On Avoiding Traffic Jams with Dynamic Self-Organizing Trip Planning*

Thomas Liebig[1] and Maurice Sotzny[2]

1 **TU Dortmund University, Dortmund, Germany**
thomas.liebig@tu-dortmund.de
2 **TU Dortmund University, Dortmund, Germany**
maurice.sotzny@tu-dortmund.de

Abstract

Urban areas are increasingly subject to congestions. Most navigation systems and algorithms that avoid these congestions consider drivers independently and can, thus, cause novel congestions at unexpected places. Pre-computation of optimal trips (Nash equilibrium) could be a solution to the problem but is due to its static nature of no practical relevance.

In contrast, the paper at-hand provides an approach to avoid traffic jams with dynamic self-organizing trip planning. We apply reinforcement learning to learn dynamic weights for routing from the decisions and feedback logs of the vehicles. In order to compare our routing regime against others, we validate our approach in an open simulation environment (LuST) that allows reproduction of the traffic in Luxembourg for a particular day. Additionally, in two realistic scenarios: (1) usage of stationary sensors and (2) deployment in a mobile navigation system, we perform experiments with varying penetration rates. All our experiments reveal that performance of the traffic network is increased and occurrence of traffic jams are reduced by application of our routing regime.

1998 ACM Subject Classification I.2.11 Distributed Artificial Intelligence

Keywords and phrases situation-aware trip planning, self-organizing traffic, reinforcement learning

Digital Object Identifier 10.4230/LIPIcs.COSIT.2017.17

1 Introduction

During the transition towards smart cities, intelligent traffic systems are used to detect current traffic hazards [10], to predict future traffic states [20] or to provide situation dependent navigation suggestions to drivers [13]. Due to the complex nature of everyday traffic, precise travel time prediction has proven to be an algorithmically challenging problem. Its accuracy is inherently dependent on various inputs, such as spatio-temporal variables, road supply, road demand, vehicle usage, and overall network quality [4].

The routes, however, should avoid current and upcoming traffic jams. This can easily be done individually, by a navigation device or a routing app (e.g. Google) but this could become problematic, as it does not consider greedy route choice amongst the drivers. Every driver uses comparable edge weights and optimal roads are overrepresented. This may lead to novel unexpected congestion on optimal roads during peak periods. And, in turn, optimal roads are no longer optimal.

* This research has been financed by the European Union through the Horizon 2020 688380 VaVeL project.

13th International Conference on Spatial Information Theory (COSIT 2017).
Editors: Eliseo Clementini, Maureen Donnelly, May Yuan, Christian Kray, Paolo Fogliaroni, and Andrea Ballatore;
Article No. 17; pp. 17:1–17:12

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Figure 1 Motivating example to dynamic self-organizing routing. Whereas a car may take the apparently optimal route from S to D (middle) it may also avoid causing unnecessary congestion and use the route depicted to the right. Best viewed in color.

This problem could be approached in two ways. If one knows all trips in advance, one could find optimal static weights amongst the drivers to gain optimal flow through the city. This approach results in a Nash equilibrium, as discussed in [19]. The second approach is to apply dynamic routing and perform self-organization, this approach is yet unexplored and subject to the paper-at-hand.

In contrast to static routing methods, which do not care about other persons decision, in dynamic self-organizing routing (also from a given start to a target location) the drivers continuously answer following questions:

- Which turn should I make?
- Which effects will my decision have?

A visual representation of the expected behaviour is presented in Figure 1. Given a vehicle travelling from location S to D it might choose amongst two possible paths. Though the upper route choice (depicted in the middle) uses the main road and may pertain a better static (also predicted) cost, the vehicle avoids this road as it realizes that it causes unnecessary congestion on this road. The alternative (right) has a reasonable dynamic cost and avoids unnecessary congestion on the main road.

Our approach to this problem is to apply reinforcement learning, as we just observe which decisions the vehicles have done plus their result, but none of the cases that would have happened if some vehicles would have turned differently. We therefore apply a method for learning from bandit feedback to the routing problem. Our experiments reveal that this approach successfully increases performance of the traffic network and reduces traffic jams.

The paper is structured as follows. Section 2 discusses other, non–self-organizing, approaches to congestion preventing traffic control. In Section 3 we present the mapping of the problem to bandit feedback learning and introduce a recent approach to this problem. Section 4 presents the experiments we performed. Here we test the overall achieved performance and the performance we achieve for different penetration rates and deployment scenarios, i.e. stationary or moving sensors. Finally, in Section 5 we discuss future research directions.

2 Related Work

In this section we fist present literature on trip planning problems. Afterwards, we will briefly discuss related approaches for traffic control.

Before digging into our approach on dynamic trip planning, we present some fundamentals on (static) trip planning. The task to plan a route from one start location to a target location is called trip planning, when multiple means of transportation (also called ‘travel modes’) are involved this becomes multi-modal trip planning. The integration of transportation systems with personal constraints, residential and city services systems can offer real promise for implementing an intelligent transportation infrastructure that can efficiently address issues

beyond congestion, resiliency and safety. Trip planning operates on a graph representation of the road and transit network the so-called traffic network G consisting of vertices V (e.g. junctions) and connecting edges E (e.g. streets). A cost function maps each edge to a positive number that denotes how much it would 'cost' to travel the corresponding segment. The cost function needs to be consistent throughout the traffic network, but can be defined in several ways, such that it holds the most relevant aspects: for example length of the segment, travel time, or comfortableness. With a given start and end location in the traffic network, trip planning searches the path that connects start and goal and minimizes the cost.

Several algorithms exist to compute this minimizing path. Dijkstra [5] proposes a best-first traversal of the graph where the candidates for traversal are hold in a priority-queue. In the slightly modified version of the algorithm A* [8] the order in the priority-queue for the traversal not only depends on the cumulated costs to reach a vertex in the graph but also on the expected costs to reach the goal from this vertex. Bound by Minkowski's inequality, whereas $||x + y||_p \leq ||x||_p + ||y||_p$ (known as triangle inequality for $p = 2$), A* prunes the search space in comparison to Dijkstra's Algorithm. A sound heuristic for the remaining cost estimation is the geographical distance that is always lower than the road-based distance. In case of static cost functions contraction hierarchies [6] are a data structure that speeds-up the A* algorithm and enables trip calculation in large traffic networks at European scale. Instead of searching the shortest path directly within the traffic network, contraction hierarchies reduce the search space to the most important ones. In a preprocessing step these important segments are identified (based on the topology) and the network is extended by edges between these important links.

According to Hoogendorn [9], individual movement is performed in three layers:

- *Strategic Level*: In the strategic level the driver chooses its target and the strategy how to get there. This is the self estimated best route, among a collection of different alternatives. This can be done based on experience. Examples could be the global shortest path or the familiar path to a given destination.
- *Tactical Level*: Short-terms decisions are made at the tactical level, avoiding jams or switching to a faster route for instance. Thus, the person chooses the path to avoid obstacles. Basic rules for motions are defined at the tactical level, which include accelerating, decelerating and stopping.
- *Operational Level*: In the operational level, the motion to the next intermediate point is performed, for example, decision for a movement direction and speed or planning of the next step.

Based on this characteristic, it is clear that on each layer of this hierarchy smart methods could improve performance of traffic. The game-theoretic Nash equilibrium [18], applied in [19] operates on the *strategic level* and the route is chosen such that any driver may not get a better travel time by changing its own travel plan. Recently proposed system for self-organising traffic [22], uses slots instead of traffic lights and operates on the *operational level* of motion.

In contrast, our approach works on the *tactical level*, as it predicts and avoids jams. As opposed to [13], our approach allows altering the route at driving time and prevents creation of jams by the given navigation advices. Latter aspect is also focus of the research performed in [16, 15]. Their approach is to obfuscate the given signals such that the traffic distributes better, whereas we use regular sensor data as available from traffic loop networks or navigation devices and provide space time dependent suggestions, which are (based on the different locations of the traffic participants) individual.

3 Combination of Routing Decisions and Congestion Feedback for Reinforcement Learning

Traffic closely resembles a bandit feedback learning environment (compare [1] for an introduction to bandit learning). Bandit learning is a reinforcement learning task, where the behaviour of some blackbox (e.g. a bandit) should be learned just by the feedback we observe, several actions can be taken (in the bandit problem this equals drawing an arm). However, only the result of the actions can be observed and it is unknown what would have happened otherwise. Vehicles serve as agents which move in a road network. The actions are represented by the roads a vehicle can choose at an intersection. Once a road was chosen, a reward will be assigned for that particular road depending on its actual state. The reward for all other roads which could have been chosen remains unknown. This lack of fully labeled data makes a supervised learning approach particularly complex.

The *Policy Optimizer for Exponential Models* (POEM) [21] is able to learn solely based on the reward values provided by the environment. Additionally, POEM does not perform on-line learning, but rather uses logged data. This abstraction is known from bandit problems, where a reward should be optimized from the sole information gained after turning the arm of the bandit. This presents a more robust approach, since a learned model can be thoroughly tested before deployment. The system will also not evolve over time, which could lead to unpredictable behavior. This is particularly undesirable in the context of vehicle routing.

The following sections outline how POEM[1] can be used to predict congestion in road networks. The results of POEM are then utilized to dynamically route vehicles around congested areas using A*.

3.1 Learning Setting

The choices a vehicle takes at each intersection are made according to a specific policy. The Nash equilibrium [18] finds a local optimum amongst all policies (using complete knowledge on future traffic demand) such that no vehicle may gain any advantage over this policy by altering its own policy, whereas the central idea of the reinforcement learning algorithm POEM [21] is to use logged data to improve an existent policy h_0.

In [21] POEM assigns a structured output to an arbitrary input based on its probability of being correct. Therefore, before applying POEM to congestion avoidance, a suitable mapping of the routing problem to a policy h_0, along with an input space X and output space Y must be modeled. Additionally, a cardinal loss feedback mapping δ is required, which serves as the reward function about all selected input/output combinations.

The input space X was chosen as $X := [0,1]^m$. Here, each $\vec{x} = (x_1, \ldots, x_m)^T \in X$ represents a feature vector of (normalized) sensor measurements for a road segment. For instance, a road's density, occupancy, mean speed, vehicle count or waiting time can be used. Any value not in $[0,1]$ was scaled using min-max scaling.

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 a road is congested or not already provided adequate results. Thus, let $Y := \{(0),(1)\}$, where (0) indicates a road is not congested, and (1) corresponds to congestion, respectively.

The policy $h_0(Y \mid \vec{x})$ is a probability distribution over the output space. In other words,

[1] For implementation and more theoretical information on POEM, we point the interested reader to [21] and the website at `http://www.cs.cornell.edu/~adith/POEM/index.html`.

it assigns a probability to each output $\vec{y}$ given any input $\vec{x}$ based on how likely $\vec{y}$ is to be correct under conditions $\vec{x}$. Hence, predictions are made by sampling $\vec{y} \sim h_0(Y \mid \vec{x})$. The goal of POEM is then to improve this policy. Initially 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 (compare [21]). Let $h_0(\vec{y} \mid \vec{x}) := 0.5$, meaning both labels are assigned a probability of 0.5 for all $\vec{x}$.

Lastly, in order to improve an existing policy, POEM requires a cardinal loss feedback mapping $\delta : X \times Y \to \mathbb{R}$. This was achieved by applying one of the following two primitive congestion detection methods to the sensor readings: 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 [2]. 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 maximum speed.

3.2 Application

In order to not only detect congestion but also reduce it, vehicles must receive frequent information updates about the current state of the road network. Then, POEM will be used to predict the next state of the road network. This information will consequently be used by vehicles to bypass roads which are deemed congested. Thus, those results must also be applied in a routing algorithm, such as Dijkstra or A*.

Let $G = (V, E, c, q)$ be a graph representing a road network. Here, c and q are the default cost and heuristic functions. Additionally, assume all vehicles have knowledge about a congestion labeling policy $h \in \mathcal{H}_{lin} \cup \{h_0\}$ [21]. When using dynamic routing, vehicles will receive updates about roads in regular intervals $T \in \mathbb{N}$. The update can then be written as $u_T : E \mapsto X$.[2] Then, when a vehicle receives update u_T it is able to predict how likely a road is to be congested during interval $T + 1$ using h.

The described model receives sensor information only about whole road segments, rather than individual lanes, which might be problematic, as congestion does not always arise on every lane equally. That challenging situation is most likely to occur at junctions where each lane will allow a vehicle to go in a different direction. We address this problem by aggregating sensor data for each connected edge pair (using a line graph of G, compare [7]). Additionally, the resulting data allows more precise congestion detection as individual turning lanes are separated in the model.

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 which will likely be congested:

$$p^0_{(e_1,e_2)} := h((0) \mid 0.5u_T(e_1) + 0.5u_T(e_2)) \,, \tag{1}$$

$$c' : E^2 \to \mathbb{R}_+, \ (e_1, e_2) \mapsto \frac{c(e_2)}{p^0_{(e_1,e_2)}} \,. \tag{2}$$

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

The function c' calculates the new weight of an edge e_2 depending on its preceding edge it was reached by. For instance, a vehicle on an edge $e_1 = (u, v)$ would calculate the weight

[2] Here, it is assumed updates are received equally for all edges.

for edge $e_2 = (v, w)$ using $c'(e_1, e_2)$. A vehicle which starts its route on edge e_2 would use $c'(e_2, e_2)$.

Essentially, c' divides the default weight of an edge by its probability of not being congested in interval $T + 1$. This 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.

Finally, it was assumed sensor data updates are available for every road. In real-world road networks permanently installed sensors are much more scarcely distributed throughout the network. This problem can be partly alleviated by directly implementing sensors in the vehicle (e.g. using navigation applications provided by smartphones, or self-driving cars). However, some roads will still remain uncovered. Here, u_T can map to $\{0\}^m$. For the previously defined features in X (a road's density, occupancy, mean speed, vehicle count and waiting time) its dimension m would equal to 5. This will cause h to assign a probability of 0.5 to both labels (as defined by $\mathcal{H}_{lin}$ in [21]). Another solution could be to map u_T to the average of all sensor readings in an interval. Thus, uncovered roads would reflect the average state of a road network.

3.3 Logging

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

$$D := \{(\vec{x}_i, \vec{y}_i, \delta_i, p_i) \mid i \in \mathbb{N}_{\leqslant n}\},\ p_i = h(\vec{y}_i \mid \vec{x}_i)\ . \tag{3}$$

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 shortest routes for any admissible heuristic. Additionally, POEM is initially applied using the default policy h_0, which will scale all weights equally by a factor of two. The scaling will 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.

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

$$\vec{y}_i = \begin{cases} (0),\ h((0) \mid \vec{x}_i) > 0.5\,, \\ (1),\ h((1) \mid \vec{x}_i) > 0.5\,, \\ random((0),(1)),\ otherwise\,. \end{cases} \tag{4}$$

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

In the next section we test the performance in three cases: complete knowledge, stationary sensors, and moving sensors.

4 Experiments

The deployment of our self-organizing routing algorithm in an urban area could be done in two ways. Either the data of an existing stationary traffic information system is used (e.g. a

SCATS [10] system) and fed into a navigation platform that can be used by the citizens. The other option is to turn vehicles directly into sensors and retrieve segment-wise statistics on travel-time, density and traffic flow directly from the navigation app. In the latter case, one might be worried about individual privacy because mobility statistics are recorded centrally, however recent work [12] provides an approach to protect individual privacy in this case using homeomorphic encryption. This approach encrypts the data such that it still allows for analysis on the cryptotext but just the result can be decrypted. In the following we will test these two deployment settings using stationary and moving sensors and compare it to Nash equilibrium and uninformed routing.

For comparability of experiments with different routing algorithms it is essential to guarantee the same traffic demand (i.e. origin/destination pairs) over time. For repeatability of the same origin/destination setting, we perform analysis with a microscopic traffic simulator, SUMO [11]. The simulator models individual vehicles (on a microscopic level, so it controls also acceleration and deceleration) and is largely applied in traffic simulation and applications. It allows us to control traffic demand and provides us complete knowledge on the performance of the street network and on the routing performance. In contrast to arbitrary toy experiments, we aim at modeling sound traffic scenarios, thus, we use an open simulation scenario in the city of Luxembourg [3] which enables reproduction of 24 hours of mobility in this city.

The common procedure of SUMO is to generate the route of each vehicle before the simulation starts, which is why its live routing capabilities are rather limited. However, SUMO provides the *Traffic Control Interface* (TraCI), a network interface which allows full control over the current simulation. We used this to implement a Java application (SUMO-CA) which simulates a central authority. In order to calculate vehicle routes, SUMO-CA loads a road network and converts it to a directed, weighted multi-graph. When running a simulation, SUMO-CA will receive and parse sensor measurements in regular intervals. This information is utilized to predict the next state of the road network using POEM (compare Section 3). Finally, those results are used to update vehicle routes as shown in Section 3.2. SUMO-CA is additionally capable of logging the dataset discussed in Section 3.3.

Unless stated otherwise, each experiment will start at 7:45 o'clock (simulation time) and runs 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. Finally, in order to create more realistic jams on arterial roads, SUMO was set to scale the original demand by a factor of 1.3.

4.1 Measuring Relative-Weighted Difference

Evaluating vehicle detours is problematic. Neither absolute nor relative differences will adequately represent measured detours. The reasoning behind this is that long routes will allow longer, absolute detours, whereas, short routes will allow longer, relative detours. Hence, a different metric is required. We propose usage of the weighted relative detour as follows.

Let $y_A, y_B \in \mathbb{R}_+^*$ be arbitrary measurements of one vehicle when algorithms A and B were applied respectively. Then weighted relative detour $diff_{rw}$ will then calculate the relative difference, while at the same time weighting it using the absolute difference.

$$diff_{rw}(y_A, y_B) := |y_A - y_B| * \frac{y_A - y_B}{y_A + y_B} \tag{5}$$

4.2 Charts

Various charts present the evaluation results.[3] The x-axis shows multiple methods which were evaluated. The baseline is a uninformed uniform cost search (UCS), where each road was assigned its static, default weight and every vehicle chooses its path individually by A*. In this case congestions are likely to appear. Next, our approach with all evaluated rerouting interval sizes is presented. Lastly, a Nash equilibrium (NASH) is shown as a baseline.

On the y-axis we use different metrics. Vehicle throughput is measured in number of cars that reach their goal within the simulation time. Edge travel time, trip detour duration, and trip wait time duration are, if not stated otherwise, denoted in seconds. Edge travel time is a traffic network indicator and denotes the travel time per street segment. Trip detour duration highlights the deviation of each traveled trip in comparison to the UCS routing. The wait time is the time the vehicle is actually waiting in a jam.

In order to adequately present distributions box plots [23, 17] were used. Here, boxes represent the lower, middle and upper quartiles, whereas whiskers will represent the second and 98th percentiles. The outliers were omitted in the graphs.

4.3 Experiment One – Complete Knowledge

This experiment will route 100%, 75%, 50% and 25% of vehicles, chosen randomly, uniformly distributed, using 100% of available live sensor data. In other words, every road is equipped with a permanent sensor, which measures its vehicle count, average speed, occupancy, density and waiting time. This represents the best case scenario regarding information availability. The feedback was created using $\delta_{density}$.

The results show that vehicle throughput increases considerably, even at a usage rate of 25%, which can be seen in Figure 2. It additionally confirms that long update intervals may cause more congestion at high penetration rates. This is particularly visible in Figures 2, 3 and 4.

4.4 Experiment Two – Stationary Sensors

The results in previous experiment one were achieved by placing a sensor on every road. In real-world, sensors are much more scarcely distributed throughout the network [10]. This scenario with stationary traffic sensors is evaluated in experiment two and described in this section. Here, we evaluate sensor coverage of 25% or 10% of the roads. The locations were chosen randomly, uniformly distributed. Just like in experiment one, each sensor measures every vehicle. However, just 40% of vehicles will receive navigation updates, which is a more attainable penetration rate of navigation systems.

The results of experiment two, depicted in Figures 5, 6 and 7, reveal that even at lower penetration rates, the POEM algorithm successfully labels congested roads. Although the travel time per edge does not decrease as noticeable as it did in experiment one, road users still have a time-wise advantage. The results could possibly be improved by placing sensors not evenly throughout the road network, but rather around congestion prone areas. This sensor placement is subject to future research. Here we assume that the intelligent traffic system (pertaining the traffic loops) is already installed in the city (as is in most major cities) and situation of the loops cannot easily be altered.

[3] The charts are placed at the end of the paper in two-column layout.

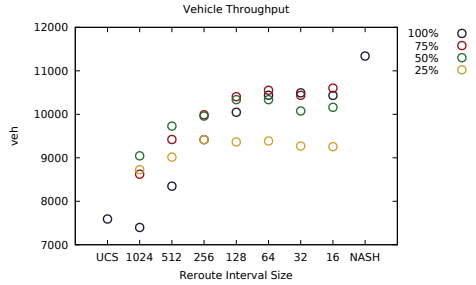

Figure 2 This chart shows results of experiment one. For an interval size of 64 seconds, throughput increases by over 50% when 100%–50% of vehicles were rerouted. Interestingly, an interval size of 1024 seconds noticeably decreased throughput when 100% of vehicles were rerouted. Here, vehicles are rerouted only once and most likely chose similar diversions on small byways, which creates more congestion.

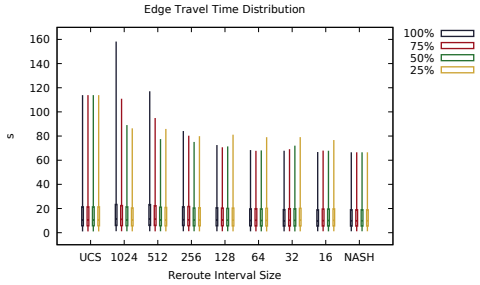

Figure 3 This chart shows results of experiment one. The noticeable increase in the 98th percentile for a routing interval of 1024 seconds where 100% of vehicles were routed coincides with results in chart 2. The same applies to the gradual decrease in travel times per edge.

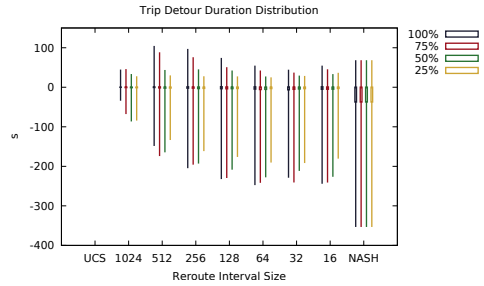

Figure 4 This chart shows results of experiment one. Here, the relative, weighted detours with respect to UCS are presented. It shows most vehicles are unaffected. However, it also shows the benefits considerably outweigh the drawbacks.

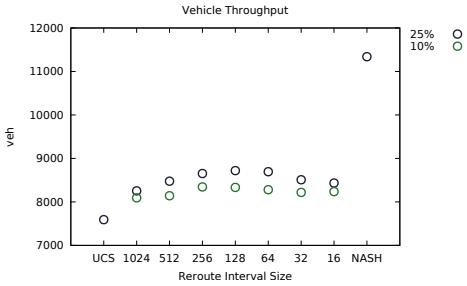

Figure 5 This chart shows results of experiment two. The chart shows that even with sensor data collected at only 10% of all roads, throughput was increased by as much as 10%.

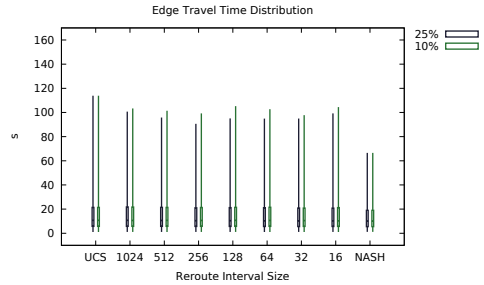

Figure 6 This chart shows results of experiment two. Although a general decrease in travel time per edge can be seen, it is not as noticeable as it was with a sensor on each road, shown in figure 3. However, this can likely be improved by placing sensors on particularly congested roads.

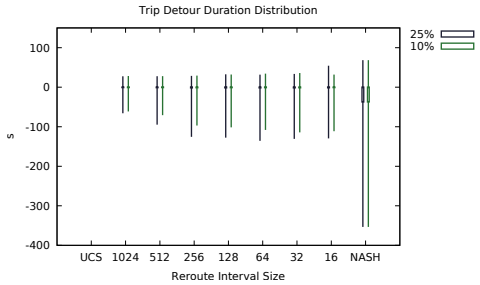

Figure 7 This chart shows results of experiment two. Here, the relative, weighted detours with respect to UCS are presented. Again, the benefits outweigh the drawbacks.

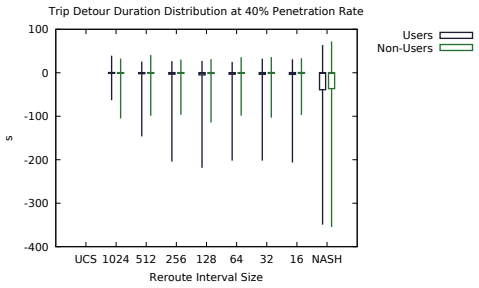

Figure 8 This chart shows results of experiment three. The users of such a navigation system have a clear advantage over non-users compared to UCS. However, non-users also benefit noticeably.

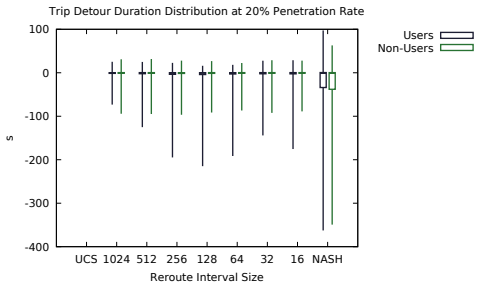

Figure 9 This chart shows results of experiment three. It can be seen that a penetration rate of 20% still provides users with similar advantages. However, performance is considerably lower for smaller interval sizes.

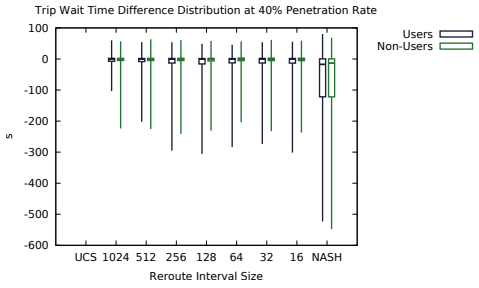

Figure 10 This chart shows results of experiment three. For smaller interval sizes, a user's waiting time will generally decrease more than that of a non-user. However, it is considerably outperformed by a Nash equilibrium.

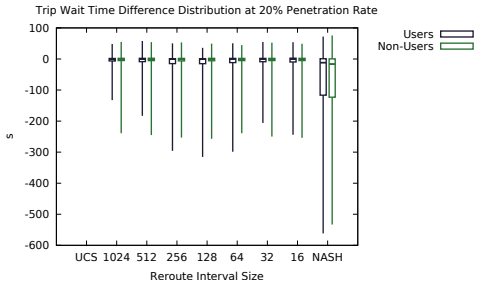

Figure 11 This chart shows results of experiment three. The chart shows that even at a lower penetration rate of 20%, it is still possible for users to outperform non-users with respect to waiting time.

4.5 Experiment Three – Moving 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 an induction loop. Alternatively, smartphone navigation applications can be used as sensors. The measured data will be incomplete, as its only collected for a subset of roads and vehicles. The incompleteness results in erroneous measurements of density values. However, measured vehicle mean speeds will remain largely unaffected. Hence, δ_{speed} was used as feedback. The experiment will evaluate penetration rates of such applications of 40% and 20%.

For vendors of such applications, routing performance regarding all users is not particularly interesting. Their interest mostly focuses on how great of an advantage users will have compared to non-users. This is why the focus lies primarily on those results.

Figures 8 and 9 show that, compared to UCS, users and non-users of such an application would benefit from its use. However, users will have a considerably greater advantage. Figures 10 and 11 show that, when looking only at medium sized intervals, time spent waiting due to congestion or traffic lights also decreases more for users. Generally, larger intervals perform poorly compared to UCS. Here, many vehicles most likely chose similar diversions and did not distribute evenly throughout the network. In turn more congestion is created on low-capacity roads.

5 Discussion and Future Work

Previous experiments revealed that the application of reinforcement learning to the routing problem is beneficial. All our experiments highlight that, by usage of our self-organizing routing regime, performance of the traffic network is increased and occurrence of traffic jams are reduced. We used regular update intervals at which the route could become updated, but this does not imply that in each step the route is altered. However, in order to achieve acceptance of the users, the travel times per user have to be reduced. Future research has to show whether or not the models learned in one geographic region could be transferred to another; we plan evaluation in the city of Cologne. In this case the demand data and road network is provided by TAPAS Cologne [24]. However, the road network quality in this scenario is considerably worse than that of Luxembourg, as it was not manually revised, but rather is a raw *OpenStreetMap* import.

High performing navigation systems have many advantages. An individual person benefits directly, as more precise predictions would shorten travel times and simplify travel planning. This also applies to logistics providers, which could optimize routing schedules and thus increase overall performance. These individual advantages would then extend to the community. Self-organized routes decrease overall traffic volume and congestion, which would serve not only the people, but also the environment [14].

References

1 Jean-Yves Audibert and Rémi Munos. Introduction to bandits: Algorithms and theory. *ICML Tutorial on bandits*, 2011.

2 Harideo 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.

3 Lara Codeca, Raphaël Frank, and Thomas 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.

4 Serdar Çolak, Antonio Lima, and Marta C González. Understanding congested travel in urban areas. *Nature communications*, 7, 2016.

5 Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

6 Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *International Workshop on Experimental and Efficient Algorithms*, pages 319–333. Springer, 2008.

7 F. Harary and R. Norman. Some properties of line digraphs. *Rendiconti del Circolo Matematico di Palermo*, 9(2):161–168, May 1960. doi:10.1007/BF02854581.

8 Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

9 S. P. Hoogendoorn, P. H. L. Bovy, and W. Daamen. Microscopic Pedestrian Wayfinding and Dynamics Modelling. In M. Schreckenberg and S. D. Sharma, editors, *Pedestrian and Evacuation Dynamics*, pages 123–155, 2002.

10 Dermot Kinane, François Schnitzler, Shie Mannor, Thomas Liebig, Katharina Morik, Jakub Marecek, Bernard Gorman, Nikolaos Zygouras, Yannis Katakis, Vana Kalogeraki, et al. Intelligent synthesis and real-time response using massive streaming of heterogeneous data (insight) and its anticipated effect on intelligent transport systems (its) in dublin city, ireland. *ITS, Dresden, Germany*, 2014.

11 Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura 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.

12 Thomas Liebig. Privacy preserving centralized counting of moving objects. In Fernando Bacao, Maribel Yasmina Santos, and Marco Painho, editors, *AGILE 2015*, Lecture Notes in Geoinformation and Cartography, pages 91–103. Springer International Publishing, 2015. doi:10.1007/978-3-319-16787-9_6.

13 Thomas Liebig, Nico Piatkowski, Christian Bockermann, and Katharina Morik. Dynamic route planning with real-time traffic predictions. *Information Systems*, 2016.

14 Ruilin Liu, Hongzhang Liu, Daehan Kwak, Yong Xiang, Cristian Borcea, Badri Nath, and Liviu Iftode. Themis: A participatory navigation system for balanced traffic routing. In *2014 IEEE Vehicular Networking Conference (VNC)*, pages 159–166. IEEE, 2014.

15 Jakub Mareček, Robert Shorten, and Jia Yuan Yu. Signalling and obfuscation for congestion control. *International Journal of Control*, 88(10):2086–2096, 2015.

16 Jakub Mareček, Robert Shorten, and Jia Yuan Yu. r-extreme signalling for congestion control. *International Journal of Control*, pages 1–13, 2016.

17 R. McGill, J. W. Tukey, and W. A. Larsen. Variations of Box Plots. *The American Statistician*, 32(1):12–16, 1978.

18 John Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

19 Tim Roughgarden and Éva Tardos. How bad is selfish routing? *Journal of the ACM (JACM)*, 49(2):236–259, 2002.

20 Marco Stolpe, Thomas Liebig, and Katharina Morik. Communication-efficient learning of traffic flow in a network of wireless presence sensors. In *Proceedings of the Workshop on Parallel and Distributed Computing for Knowledge Discovery in Data Bases (PDCKDD 2015)*, CEUR Workshop Proceedings, page (to appear). CEUR-WS, 2015.

21 Adith Swaminathan and Thorsten Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research*, 16:1731–1755, 2015.

22 Remi Tachet, Paolo Santi, Stanislav Sobolevsky, Luis Ignacio Reyes-Castro, Emilio Frazzoli, Dirk Helbing, and Carlo Ratti. Revisiting street intersections using slot-based systems. *PloS one*, 11(3):e0149607, 2016.

23 J. W. Tukey. *Exploratory Data Analysis*. Number 1 in Exploratory Data Analysis. Addison Wesley Publishing Company, 1970.

24 Sandesh Uppoor, Oscar Trullols-Cruces, Marco Fiore, and Jose M. Barcelo-Ordinas. Generation and analysis of a large-scale urban vehicular mobility dataset. *IEEE Transactions on Mobile Computing*, 13(5):1061–1075, 2014.