section over a period of time.

It is rarely measured using conventional sensors as it requires sophisticated point-to-point sensors. This should not be confused with waiting time in a single point, which is usually measured with induction loops or cameras and used to control signals.

Congestion Detection

Traffic congestion is usually characterized by slow, inconsistent vehicle speeds and occurs whenever road network demand is greater than its capacity. This section presents two primitive methods to quickly indicate whether or not a road might be congested.

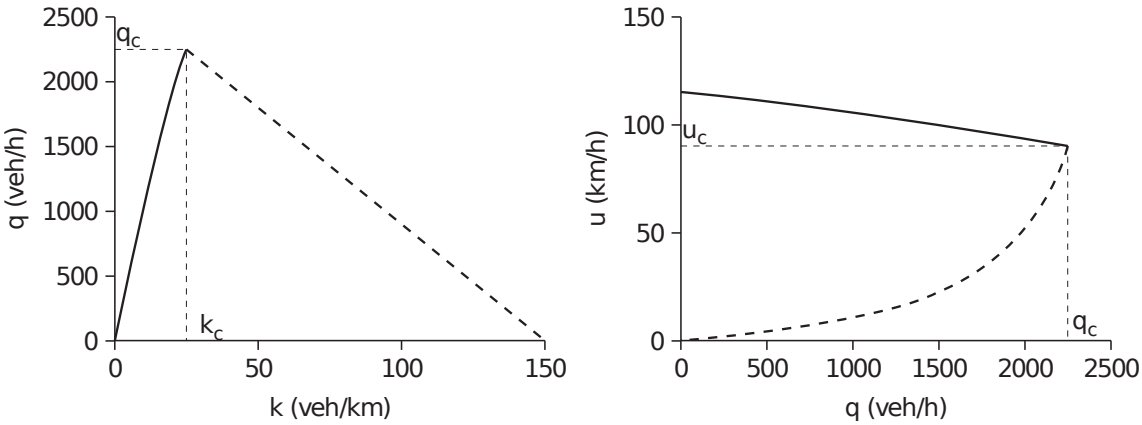

Figure 3.1: The graphs show exemplary relations between flow, density and speed in a macroscopic model. The exact numbers will depend on various variables such as road conditions, speed limits or weather. Source: *Traffic Flow Theory and Modelling* [20]

In macroscopic models flow, density and mean speed are related. Those relations are derived from empirical data in [20] and presented in figure 3.1. When a vehicle enters a relatively empty road, it can generally drive at maximum allowed speed without influencing others. The density of a road will inherently increase when more vehicles enter it. This continues up to a critical density k_c, where a road will operate at its maximum capacity q_c. At critical density each vehicle is still able to drive at maximum speed, but actions by drivers such as breaking or speeding could easily lead to jams as other drivers will react accordingly. A density value above k_c means vehicles cannot keep sufficient safety distances to each other anymore. Hence, some drivers will chose to drive at slower speeds. This reduces necessary safety distances, which in turn allows more vehicles to enter. This continues until a jam density k_j at which each vehicle stops is reached [20].

The exact values of k_j and k_c depend on static and dynamic variables such as lane count, road condition or weather. However, rough estimations about k_j and k_c can be made. At k_j vehicles will not move and be relatively close to each other. Hence, an intuitive estimation of k_j^e for a road e would use its lane count LC_e, average vehicle length

veh_l (m) and minimum safety distance veh_s (m). Then, an estimation $\tilde{k}_c^e$ of k_c^e can be made, as [13] states that it is generally one seventh of the jam density:

$$\tilde{k}_c^e := \frac{1}{7} * 1000 * LC_e * (veh_l + veh_s)^{-1} \tag{3.1}$$

The primitive density congestion detection method $\delta_{density}$ labels a road as congested when its density is larger than its estimated critical density. Additionally, slow vehicle speeds can also indicate congestion. Similar methods were studied in [36] and are implemented in [8]. The primitive mean speed congestion detection method δ_{speed} labels a road as congested when its mean speed is below $10km/h$ of the respective speed limit. This threshold parameter was chosen due to its overall good performance in simulations.

3.3 Road Network Modeling

This section describes how road networks can be modeled as graphs. The subsequently used shortest path algorithm A* is presented as well.

A road network can be modeled by a directed graph $G = (V, E, c)$. The nodes $v \in V$ represent junctions, which are located in a 3-dimensional Euclidean vector space. For notational convenience, let $v \in V$ not only represent a node in V, but also its Euclidean coordinates. This allows calculation of the direct distance between nodes $u, v \in V$ using the Euclidean norm as shown in equation 3.5. Next, a road connecting nodes $u, v \in V$ is represented by a directed edge $(u, v) \in E \subset \{(p, q) \mid p, q \in V,\ p \neq q\}$. An edge $e \in E$ is associated with its positive length $l : E \to \mathbb{R}_+^*$ and its positive speed limit $s : E \to \mathbb{R}_+^*$. Finally, $c : E \to \mathbb{R}_+$ maps a nonnegative cost value to each edge. A simple example for c would be l. This can be improved by additionally using the provided speed limit to calculate the minimum time it would take to pass an edge, which yields:

$$c(e) := l(e) * s(e)^{-1} \tag{3.2}$$

This equation assumes that every vehicle is capable of driving the maximum speed limit. In reality, each vehicle must also consider its own maximum speed.

This graph representation of a road network can now be used to calculate a path between two nodes with minimum costs, which is commonly known as the single-source shortest path problem. All subsequent sections will use *minimum cost* and *shortest distance* interchangeably. The next section presents, amongst other algorithms, A* as an efficient solution.

Route Calculation with A*

The single-source shortest path problem is about finding a path between two nodes $s, t \in V$ with minimum costs in a graph $G = (V, E, c)$. Here, Dijkstra's algorithm [16] is the most established solution when dealing with weighted graphs [9].

Dijkstra's algorithm maintains a priority queue Q of nodes. Here, nodes are sorted by currently known shortest path distances to the starting node s. The algorithm starts by initializing all distances to ∞, excluding s, which is initialized to 0. Then, a loop will iteratively remove node u, also known as expanding node u, with shortest distance to s from Q. Having removed node u implies a shortest path between nodes s and u was discovered. Finally, in each iteration Dijkstra's algorithm inspects each node v adjacent to u, $v : (u, v) \in E$, and updates Q when a shorter path from s to v via u exists. The algorithm stops when Q is empty or a shortest path between nodes s and t has been discovered.

The A* algorithm improves Dijkstra's algorithm with respect to computational complexity. This is achieved by ideally minimizing the search space by processing nodes closer to the target node in an earlier iteration. To do so A* additionally considers the actual shortest distance between t and other nodes when ordering Q. However, since such values are generally unknown a heuristic $\pi_t : V \to \mathbb{R}_+$ must be used to provide estimations. In A* heuristics are usually characterized by two properties. It is said to be *admissible* when it is a lower bound on the actual shortest distance. In other words π_t is admissible if and only if it does not overestimate the distance between two nodes:

$$\forall v \in V : \pi_t(v) \leqslant \pi_t^{'}(v) \tag{3.3}$$

where $\pi_t^{'}(v)$ denotes the actual shortest distance between nodes v and t. Furthermore, it is said to be *consistent* when it satisfies a triangle inequality:

$$\forall (u, v) \in E : \pi_t(u) \leqslant c((u, v)) + \pi_t(v) \tag{3.4}$$

When a heuristic is admissible and consistent, no further alterations need to be made to Dijkstra's algorithm other than updating Q by additionally adding $\pi_t(v)$ to every node $v \in Q$. Hence, using $\pi_t : V \to \{0\}$ is equivalent to Dijkstra's algorithm, whereas using $\pi_t = \pi_t^{'}$ will only consider edges on the shortest path [9]. In other words estimation performance of π_t will directly influence the decrease in computational complexity.

However, when a heuristic is admissible and inconsistent, a more complex implementation must be chosen to guarantee optimal results. The reason is that without consistency, removing a node from Q does not always imply that a shortest path to it was discovered. In such instances nodes need to be reinserted to Q when a shorter path is discovered during node inspection.

When a heuristic is inadmissible A* cannot guarantee optimal results. The algorithm may prune the search space too much and expand node t before it can verify that there are no shorter paths over adjacent nodes. Lastly, it should be noted that consistency implies admissibility [19].

The heuristic which was used throughout the thesis, shown in equation 3.5, uses the Euclidean distance between two nodes and divides it by the maximum speed limit in the road network. More sophisticated approaches are presented in [9].

$$\pi_t(v) = \|v - t\| * {s_{max}}^{-1},\ s_{max} := \max_{e \in E} s(e) \tag{3.5}$$

where $\|v - t\|$ denotes the Euclidean distance between nodes v and t.

3.4 Reinforcement Learning

The domain of machine learning deals with making predictions about data without a set of static rules but rather by examination of data. The domain is generally subdivided in three categories: supervised-, reinforcement- and unsupervised learning. Firstly, supervised learning algorithms learn using a set of fully labeled examples. In other words correct predictions for a subset of data must be available. Secondly, reinforcement learning algorithms learn by interactions with an environment. The approach closely resembles the natural acquisition of intelligence through experiments [37]. Thirdly, unsupervised learning algorithms explore structures in datasets without prior knowledge about existence of such structures. This section outlines general reinforcement learning.

Figure 3.2 shows a sketch of the main components and interactions in a reinforcement learning setting. The environment is characterized by its current state and a set of possible actions. Additionally, each action is assigned an immediate reward value depending on the current state. The agent represents the learning entity. It moves through the environment by iteratively selecting actions according to its internal policy and thereby collecting rewards. The goal of an agent is to maximize its collected reward. However, selecting an action will cause the environment to advance into a new state, which is determined by the selected action itself rather than its reward. This is the identifying characteristic of reinforcement learning. The rewards are not instructive, meaning it alone does not decide which action to take. Only a combination of environmental state, action and reward can

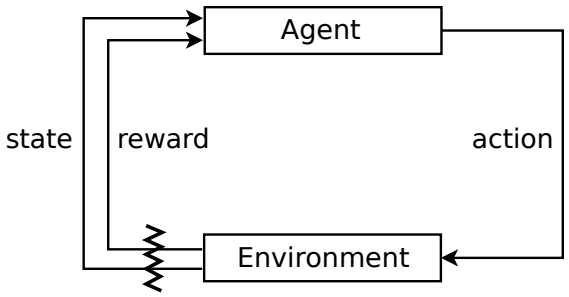

Figure 3.2: The interactions between an agent and its environment in a reinforcement learning setting.

be used to make a prediction about a prospective series of actions [37]. Therefore, in order to solve a reinforcement learning problem, a policy must not only exploit actions with good immediate rewards, but also explore other actions which may lead to a greater long-term reward.

This problem is known as exploration versus exploitation and is a deciding factor in the success of reinforcement learning algorithms. In order to gain good long-term rewards a policy must explore the search space by examining a set of varying actions. The expected rewards should not be considered during exploration, since areas of good long-term rewards may only be accessible through actions with low immediate rewards. When a search space is sufficiently explored, an algorithm will begin exploiting actions which are known to lead to a good overall reward. This is achieved by predicting such actions based on the current environment state.

The rewards provided by the environment actions are not always known in advance. In other words a reward is only assigned to an action after it was chosen. This is known as a multi-armed bandit problem [7, 37]. This subclass of reinforcement learning problems is especially challenging since a solution must additionally estimate the expected rewards for all available actions in an environment state.

Chapter 4

Dynamic Traffic Routing

This chapter discusses the challenges of routing in road networks. The concept of dynamic routing presents a possible solution. It is additionally combined with a congestion detection method based on *Policy Optimizer for Exponential Models* (POEM) [38].

4.1 Dynamic Nature of Road Networks

In many cases road networks are exceedingly dynamic structures.

Firstly, the topology of a road network is evolving over time. Infrastructural changes are usually planned years in advance and do not pose a challenge to map providers. The problem lies in unpredictable changes, such as minor roadworks or accidents possibly resulting in lane or junction closures. When calculating a route, those situations should be considered as soon as possible. This simplifies bypassing the situation as well as the congestion arising in its close proximity.

Secondly, even though the overall flow of traffic usually follows a predictable pattern, localized situations are often too volatile to predict, making them onerous to include in regular route planning. For example, sporting events or concerts greatly increase local traffic over a short, irregular period of time. To maximize flow, a routing system should evenly distribute vehicles involved in such scenarios, while all others should be routed around the area.

Thirdly, road networks are beginning to be more dynamic by design. High demand areas are starting to implement digital signaling and signage [12]. In those cases sensors monitor the current flow of traffic and adjust signals and signs accordingly. The central idea of research presented in this paper is to additionally utilize such live sensor data in traffic routing. The goal is to increase the performance of routing systems in the described scenarios.

These dynamic properties pose great challenges to traffic routing. Therefore, in order to make adequate route suggestions, navigation systems must utilize live road network information.

4.2 Dynamic Routing

The idea of dynamic routing is to incorporate live sensor data in route calculation, mainly to bypass roads which are currently congested. This information is usually publicly broadcast by a central authority using the *Traffic Messaging Channel* [23]. This system would work best, if only a limited number of vehicles responded to live updates. However, as information is publicly available over radio, a scenario in which many vehicles respond to updates concurrently is not unlikely. This could result in many vehicles making similar route alterations, which would simply relocate congested areas. Hence, overall network congestion would not decrease. Therefore, methods which additionally aim at evenly distributing traffic need to be researched.

Scarce, irregular road network information updates alone are not effectively countering congestion [28]. Fast paced, frequent updates in contrast could distribute traffic more evenly, so much that individual travel time will decrease on average. The idea is that when many vehicles are aware of the live network state, congested areas can be bypassed in a more even manner. The scenarios are outlined by the following examples.

Assume a central authority sends updates to all vehicles whenever the state of a road changes from normal to severely congested or vice versa. Assume most vehicles which receive updates about congested roads en route will calculate new routes simultaneously in a more or less greedy manner. The likely result is that most alternative routes will bypass the congested roads using similar diversions. Especially vehicles close to the affected roads will usually only make minimal route changes. This could, in some cases, even give rise to more congestion, particularly when alternative roads in close proximity have lower capacities.

Alternatively, a central authority could send updates in short intervals. The update information would include live data about some or all roads in a network. When an update is received, a vehicle will recalculate its route using the new information. This scenario should allow vehicles to distribute more evenly throughout the whole network. Moreover, vehicles farther away would bypass not only the congested road, but also all congestion inevitably arising around it. This method should also perform well with respect to short-lived, unpredictable jams. Vehicles could also adapt to changes in signaling or signage within a matter of seconds or minutes.

In order to pursue this alternative approach, an encompassing network of sensors is obligatory. A sensor would measure various numeric parameters which describe a specific cross section of the road network. Common examples are a road's occupancy, density, mean

speed or vehicle count. However, sensors can only provide numeric values representing the current state. Additionally, the application of a machine learning algorithm would allow users to easily process sensor readings by predicting a concrete, prospective label. For example, such a label could indicate whether a road will be congested or not. The next section will introduce POEM as a suitable machine learning algorithm.

4.3 Policy Optimizer for Exponential Models

The *Policy Optimizer for Exponential Models* (POEM) [38] is a reinforcement learning algorithm for structured output prediction. More specifically, POEM learns using logged bandit feedback, meaning no interactive control over an environment is required. This section outlines the application of POEM to single label classification. The goal of POEM is to optimize an existing policy. In other words a system which makes predictions about an input space X according to a policy h_0 must already be implemented. Then, POEM will utilize data logged by the existing system to optimize h_0.

More specifically, let $X \subset \mathbb{R}^m,\ m \in \mathbb{N}$ be the input space and $Y \subset \{(0), (1)\}$ be the output space. The input space X represents all possible input variables. The output space Y represents a single label which is either assigned (1) or not assigned (0) to an input variable $\mathbf{x} \in X$. The existing policy $h_0(Y \mid \mathbf{x})$ is a probability distribution over the output space and predictions are made by sampling $y \sim h_0(Y \mid \mathbf{x})$. For notational convenience let $h_0(\mathbf{y} \mid \mathbf{x})$ denote the probability assigned to label $\mathbf{y}$ for input $\mathbf{x}$. Since h_0 is already implemented in an active system, information about the quality of the sampled outputs can be measured. However, information about other outputs remains unknown. Therefore, let $\delta : X \times Y \to \mathbb{R}$ denote a cardinal loss feedback for all observed pairs $(\mathbf{x}, \mathbf{y})$, where smaller values indicate greater satisfaction with $\mathbf{y}$ for $\mathbf{x}$. In order to improve h_0, POEM requires a dataset:

$$D := \{(\mathbf{x}_i, \mathbf{y}_i, \delta_i, p_i) \ \mid i \in \mathbb{N}_{\leqslant n}\},\ \delta_i = \delta((\mathbf{x}_i, \mathbf{y}_i)),\ p_i = h_0(\mathbf{y}_i \mid \mathbf{x}_i) \tag{4.1}$$

Then, POEM will explore the hypotheses space $\mathcal{H}_{lin}$ for a policy $h_{\mathbf{w}}$ that maximizes user satisfaction by using gradient descent on the estimated, expected loss as well as its empirical standard deviation. For single label classification a policy $h_{\mathbf{w}} \in \mathcal{H}_{lin}$ will sample $\mathbf{y}$ according to:

$$h_{\mathbf{w}}(\mathbf{y}|\mathbf{x}) = exp(y_1 * \mathbf{w}^T\mathbf{x}) * (1 + exp(\mathbf{w}^T\mathbf{x}))^{-1},\ \mathbf{w} \in \mathbb{R}^m \tag{4.2}$$

where $\mathbf{w}$ denotes the vector learned by POEM and y_1 denotes the single label stored in vector $\mathbf{y}$. A complete description and discussion of POEM as well as pseudo-code is provided in [38]. The subsequent section will apply POEM to the problem of congestion detection in road networks.

4.4 Dynamic Routing with POEM

Traffic closely resembles a reinforcement learning environment. Vehicles serve as agents which move in a road network. Actions are represented by the roads a vehicle can chose at each intersection. Once a road is chosen, a reward will be assigned for that particular road depending on its actual state. The rewards for all other roads which could have been chosen instead remain unknown. This scenario is known as a multi-armed bandit problem [6]. The lack of fully labeled data makes any supervised learning approach particularly complex.

With this in mind POEM [38] was chosen as a suitable learning algorithm. Firstly, POEM solely requires the reward values provided by the environment to learn a model. Additionally, POEM does not require interactive control over agents, but rather learns using logged data. This approach is more robust, as learned models can be evaluated before deployment. Furthermore, a model will not evolve over time, which otherwise might lead to unpredictable behavior.

4.4.1 Prerequisites

The approach of POEM is to improve an already existent policy h_0. It assigns a structured output to an arbitrary input based on its probability of being correct. Therefore, before applying POEM to congestion detection, a suitable policy h_0, along with an input space X and output space Y must be constructed. Additionally, a cardinal loss feedback mapping δ, which provides a quality measure about all observed input/output combinations, is required.

The input space need not fulfill any special characteristics. However, learning can be performed more efficiently when its variables are scaled appropriately. Therefore, let $X := [0,1]^m,\ m \in \mathbb{N}$, where each $\mathbf{x} = (x_1, \ldots, x_m)^T \in X$ represents a feature vector of sensor measurements for a sensor. For example, a road's density, occupancy, mean speed, vehicle count or waiting time can be used. All variables need to be scaled, which can be achieved by a simple min-max scaling. It is important to note that the size of aggregated values will additionally depend on the size of the collection interval.

The output space must be a set of suitable, structured outputs. As POEM should be applied to the problem of congestion control, a single label indicating whether or not a road is congested could already provide adequate results. Thus, let $Y := \{(0), (1)\}$, where label (0) indicates a road is not congested. A single label gives no indication about how severe individual jams are. Therefore, vehicles are expected to bypass all congestion in an equal manner. This could allow smaller jams to dissipate quickly, before longer ones evolve. Those scenarios should be particularly visible in combination with dynamic routing. As a consequence, a road network should also be utilized more evenly. This could result in averagely longer travel distances. Nevertheless, many users should ideally have a time-wise advantage.

The policy $h_0(Y \mid \mathbf{x})$ is a probability distribution over the output space. In other words it assigns a probability to each input/output combination based on how likely it is to be correct. The goal of POEM is then to improve upon this policy. However, no such policy exists for the constructed input and output spaces. This is a common problem when applying POEM. Therefore, a default policy is used. Hence, let $h_0(\mathbf{y} \mid \mathbf{x}) := 0.5$. For any input it assigns a probability of 0.5 to both possible outputs; consequently both initially are equally likely to be correct.

Lastly, in order to improve an existing policy POEM requires a cardinal loss feedback mapping $\delta : X \times Y \to \mathbb{R}$. For congestion, multiple metrics which measure its severeness are available. Section 3.2 presents two primitive metrics which utilize mean speed or density values. Their main advantage is that those parameters are already provided by sensors. In other words no supplementary data collection is required. As discussed in [38], proper scaling of the feedback is necessary. This is achieved by introducing simple threshold values, which reduce the value space: $\delta : X \times Y \to \{-1, 0\}$.

The primitive density congestion metric, $\delta_{density}$, would assume a road as congested when its density was greater than one seventh of its jam density [13]. The primitive mean speed congestion metric, δ_{speed} would assume a road as congested when its mean speed was less than ten kilometers per hour of its allowed speed.

4.4.2 Application

The goal is to not only detect congestion, but also to avoid it as soon as it arises. Hence, a policy is directly combined with the previously introduced idea of dynamic routing. In the following section only one of many possible approaches will be presented.

Let $G = ((V, E, c), \pi)$ be a graph which represents a road network as described in section 3.3. Here, c and π are the default weight and heuristic functions. Additionally, assume all vehicles have knowledge about a congestion labeling policy $h \in \mathcal{H}_{lin} \cup \{h_0\}$. When using dynamic routing, vehicles will receive updates about roads in regular intervals $T \in \mathbb{N}$. The update can then be written as $u_T : E \mapsto X$.[1] The value space of u_T is equal to the input space of POEM. Hence, when a vehicle receives an update u_T, it is able to predict how likely a road is to be congested during interval $T + 1$ using h. This is why smaller update intervals are expected to outperform larger ones.

The described model receives sensor information about whole roads only. It does not receive information about individual lanes which might be problematic as congestion does not always arise on every lane equally. The scenario is most likely at junctions, where each lane will allow a vehicle to go in a different direction. This problem is addressed by aggregating sensor data for each connected edge pair. That way congestion on diverging lanes can be distinguished more accurately.

[1] It is assumed updates are received equally for all edges.

In order to bypass arising congestion, a vehicle must recalculate its route with respect to the newly received update u_T. This is achieved by increasing the weight of an edge when it is likely to be congested:

$$c' : E^2 \to \mathbb{R}_+,\ (e_1, e_2) \mapsto \frac{c(e_2)}{h((0) \mid 0.5u_T(e_1) + 0.5u_T(e_2))} \tag{4.3}$$

The denominator shows the previously mentioned aggregation of sensor data. For notational simplicity, c' is defined for all of E^2. In practice only a subset of E^2 in which e_1 incident or equal to e_2 is used.

The function c' calculates the new weight of an edge e_2 depending on which incident edge it was reached by. For instance, a vehicle on an edge $e_1 = (u, v)$ would calculate the weight for edge $e_2 = (v, w)$ using $c'(e_1, e_2)$. A vehicle which started its route on edge e_2 would use $c'(e_2, e_2)$, as no edge prior to e_2 exists.

Essentially, c' divides the old weight of an edge by its probability of not being congested in interval $T + 1$. That means the weight of an edge will remain almost unchanged when no congestion is expected. The increase will conversely depend on how likely congestion is to arise. The soundness is given, as a labeling policy $h \in \mathcal{H}_{lin} \cup \{h_0\}$ will never produce a probability of zero, which can be seen in section 4.3. However, rounding errors may occur in practice. Additionally, as edge weights are solely increased, an existing admissibility contract in π will not be violated, whereas no guarantee with respect to consistency can be made. Consequently, an A* implementation will continue to always produce optimal results. Nevertheless, increasing edge weights may negatively affect its performance.

Finally, it assumed sensor data updates are available for every road. In reality permanently installed sensors are much more scarcely distributed throughout the network. This problem can be partly alleviated by directly implementing sensors in vehicles. For example, one method could use navigation applications on smartphones. However, some roads will still remain uncovered. In that case u_T can map to $\{0\}^m$. This will cause h to assign a probability of 0.5 to both labels. A more sophisticated solution could make predictions about uncovered roads using sensors in close proximity [26]. The experiments evaluated only the first method, as it is very easy to implement and performed well.

4.4.3 Logging

For POEM no interactive control over actions is required as it was specifically designed to learn using logged data. So, with respect to the previously defined setting it requires a dataset of size n:

$$D := \{(\mathbf{x}_i, \mathbf{y}_i, \delta_i, p_i) \mid i \in \mathbb{N}_{\leqslant n}\},\ p_i = h(\mathbf{y}_i \mid \mathbf{x}_i) \tag{4.4}$$

This dataset will be created during the logging phase. All edges are assigned weights using c' and routes are calculated using an implementation of A*, which produces optimal

results for any admissible heuristic. Additionally, when POEM is initially applied using the default policy h_0, all weights are scaled equally by a factor of 2. The scaling does not affect A*, meaning no route changes will occur, which in turn simplifies learning on previously collected data.

The data itself can either be collected by each vehicle or a centralized authority monitoring each vehicle. For both approaches a data entry cannot be created before any feedback is available. Thus, intermediate results must be cached.

Firstly, the aggregated feature vector $\mathbf{x}_i$ is logged. The respective label $\mathbf{y}_i$ with its corresponding probability p_i are then determined using:

$$\mathbf{y}_i = \begin{cases} (0),\ h((0) \mid \mathbf{x}_i) > 0.5 \\ (1),\ h((1) \mid \mathbf{x}_i) > 0.5 \\ random((0),(1)),\ otherwise \end{cases} \tag{4.5}$$

where, $random((0),(1))$ means a label is chosen randomly, uniformly distributed. Lastly, the feedback is logged using either $\delta_{density}$ or δ_{speed} as described in section 3.2. The respective results will inherently depend on the previously chosen label.

Chapter 5

Evaluation

This chapter thoroughly evaluates dynamic routing with POEM using simulated worst-case, realistic and best-case scenarios. The experiments evaluate overall network and individual routing performance.

5.1 Evaluation Environment

The evaluation was performed using *Simulation of Urban Mobility* (SUMO) [24], a microscopic, continuous road network simulator developed by the German Institute of Transportation Systems. SUMO implements a network interface (TraCI), which allows other applications to actively control a simulation. However, its rerouting capabilities are limited to predefined routing algorithms. This limitation was bypassed by developing SUMO$_{CA}$, a Java support application which uses TraCI to emulate a central authority in SUMO.

In order to emulate a central authority SUMO$_{CA}$ converts a plain SUMO graph to a directed multi-graph implementation using *jGraph*, an open-source Java graph library. This allows SUMO$_{CA}$ to route vehicles using custom routing algorithms. A simulation is started by creating a connection to a running SUMO server. The edge lengths and speed limits to create equations 3.2 and 3.5 are initialized using TraCI. While a simulation is running, SUMO$_{CA}$ will receive aggregated sensor measurements by SUMO in regular intervals. This information is used to log a dataset as described in section 4.4.3. Additionally, SUMO$_{CA}$ reroutes a user specified subset of vehicles with each update according to a previously set policy. The updates sent by SUMO need not be complete, meaning each update may only cover an arbitrary subset of edges or vehicles.

Although SUMO supports *Open Street Map* networks, a simulation using a raw network import usually performs poorly. The performance issues are primarily caused by missing or wrong road information, overly complex junction models and improper signal management programs. Therefore, most experiments utilize a manually revised map of Luxembourg City, provided by *Luxembourg SUMO Traffic Scenario* (LuST) [14]. The LuST network

graph implements 2373 nodes connected by 5969 edges, spanning a length of 1571 kilometers on an area of 155 square kilometers. Furthermore, LuST also provides a day of modeled private and public transit data.

The experiments start at 7:45 and run over a period of roughly 35 minutes; or exactly 2048 seconds. The reason why this particular window was chosen is that roads generally are more susceptible to congestion during rush hour. Additionally, a size of 2048 seconds allows rerouting intervals to be easily scaled using a factor of two. In order to create more realistic jams on arterial roads, SUMO was set to scale the original demand by 1.3. Unless stated otherwise each experiment will use measurements of density, occupancy, vehicle count, mean speed and waiting time.

Firstly, dynamic routing is evaluated given complete information availability in section 5.3. A sensor is located on each road and will measure all previously mentioned variables. The subsequent experiments in sections 5.4 and 5.5 reduce sensor locations and measurements. This evaluates dynamic routing with POEM in a more realistic setting, since complete information is generally not available. Then, dynamic routing with POEM is evaluated using mobile sensors in section 5.6. The experiment evaluates whether or not implementing dynamic routing using a participatory smartphone navigation system is suitable. Finally, it is evaluated whether or not an improvement can be seen when reapplying POEM to an already optimized policy in section 5.7.

5.2 Evaluation Metrics

This section gives a short introduction into the underlying motivation of some chosen evaluation metrics.

The primary goal is to evaluate the performance of dynamic routing schemes with regard to congestion minimization and overall network utilization. The arithmetic mean edge waiting time gives insight into global network congestion. The distribution of changes in travel time values compared to a lower baseline evaluates whether or not congestion on individual roads actually decreases or just relocates. The overall performance is measured by the absolute throughput, meaning how many vehicles were able to reach their destination in the given time frame. This metric was chosen, because SUMO is able to insert more vehicles into the network when traffic is distributed more evenly. This effect is most noticeable when congestion builds up around areas where vehicles are inserted into the network. The metric is sound as vehicles spawn mostly evenly distributed throughout the network. In LuST, only a relatively small amount of vehicles spawn on a designated set of roads. Those points are located around motorways.

All mentioned metrics purely evaluate network performance without providing any information about routing performance. For instance, an algorithm which decreases congestion by routing many vehicles over excessively long detours is inexpedient. To evaluate

overall routing performance, vehicle-trip related measurements need to be applied. These results are presented by the distributions of travel and loss times. Time loss is defined as the time a vehicle drives below the speed limit. Additionally, direct comparisons between individual vehicles regarding their performance with and without application of the routing algorithm are of special interest. Those metrics give insight into potential sacrifices some road user have to make, such as driving longer detours. The applied metrics evaluate detours regarding route lengths and travel times when compared to a lower baseline.

However, evaluating detours is problematic. Using absolute values might skew results, as long routes will allow long, absolute detours. Conversely, using relative values might skew results as well, as short routes will allow long, relative detours. The following metric is used to minimize this problem.

Definition (Relative-Weighted Detour) *Let $y_A, y_B \in \mathbb{R}_+^*$ be arbitrary measurements of one vehicle when algorithms A and B were applied respectively. Then,*

$$detour_{rw}(y_A, y_B) := sgn(y_A - y_B) * (y_A - y_B)^2 * (y_A + y_B)^{-1} \tag{5.1}$$

will calculate the relative difference, while simultaneously weighting it using the absolute difference. Assuming lower values of y_A, y_B are better, negative values will indicate an advantage of A over B, whereas positive values will indicate the opposite.

The evaluation results are presented using various charts. The x-axis labels a set of evaluated routing schemes. Firstly, a uniform cost search (UCS), in which each vehicle is routed once at departure using the default cost function, is shown as a lower baseline. The results of dynamic routing with POEM using various interval sizes are presented next. Lastly, a Nash equilibrium (NASH) is shown as an upper baseline. The y-axis labels the used evaluation metric. Unless stated otherwise, *vehicle* means numbers of vehicles, whereas other labels will conform with SI units. In order to adequately present distributions, box plots [30] were used. The boxes represent the lower, middle and upper quartiles, whereas whiskers will represent the second and 98th percentiles. The outliers were omitted in the graphs. For detours and differences, positive values will indicate an advantage of UCS, whereas negative values will indicate the opposite.

5.3 Experiment — Complete Information

This experiment will route 100%, 75%, 50% and 25% of vehicles using 100% of available sensor data. In other words, every road is equipped with a sensor comparable to an induction loop. This represents the best case scenario with regard to information availability. The learning of POEM is performed using all available live sensor data, while vehicles are routed using the unaltered demand data. Feedback is created using the primitive density

congestion detection method. This experiment does not specifically evaluate the performance gain of the individual vehicles using dynamic routing. Those results are presented when mobile sensors are evaluated.

Firstly, when solely looking at a penetration rate of 100%, figure 5.1 shows that dynamic routing initially creates more congestion when using larger interval sizes. As described in section 4.2, one likely explanation is that many vehicles simultaneously chose similar diversions and consequently create long jams on byroads. This explanation is especially reasonable when looking at lower penetration rates, where only some vehicles will bypass congestion. It is confirmed by the edge density distribution shown in figure 5.3 and can also be observed when looking at the mean vehicle speed shown in figure 5.5.

The differences in travel times are shown in figure 5.2. An interval size of 1024 seconds increases per edge travel times by well over a minute for some edges. This figure also shows that for smaller interval sizes congestion does not relocate, but rather dissipates. The changes in the lower, middle and upper quartiles remain close to zero. This occurs since most residential roads will neither directly nor indirectly be affected by congestion.

Figure 5.6 shows that vehicles generally spend less time driving below the speed limit when dynamic routing is applied. This could be caused by the lower average waiting times per edge, which are shown in figure 5.4. The distribution of individual travel times shown in figure 5.7 might suggest that road users have no individual advantage of using dynamic routing. However, figure 5.8 indicates the opposite. It shows that the decrease of individual travel times compared to UCS is considerably greater than any increase. The quartiles around zero also imply that most users will not be affected by dynamic routing. Figure 5.9 outlines a notable difference in trip lengths between a Nash equilibrium and dynamic routing with POEM. This most likely occurs because vehicles distribute evenly right from the start in a Nash equilibrium, while dynamic routing can only counter congestion as soon as it arises.

Lastly, figure 5.10 shows dynamic routing does increase the network throughput by up to 40% compared to UCS. Nevertheless, its performance resides well below that of a Nash equilibrium at 52%.

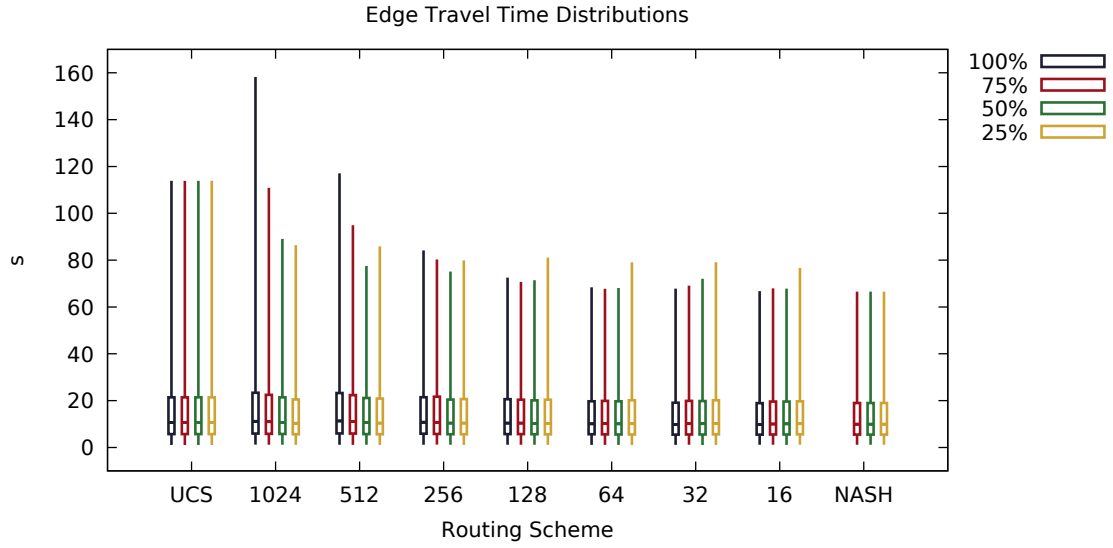

Figure 5.1: The edge travel time distributions for various routing schemes and penetration rates given complete information.

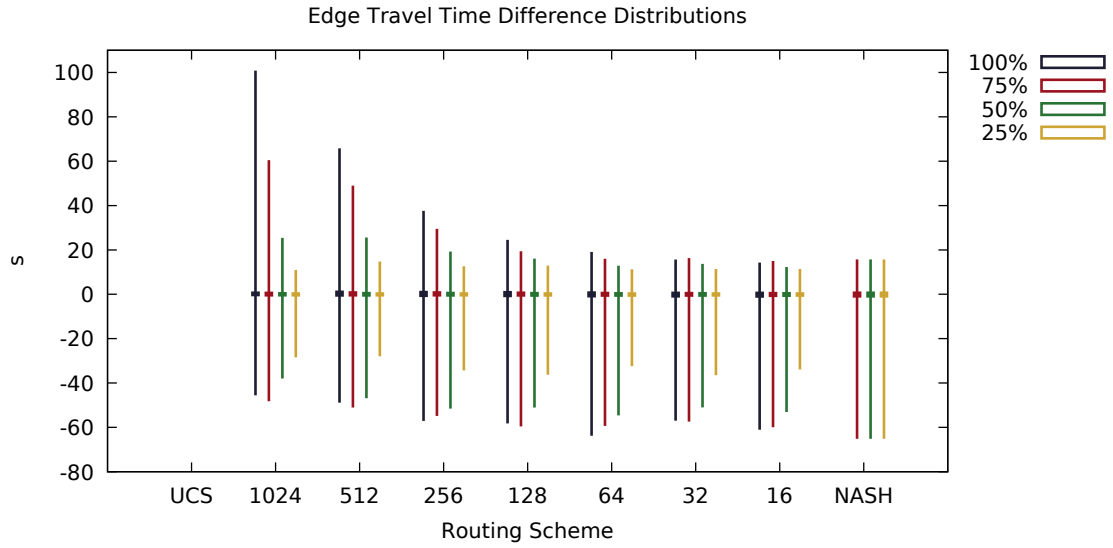

Figure 5.2: The edge travel time difference distributions for various routing schemes and penetration rates given complete information. Negative values indicate an advantage of dynamic routing.

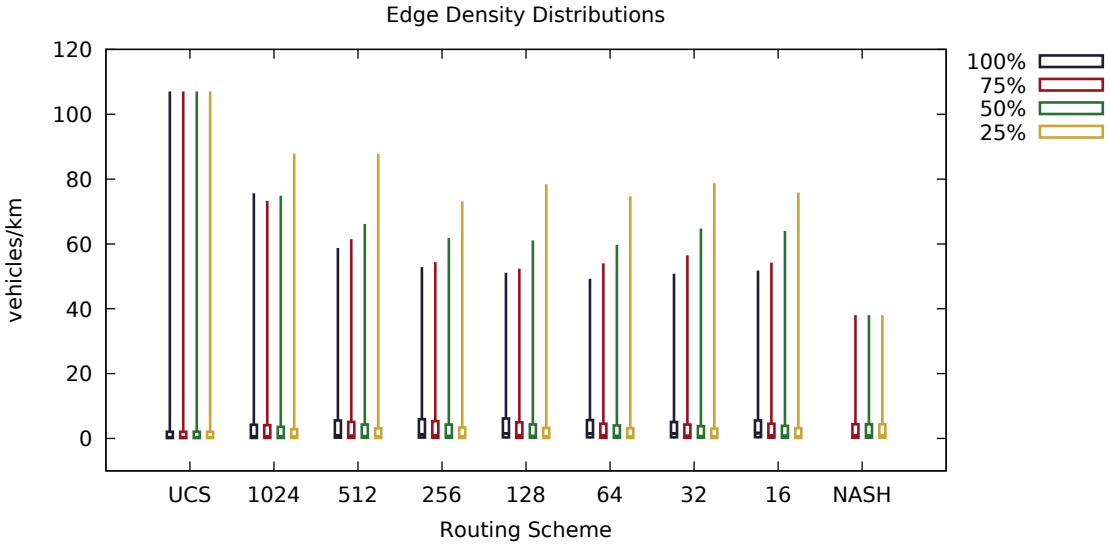

Figure 5.3: The edge density distributions for various routing schemes and penetration rates given complete information.

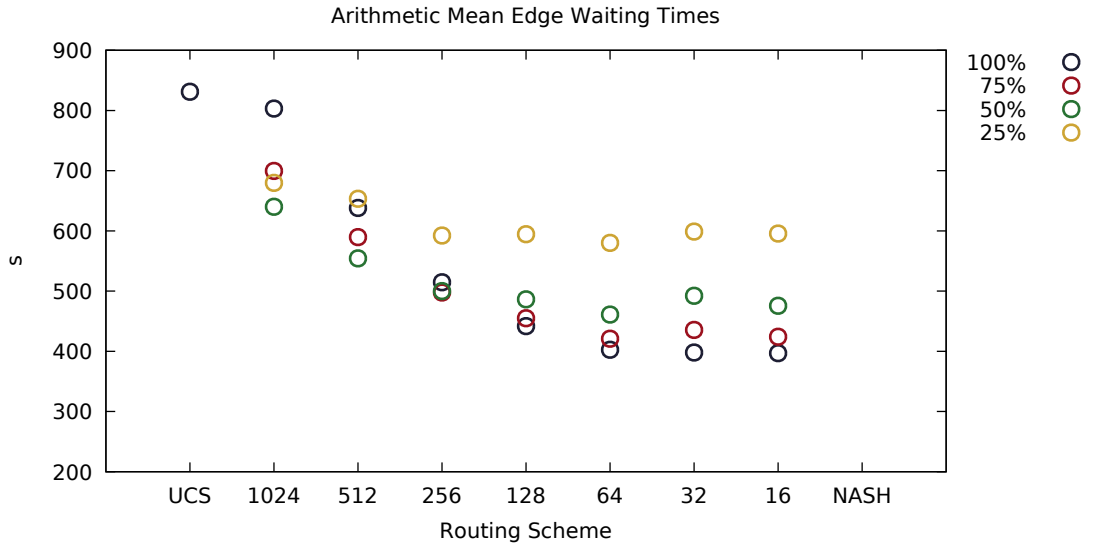

Figure 5.4: The arithmetic mean edge waiting time for various routing schemes and penetration rates given complete information.

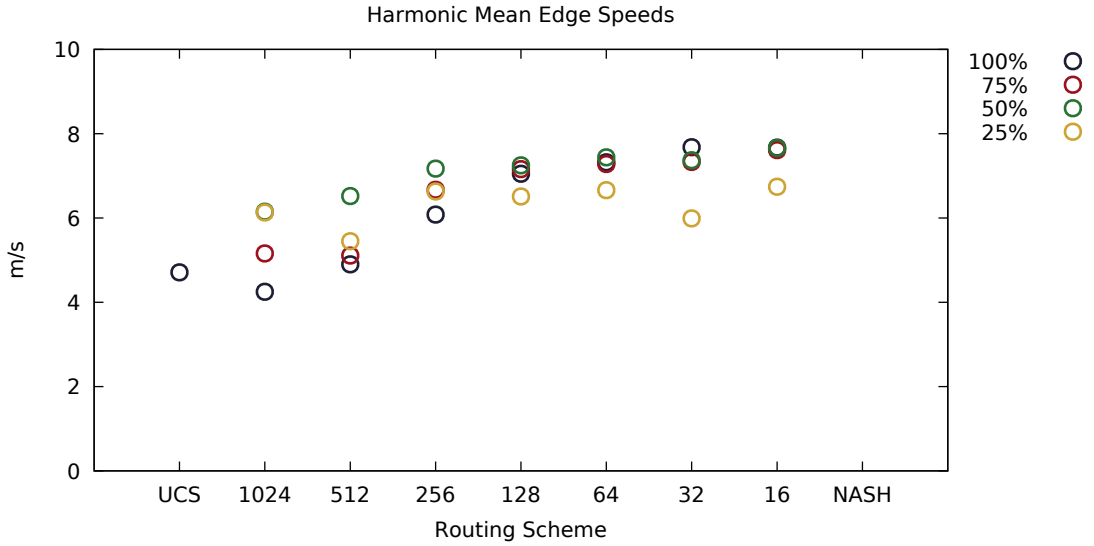

Figure 5.5: The harmonic mean edge speed for various routing schemes and penetration rates given complete information.

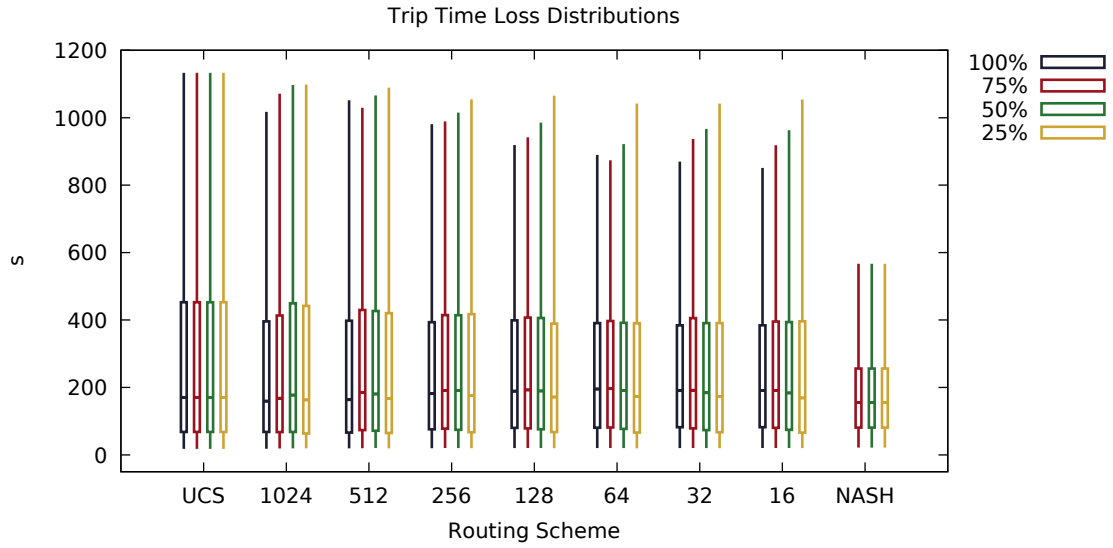

Figure 5.6: The trip time loss distributions for various routing schemes and penetration rates given complete information.

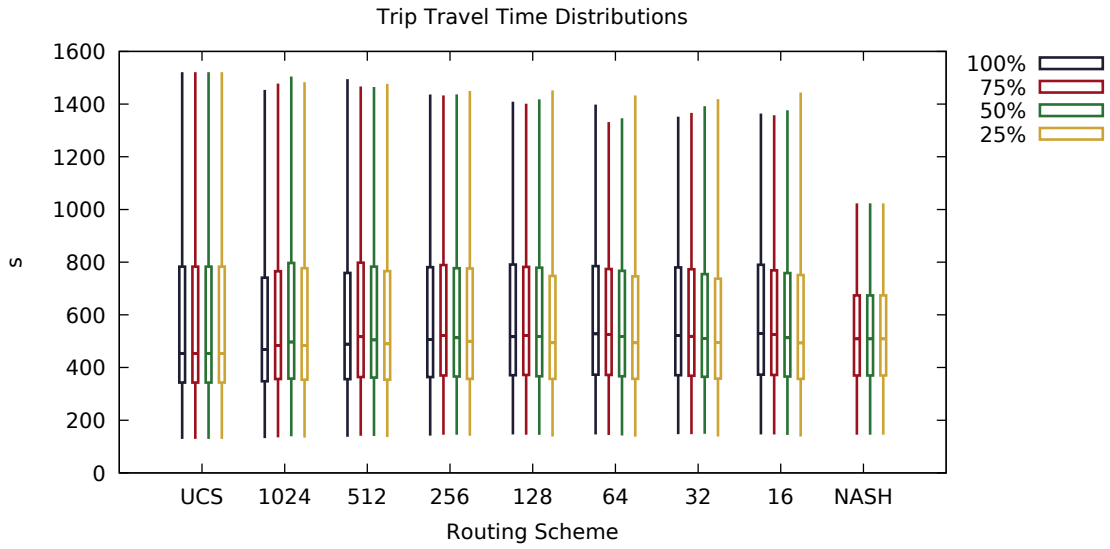

Figure 5.7: The trip travel time distributions for various routing schemes and penetration rates given complete information.

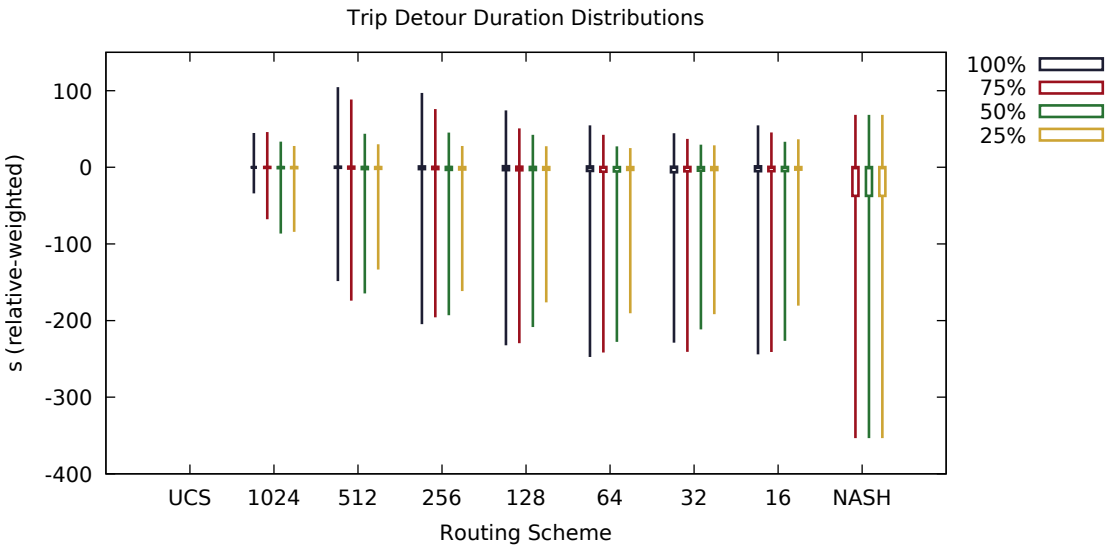

Figure 5.8: The relative-weighted trip detour duration distributions for various routing schemes and penetration rates given complete information. Negative values indicate an advantage of dynamic routing.

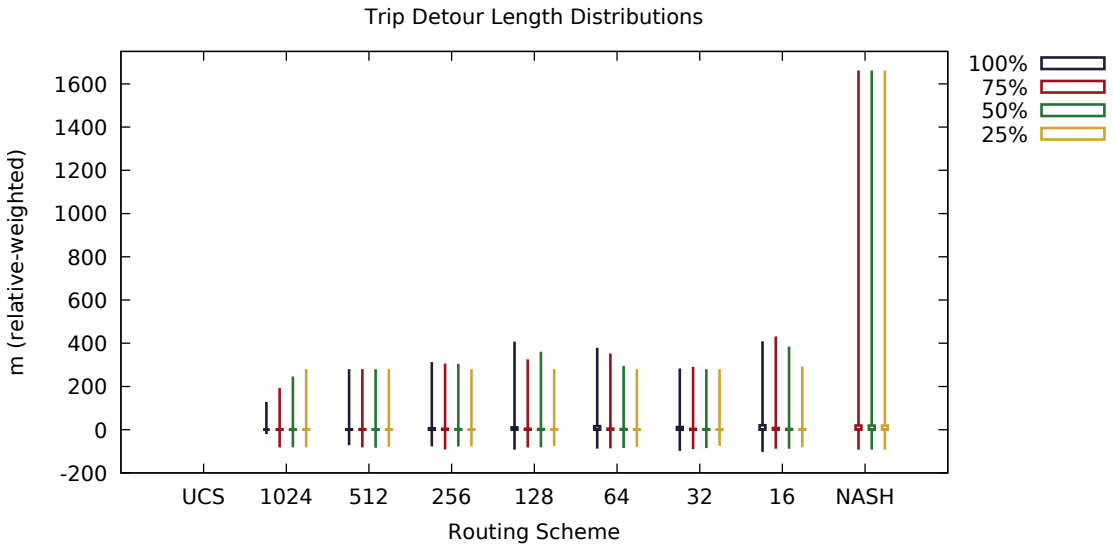

Figure 5.9: The relative-weighted trip detour length distributions for various routing schemes and penetration rates given complete information. Negative values indicate an advantage of dynamic routing.

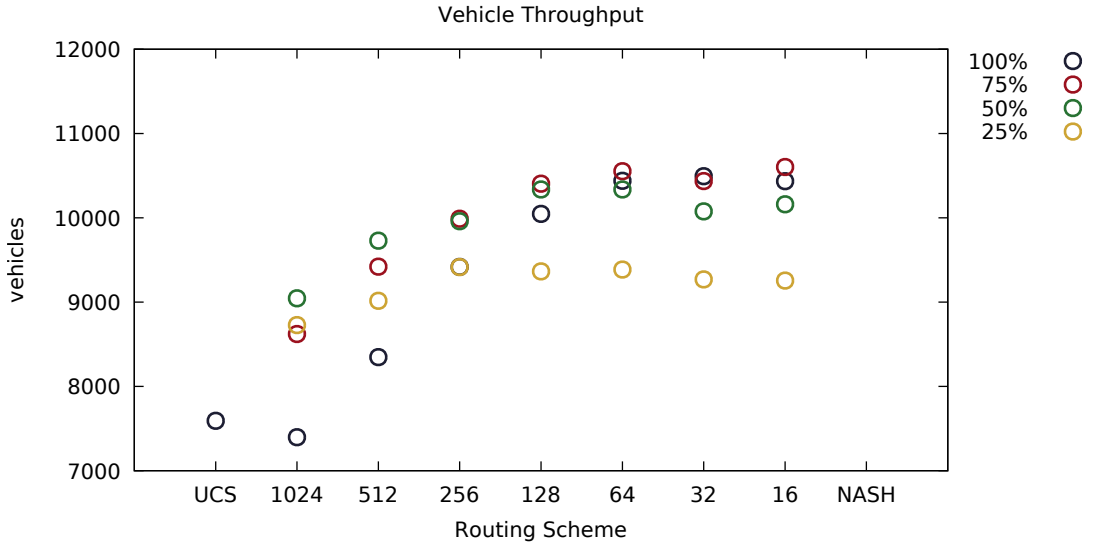

Figure 5.10: The network throughput for various routing schemes and penetration rates given complete information.

5.4 Experiment — Reduced Sensor Locations

The previous experiment placed a sensor on every road. In reality, sensors are much more scarcely distributed throughout the network. This experiment evaluates scenarios in which sensors are placed on 25% or 10% of all edges. The selected edges are identically distributed. Additionally, all figures show a sensor placement of 100% as an upper baseline. This experiment evaluates a penetration rate of 50%. The learning is performed on data collected solely using the reduced sensor set. The collection is performed using the unaltered demand data. Feedback is created using the primitive density congestion detection method. Each sensor continues to measure every vehicle.

Generally, all figures show that dynamic routing performs worse on a reduced set of sensors. This is mostly due to the limited information availability during the application phase, since congestion on roads without a sensor cannot be detected. Figure 5.11 shows that dynamic routing still positively affects edge travel times. This can also be seen when looking at the mean waiting time in figure 5.4. Those effects could be minimized by placing sensors specifically around congestion prone areas, which are usually already known to the responsible authorities. This would also increase the network throughput shown in figure 5.14. Finally, figure 5.13 shows that the advantages for some vehicles considerably outweigh the disadvantages for others.

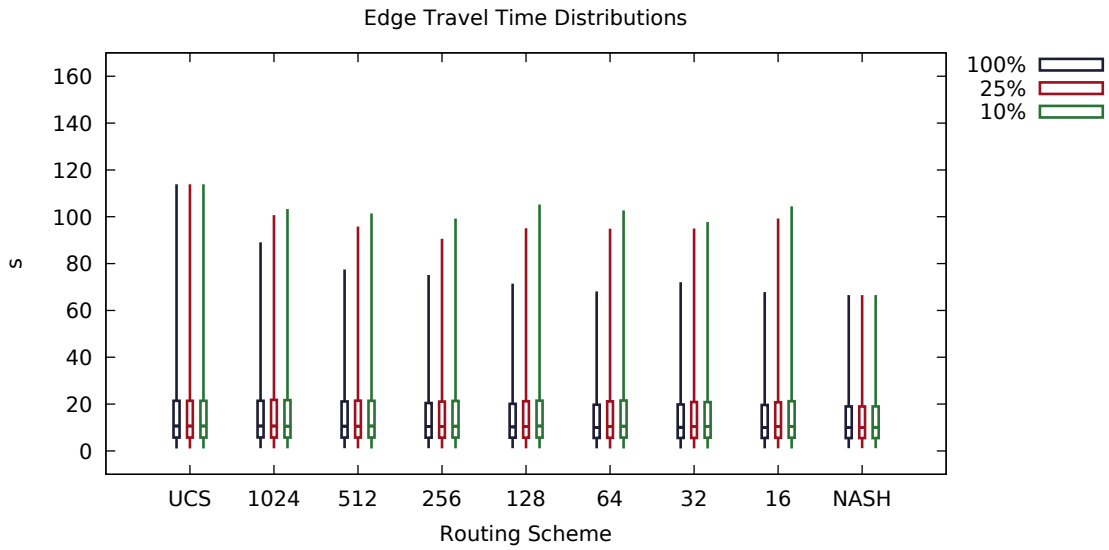

Figure 5.11: The edge travel time distributions for various routing schemes at 50% penetration rate when sensor locations were reduced.

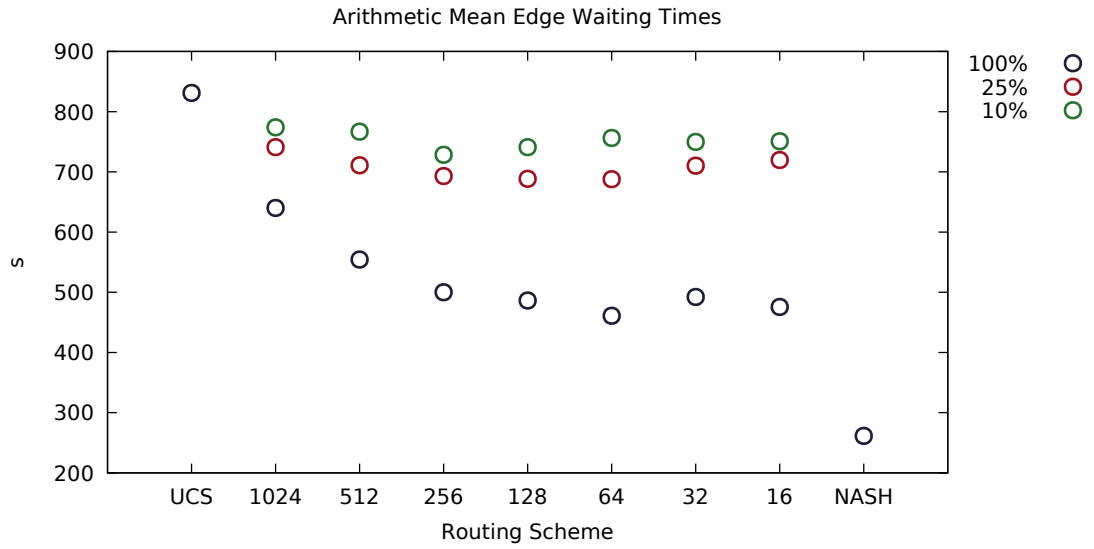

Figure 5.12: The arithmetic mean edge waiting time for various routing schemes at 50% penetration rate when sensor locations were reduced.

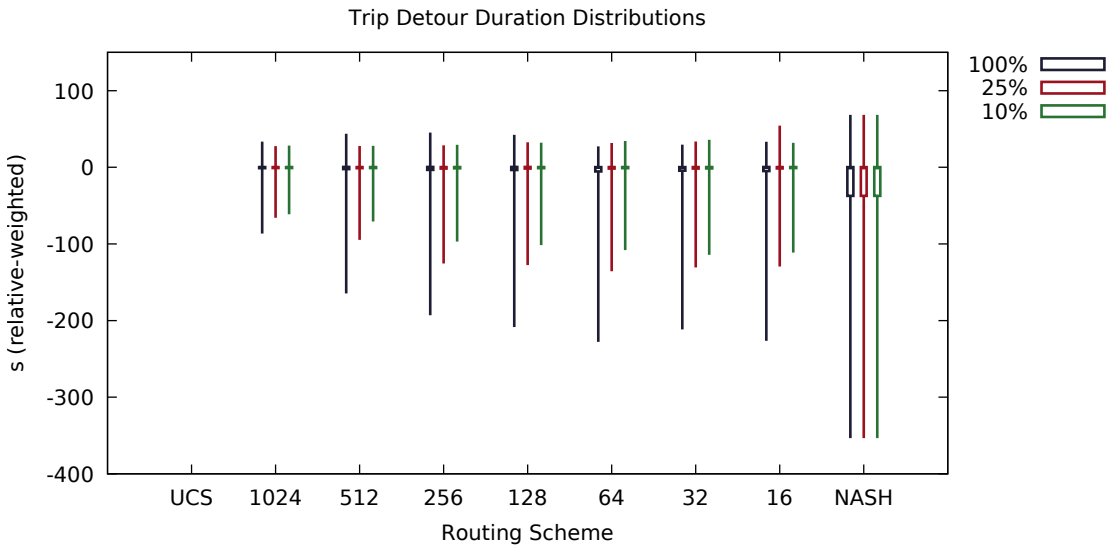

Figure 5.13: The relative-weighted trip detour duration distributions for various routing schemes at 50% penetration rate when sensor locations were reduced. Negative values indicate an advantage of dynamic routing.

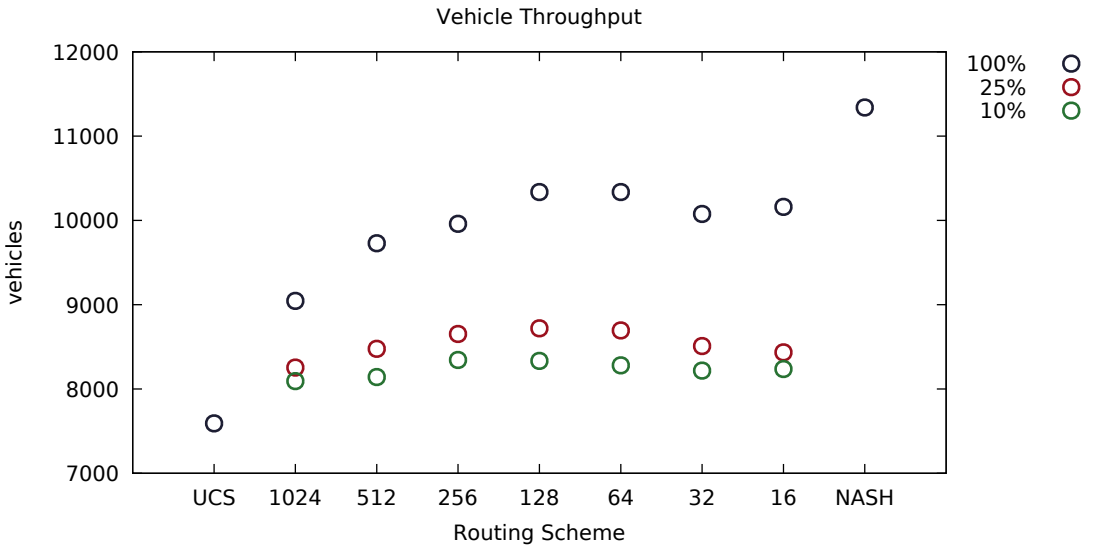

Figure 5.14: The network throughput for various routing schemes at 50% penetration rate when sensor locations were reduced.

5.5 Experiment — Reduced Sensor Information

The previous experiments used all available sensor measurements. This includes density, occupancy, vehicle count, mean speed and waiting time. This experiment evaluates dynamic routing where POEM learns using solely density and vehicle speed. These measurements were chosen since variations of both are regularly used in congestion detection methods [8, 15, 36]. A penetration rate of 50% is evaluated while sensors are again placed on every road and continue to measure every vehicle. The collection is performed using the unaltered demand data. Feedback is created using the primitive density congestion detection method. The figures additionally show results created with complete information for comparison.

All figures show that POEM is better at classifying congestion when given complete information. This is most noticeable when looking at the mean edge waiting time and vehicle throughput in figures 5.16 and 5.18. The individual routing performance is also negatively affected, which can be seen when looking at the relative-weighted detour durations in figure 5.17.

The reason why POEM performs worse might be that vehicle mean speeds alone do not carry enough information. A more sophisticated approach could combine mean vehicle speeds with respective speed limits, road conditions and signal positions. For instance, mean speeds in city centers will generally be lower than those on motorways due to frequent stops.

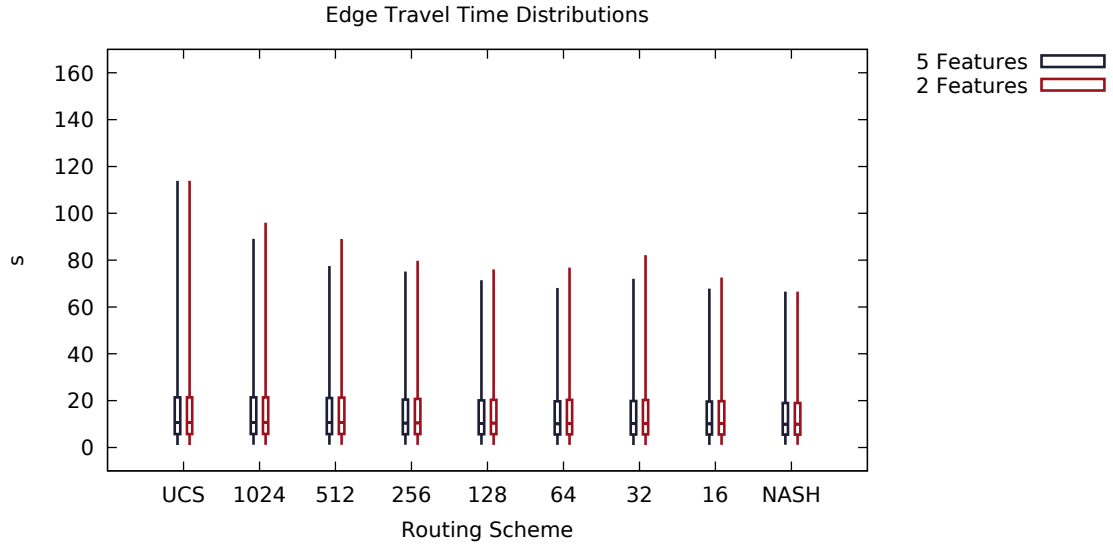

Figure 5.15: The edge travel time distributions for various routing schemes at 50% penetration rate when sensor information was reduced.

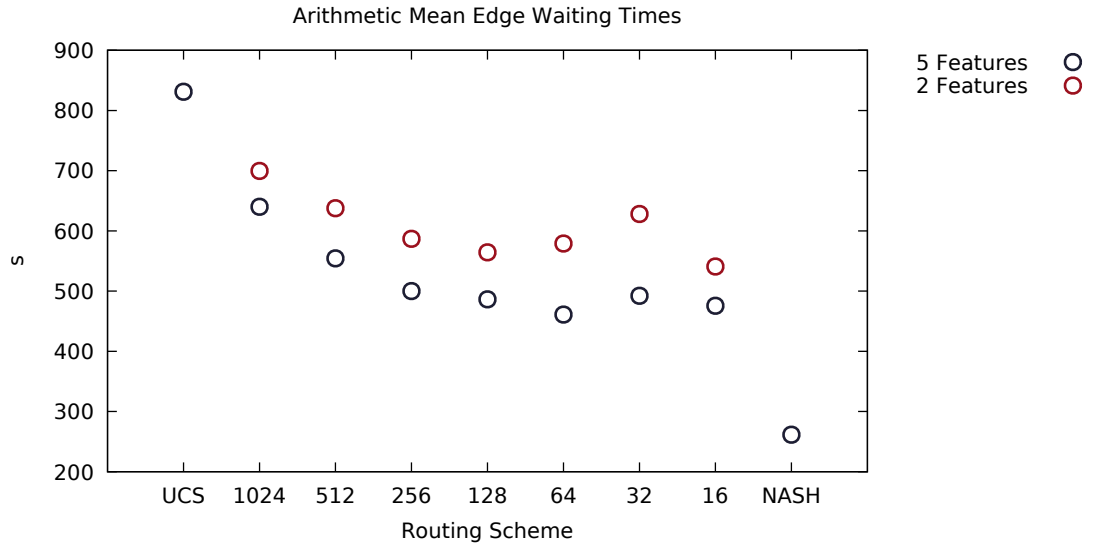

Figure 5.16: The arithmetic mean edge waiting time for various routing schemes at 50% penetration rate when sensor information was reduced.

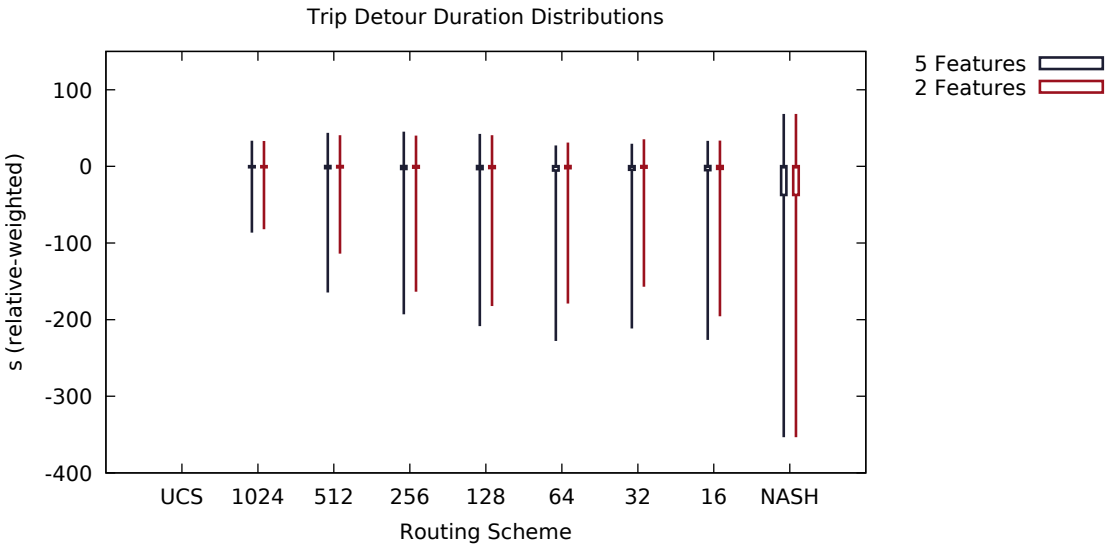

Figure 5.17: The relative-weighted trip detour duration distributions for various routing schemes at 50% penetration rate when sensor information was reduced. Negative values indicate an advantage of dynamic routing.

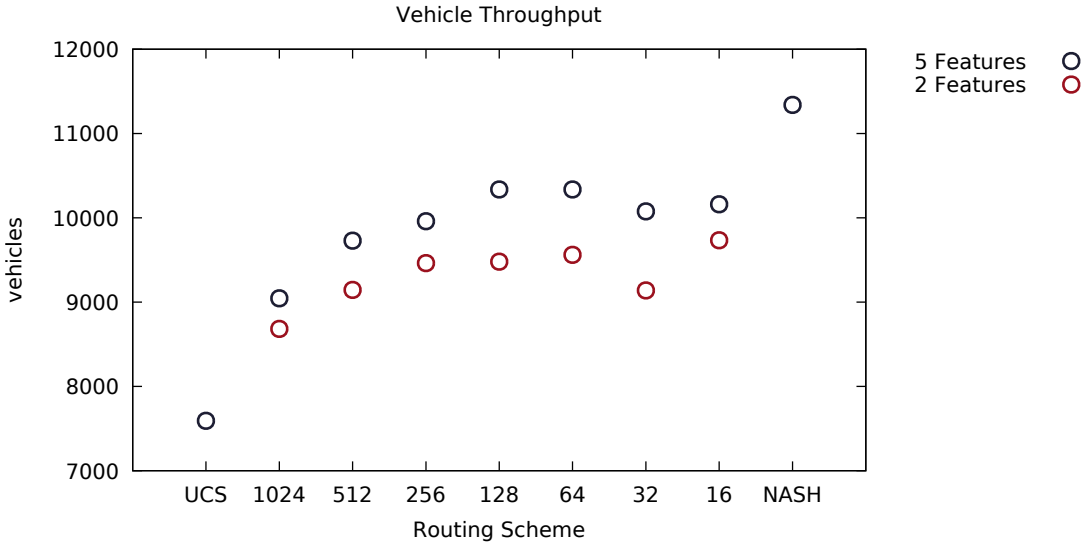

Figure 5.18: The network throughput for various routing schemes at 50% penetration rate when sensor information was reduced.

5.6 Experiment — Mobile Sensors

The sensors used in previous experiments measure every vehicle on their respective position. This information could be gathered using one of many permanently installed sensors, such as induction loops. Alternatively, smartphone navigation applications can be used as sensors. This experiment evaluates penetration rates of such applications of 40% and 20%, where participating vehicles are identically distributed. The measured data is inherently incomplete, as it is only collected for a subset of roads and vehicles. This especially presents a challenge to congestion detection, as information regarding vehicle density is not accurate anymore. For example, when only one vehicle using such an application is stuck in a jam on a motorway, very low density values are measured. This is why feedback is created using the primitive mean speed congestion detection method.

For vendors of such applications routing performance regarding all users is not particularly interesting. Their interest mostly lies in how great of an advantage users will have compared to non-users. Additionally, with such applications a jam can only be detected when sufficiently many users are already caught up in it. It would be important to evaluate how evenly the advantages distribute throughout the user base. However, such an evaluation would require long-term usage information, which is why it could not be performed.

Firstly, the relative-weighted detour duration presented in figures 5.19 and 5.20 show that users generally outperform non-users by a factor of two. This cannot be observed when looking at the Nash equilibrium, which indicates the results were indeed achieved by dynamic routing and not by a particularly favorable origin-destination distribution in the user set. The figures also show that dynamic routing performs best when using moderately sized intervals. This can also be observed when looking at the relative-weighted detour waiting times in figures 5.21 and 5.22.

The performance at low penetration rates is particularly interesting in real-world scenarios, since currently no single system dominates the market. It could be further improved by additionally utilizing stationary sensor data.

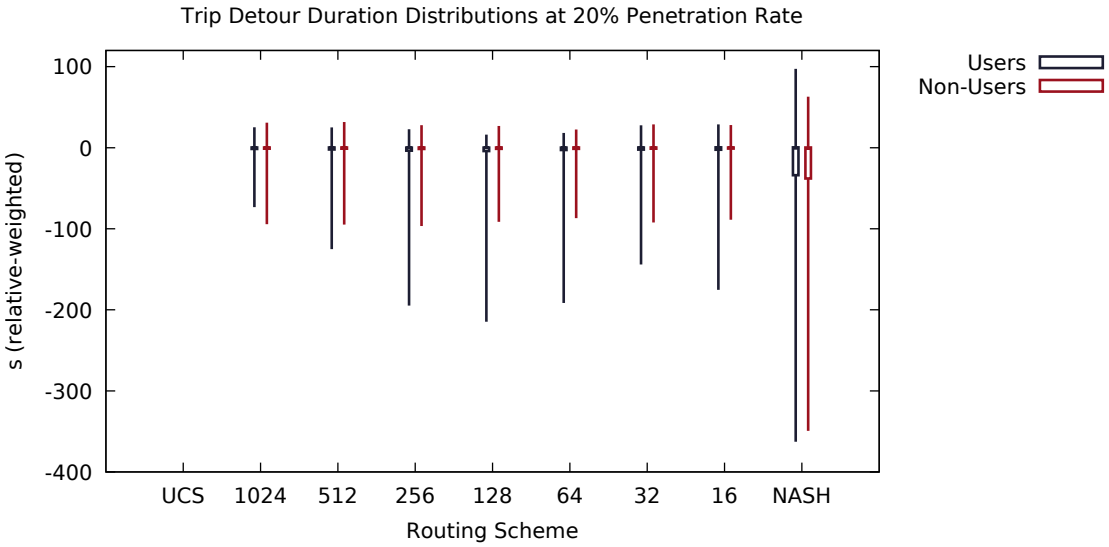

Figure 5.19: The relative-weighted trip detour distributions for various routing schemes given an in-vehicle sensor penetration rate of 20%. The sensor information was only available to vehicles equipped with a sensor. Negative values indicate an advantage of dynamic routing.

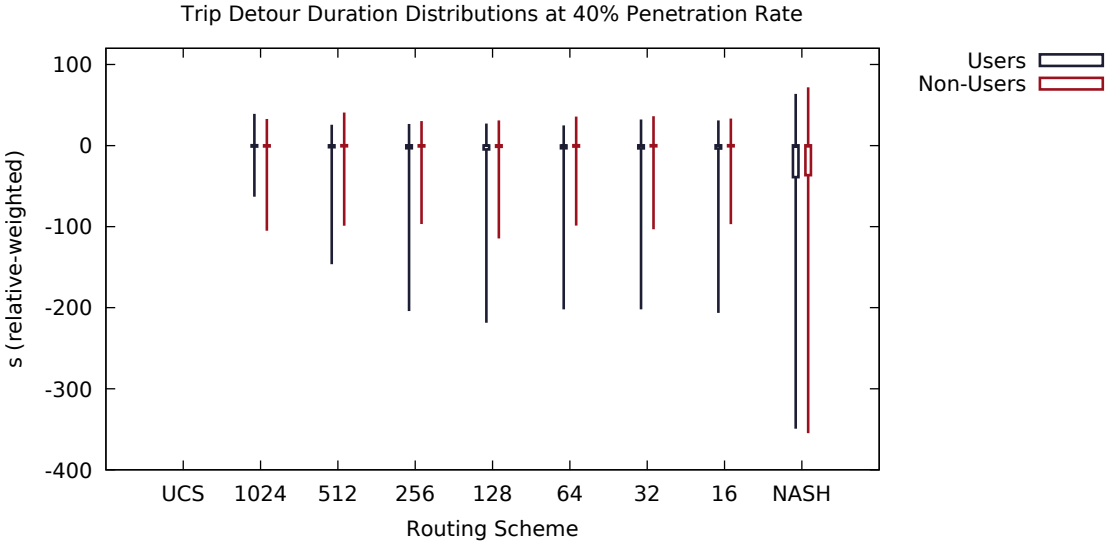

Figure 5.20: The relative-weighted trip detour distributions for various routing schemes given an in-vehicle sensor penetration rate of 40%. The sensor information was only available to vehicles equipped with a sensor. Negative values indicate an advantage of dynamic routing.

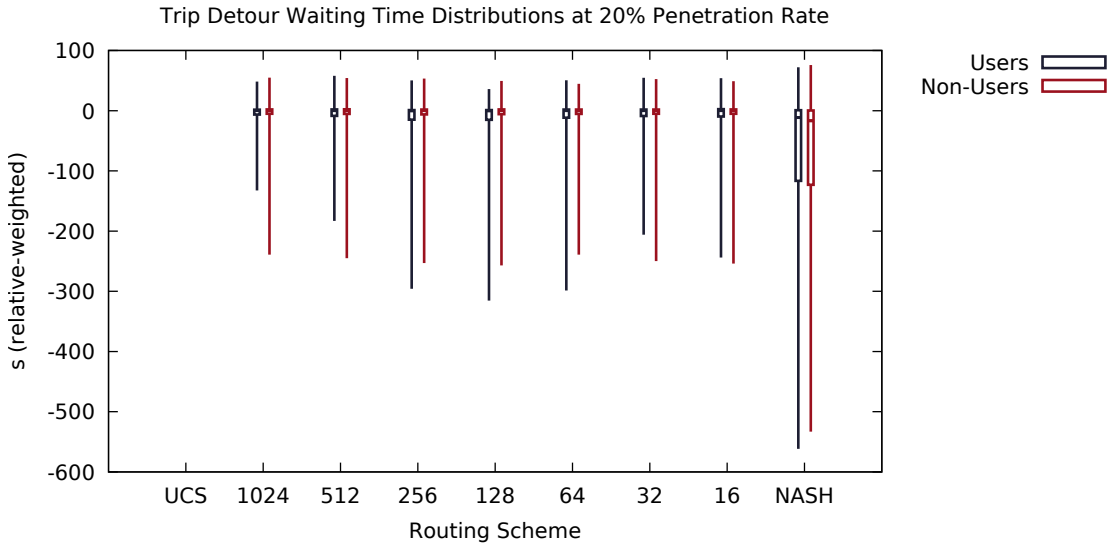

Figure 5.21: The relative-weighted trip waiting time distributions for various routing schemes given an in-vehicle sensor penetration rate of 20%. The sensor information was only available to vehicles equipped with a sensor. Negative values indicate an advantage of dynamic routing.

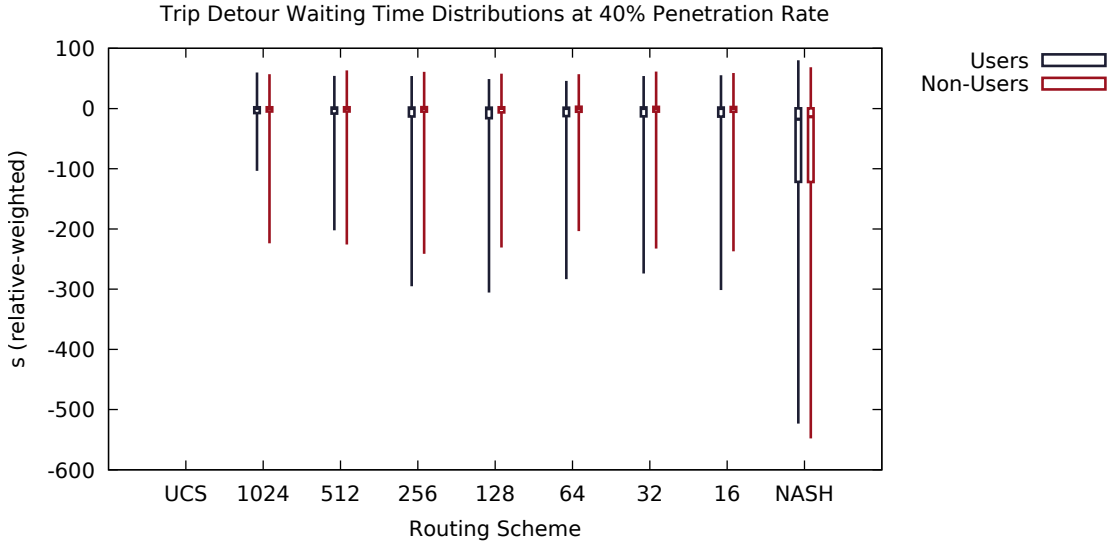

Figure 5.22: The relative-weighted trip waiting time distributions for various routing schemes given an in-vehicle sensor penetration rate of 40%. The sensor information was only available to vehicles equipped with a sensor. Negative values indicate an advantage of dynamic routing.

5.7 Experiment — Reapplication

The experiment with complete information in section 5.3 showed that dynamic routing performs poorly on large interval sizes when all vehicles are rerouted. This experiment evaluates whether or not a policy can be further improved by reapplying POEM. Firstly, POEM is applied on the default demand and policy. Then, POEM is applied on the data created by using the previously learned policy. A penetration rate of 100% is evaluated. Feedback is again created using the primitive density congestion detection method.

The results in figure 5.23 show that the creation of more jams in larger interval sizes can be avoided by reapplying POEM. However, it does not considerably outperform UCS, which might be caused by the large interval size in relation to the evaluation period. For an interval size of 1024 seconds vehicles are only rerouted once, which means many vehicles are not rerouted at all. This can also be seen when looking at the individual detour durations shown in figure 5.24. The figures also show that performance gets worse for smaller interval sizes.

Those effects can be explained by the exploration vs exploitation dilemma. The policy created by the initial application of POEM creates more congestion in large interval sizes. The logged data thus explores congested roads very well, which in turn allows POEM to further optimize the policy. In contrast, congestion is minimized in small interval sizes. The logged data does not explore congested roads, but rather exploits uncongested ones, meaning it does not provide POEM with good information about characteristics of congestion.

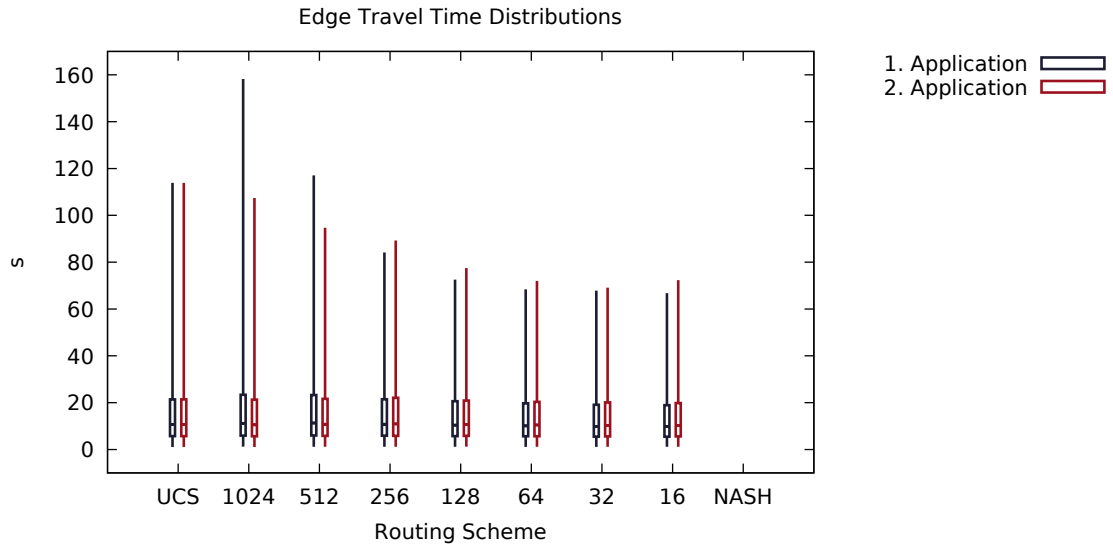

Figure 5.23: The edge travel time distributions for various routing schemes at 100% penetration rate when reapplying POEM and given complete information.

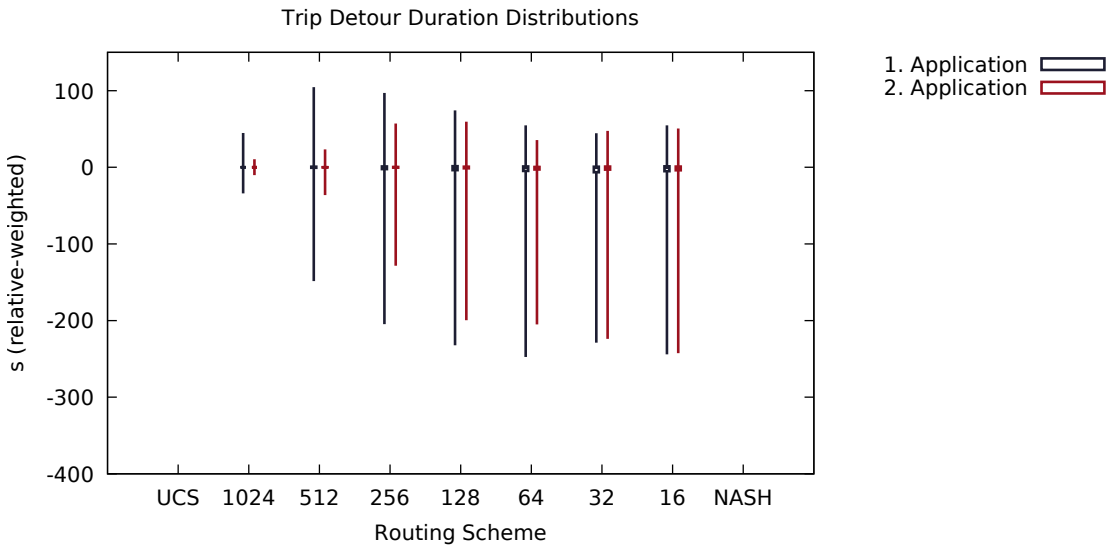

Figure 5.24: The relative-weighted trip detour duration distributions for various routing schemes at 100% penetration rate when reapplying POEM and given complete information. Negative values indicate an advantage of dynamic routing.

Chapter 6

Conclusion

The experiments showed that reinforcement learning, specifically *Policy Optimizer for Exponential Models* (POEM) [38] can be used to successfully detect congestion in road networks. The predictions by POEM were used by dynamic routing in order to distribute vehicles more evenly, which effectively reduced overall network congestion. The experiments also showed that dynamic routing with POEM can be deployed using a smartphone navigation system. This greatly simplifies deployment as it does not require any new sensors to be deployed. However, existing sensors can still be included when available to increase prediction accuracy. The deployment in smartphone applications is also of commercial interest, since it provides its users with an advantage.

Future research will evaluate whether or not dynamic routing with POEM can be applied on a larger spatio-temporal scale. However, Luxembourg is currently the largest open-source road network of adequate quality for realistic simulations in *Simulation for Urban Mobility* (SUMO) [24]. This is why either a larger road network must be revised or a real-world application using open-source maps such as *Open Street Map* must be developed. The real-world application allows long-term user statistics to be created. Those can be used to study which users are provided with the greatest advantage. For simulations Cologne is currently under revision [39].

Another area of research is the application of POEM for congestion detection, since only one of multiple possible approaches was outlined. For instance, min-max scaling of feature vectors may be substituted with standard-score or unit-length scaling. Furthermore, section 5.5 already discussed an improvement of the utilized features, which will most likely have a great impact on the performance of congestion detection. Similarly, more sophisticated methods of feedback creation might also lead to better results. Including road types and conditions will lead to increased accuracy due to the varying characteristics of city and motorway traffic. Finally, POEM must also be compared to other congestion detection algorithms in order to show its advantage with respect to performance.

Although dynamic routing performed well in simulated scenarios, case studies must evaluate whether or not it is applicable in real-world situations. Special consideration must be given to unexpected, intricate route changes, which might be a common occurrence in dynamic routing. The performance in unusual situations such as accidents or road closures must also be evaluated. However, even when dynamic routing proofs to be unsuitable, POEM can still be used in combinations with other routing schemes. Future research will specifically aim at comparing POEM with *r-Extreme Signalling* [28].

This thesis also showed that reinforcement learning is generally applicable in a road network environment. While POEM was only applied to congestion detection, it might also proof to be advantageous in other problems. One prospective area of research is the control of traffic signals using reinforcement learning, which already showed promising results in [4] and [34]. Another area of research will be the implementation of reinforcement learning on a per-vehicle scale, meaning each vehicle represents a learning agent which solely learns using data created by itself. This greatly simplifies deployment, since it does not require any coordination between drivers. However, a single vehicle only generates a small amount of data, which poses a challenge to reinforcement learning algorithms.

List of Figures

3.1 The relations between flow, density and speed 7
3.2 The components in a reinforcement learning setting 10

5.1 The edge travel time given complemente information 25
5.2 The edge travel time difference given complete information 25
5.3 The edge density given complete information 26
5.4 The mean edge waiting time given complete information 26
5.5 The mean edge speed given complete information 27
5.6 The trip time loss given complete information 27
5.7 The trip travel time given complete information 28
5.8 The trip detour duration given complete information 28
5.9 The trip detour length given complete information 29
5.10 The network throughput given complete information 29
5.11 The edge travel time given reduced sensor locations 30
5.12 The mean edge waiting time given reduced sensor locations 31
5.13 The trip detour duration given reduced sensor locations 31
5.14 The network throughput given reduced sensor locations 32
5.15 The edge travel time given reduced sensor information 33
5.16 The mean edge waiting time given reduced sensor information 33
5.17 The trip detour duration given reduced sensor information 34
5.18 The network throughput given reduced sensor information 34
5.19 The trip detour duration given mobile sensors at 20% penetration rate . . . 36
5.20 The trip detour duration given mobile sensors at 40% penetration rate . . . 36
5.21 The trip detour waiting time given mobile sensors at 20% penetration rate . 37
5.22 The trip detour waiting time given mobile sensors at 40% penetration rate . 37
5.23 The edge travel time when reapplying POEM 39
5.24 The trip detour duration when reapplying POEM 39

Bibliography

[1] Real time traffic information. Technical report, TomTom, 2015. `https://goo.gl/zyh6tk` Accessed: 10.10.2016.

[2] Traffic index. Technical report, TomTom, 2016. `http://goo.gl/1v5b5z` Accessed: 10.10.2016.

[3] P. T. C. . U. Areas and I. U. Transport. Congestion in urban areas - examples of counter-measures. Technical report, 2010.

[4] I. Arel, C. Liu, T. Urbanik, and A. Kohls. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4(2):128–135, 2010.

[5] J. Aslam, S. Lim, X. Pan, and D. Rus. City-scale traffic estimation from a roving sensor network. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 141–154. ACM, 2012.

[6] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[7] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 322–331. IEEE, 1995.

[8] D. Barth. The bright side of sitting in traffic: Crowdsourcing road congestion data. `http://goo.gl/NeerFB`, 2009. Accessed: 01.10.2016.

[9] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck. Route planning in transportation networks. *arXiv preprint arXiv:1504.05140*, 2015.

[10] R. Bhoraskar, N. Vankadhara, B. Raman, and P. Kulkarni. Wolverine: Traffic and road condition estimation using smartphone sensors. In *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*, pages 1–6. IEEE, 2012.

[11] G. D. Bottoms. Continuing developments in light rail transit in western europe. In *Proc., 9th National Light Rail Transit Conference*, pages 713–728. Citeseer, 2003.

[12] K. Chatterjee, N. Hounsell, P. Firmin, and P. Bonsall. Driver response to variable message sign information in london. *Transportation Research Part C: Emerging Technologies*, 10(2):149–169, 2002.

[13] H. Chaudhary. Application of the theory of a single first order equation to traffic flow. *Journal of the Institute of Engineering*, 9(1):175–181, 2014.

[14] L. Codeca, R. Frank, and T. Engel. Luxembourg SUMO Traffic (LuST) Scenario: 24 hours of mobility for vehicular networking research. In *Vehicular Networking Conference (VNC), 2015 IEEE*, pages 1–8. IEEE, 2015.

[15] A. M. de Souza, R. S. Yokoyama, G. Maia, A. Loureiro, and L. Villas. Real-time path planning to prevent traffic jam through an intelligent transportation system. In *Computers and Communication (ISCC), 2016 IEEE Symposium on*, pages 726–731. IEEE, 2016.

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

[17] L. Figueiredo, I. Jesus, J. T. Machado, J. Ferreira, and J. M. de Carvalho. Towards the development of intelligent transportation systems. In *Intelligent Transportation Systems*, volume 88, pages 1206–1211, 2001.

[18] T. Hägerstrand. Tiidsgeografisk Beskrivning. Syfte och Postulat. *Svensk Geografisk Årsbok*, 50:86–94, 1974.

[19] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[20] S. Hoogendoorn and V. Knoop. *Traffic Flow Theory and Modelling*. Edward Elgar Publishing Limited, 2012.

[21] L. Huapu. Urban transportation in china: Current state of reform and future trends. *JOURNEYS*, page 7, 2009.

[22] R. Katzev. Car sharing: A new approach to urban transportation problems. *Analyses of Social Issues and Public Policy*, 3(1):65–86, 2003.

[23] D. Kopitz and B. Marks. *RDS: The Radio Data System*. Artech House, 1999.

[24] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker. Recent development and applications of SUMO - Simulation of Urban Mobility. *International Journal on Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.

[25] T. Liebig. Privacy preserving centralized counting of moving objects. In F. Bacao, M. Y. Santos, and M. Painho, editors, *AGILE 2015*, Lecture Notes in Geoinformation and Cartography, pages 91–103. Springer International Publishing, 2015.

[26] T. Liebig, N. Piatkowski, C. Bockermann, and K. Morik. Dynamic route planning with real-time traffic predictions. *Information Systems*, 1(1):(In Press), 2016.

[27] R. Liu, H. Liu, D. Kwak, Y. Xiang, C. Borcea, B. Nath, and L. Iftode. Themis: A participatory navigation system for balanced traffic routing. In *2014 IEEE Vehicular Networking Conference (VNC)*, pages 159–166. IEEE, 2014.

[28] J. Mareček, R. Shorten, and J. Y. Yu. r-extreme signalling for congestion control. *International Journal of Control*, pages 1–13, 2016.

[29] B. McCann. A review of scats operation and deployment in dublin. In *Proceedings of the 19th JCT Traffic Signal Symposium & Exhibition. JCT Consulting Ltd*, 2014.

[30] R. McGill, J. W. Tukey, and W. A. Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, 1978.

[31] J. Nash. Non-cooperative games. *Annals of Mathematics*, pages 286–295, 1951.

[32] S. Oh, S. Ritchie, and C. Oh. Real-time traffic measurement from single loop inductive signatures. *Transportation Research Record: Journal of the Transportation Research Board*, (1804):98–106, 2002.

[33] J. Pan, M. A. Khan, I. S. Popa, K. Zeitouni, and C. Borcea. Proactive vehicle re-routing strategies for congestion avoidance. In *2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems*, pages 265–272. IEEE, 2012.

[34] L. Prashanth and S. Bhatnagar. Reinforcement learning with function approximation for traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):412–421, 2011.

[35] T. Roughgarden and É. Tardos. How bad is selfish routing? *Journal of the ACM (JACM)*, 49(2):236–259, 2002.

[36] S. Roy, S. Bandyopadhyay, M. Das, S. Batabyal, and S. Pal. Real time traffic congestion detection and management using active rfid and gsm technology. *ITSC, Kyoto, Japan*, 2010.

[37] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*, volume 1. MIT press Cambridge, 1998.

[38] A. Swaminathan and T. Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research*, 16:1731–1755, 2015.

[39] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas. Generation and analysis of a large-scale urban vehicular mobility dataset. *IEEE Transactions on Mobile Computing*, 13(5):1061–1075, 2014.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den 17. Oktober 2016

Maurice Sotzny