To Re-Route, or not to Re-Route: Impact of Real-Time Re-Routing in Urban Road Networks

Mohamed Amine Falek, Antoine Gallais, Cristel Pelsser, Sébastien Julien, Fabrice Theoleyre

To Re-Route, or not to Re-Route: Impact of Real-Time Re-Routing in Urban Road Networks

Amine M. Falek[a,b], Antoine Gallais[b,c], Cristel Pelsser[b], Sebastien Julien[a], Fabrice Theoleyre[b]

[a]Technology & Strategy group, 4 rue de Dublin, 67300 Schiltigheim, France.
[b]ICube Lab, CNRS / University of Strasbourg, Pole API, Boulevard Sebastien Brant, 67412 Illkirch Cedex, France.
[c]LAMIH lab, CNRS / Polytechnic University Hauts-de-France, Le Mont Houy, 59313 Valenciennes Cedex 9, France.

ARTICLE HISTORY
Compiled May 21, 2020

ABSTRACT
Route planning represents a major challenge with a substantial impact on safety, economy, and even climate. An ever-growing urban population caused a significant increase in commuting times, therefore, stressing the prominence of efficient real-time route planning. In essence, the goal is to compute the fastest route to reach the target location in a realistic environment where traffic conditions are time-evolving. Consequently, a large volume of traffic data is potentially required and the route continuously updated. We thereby address the re-routing problem to answer questions such as *when*, *how often*, and *where* is re-routing worthwhile. We base our study on a real dataset, comprising the travel times of the road segments of New York, London, and Chicago, collected over three months. By exploiting this dataset, we implement an optimal algorithm, able to mimic ideal predictions of road segment speeds in the network. Thereby, allowing us to compute the lower bound of travel-time to serve as a reference against other routing techniques. Mainly, we quantify the achieved travel-time gain of a static, no re-routing, and continuous re-routing strategies. Surprisingly, we find that traffic conditions are sufficiently stable for short time windows, and re-routing a vehicle is very seldom useful when exploiting accurate statistics at departure time. Typically, real-time re-routing should only be triggered during rush hours, for long routes, passing through well-identified road segments.

KEYWORDS
road networks; route planning; real-time data; vehicle re-routing; traffic prediction

1. Introduction

Road network traffic congestion is a well known common cause of anger and frustration. From an economic point of view, congestion has an even more drastic impact. The 2015 urban mobility scorecard (Schrank et al., 2015) reports that urban Americans traveled an extra 6.9 billion hours and had to purchase an extra 3.1 billion gallons of fuel. To reliably arrive on time, travelers had to plan 48 minutes for a trip that should last only 20 minutes in light traffic. The estimated cost of congestion is steeply increasing

Amine M. Falek. Email: a.falek@technologyandstrategy.com - falek@unistra.fr
This work has been done when A. Gallais was employed by the University of Strasbourg

since reported in 1982, with a cost of \$42 billion against a total of \$160 billion in 2015. While solutions exist to control traffic lights dynamically (Li and Sun, 2019), it is highly probable that congestion would still occur.

Route planning algorithms (Bast et al., 2016) compute a *route*, which consists of a sequence of road segments between a departure and arrival locations, and a *travel-time*, denoting its associated end-to-end cost. Historically, mobile navigation devices were used to guide the driver by autonomously computing a suitable route. With the wide adoption of smartphones, though, many applications nowadays benefit from the cloud where all computations are performed (Li et al., 2017).

Route planning requires traffic information (estimated speed, vehicle counts) gathered from various sensors and probes within the network. Some mobile navigation devices rely on the median recorded travel-times of a set of monitored road segments to predict congestion. Smartphones and vehicles which usually embed GPS devices are also exploited by regularly relaying traffic measurements to the cloud so that a real-time model of the road network can be constructed (Ahsani et al., 2019). Real-time traffic information is particularly valuable to reduce the cost of urban freight transportation (Flamini et al., 2018).

Prediction as a service aims at predicting the travel time of each road segment, based on past measurements (Wu et al., 2016). That way, a vehicle would use an alternative, faster route if congestion has been predicted along its primary route. Uncertainty may also be considered to compute a route with a given latest time of arrival (Lee et al., 2019). While long-term prediction algorithms exist in the literature (Chen et al., 2019), short-term predictions are still challenging. Traffic jams are complicated to predict (Hu et al., 2017), and their impact on travel time depends on various local factors (e.g., speed limit, type of street, neighboring local roads).

While vehicle re-routing may reduce the impact of congestion, relying on real-time data can be computationally expensive. Computing a route, directly on a small embedded device is inconvenient as traffic data must be downloaded first, which may correspond to a large volume of data when the target location is geographically far away. Practically, this means that hundreds of weights have to be collected periodically, even if the target is in the same urban area. This represents a massive amount of data when considering a large collection of users. Alternatively, the route computation can be executed directly in the cloud: centralized servers reply in real-time to queries. Thereby continuously re-evaluating the route of each vehicle until it reaches its destination. Consequently, the load in the cloud is roughly proportional to the sampling rate used to refresh the real-time data. In particular, it becomes challenging to exploit route planning algorithms that rely on pre-processed data because of the continuous traffic changes measured on the network.

In this paper, we quantify the benefit of using dynamic shortest path algorithms that re-route vehicles when congestion increases. Indeed, alternative paths may provide a faster route, but can only be identified when using real-time information. For this purpose, we exploit a real dataset comprising the travel times of the road segments of several cities (i.e., New-York, London, Chicago, and Cologne). To compare simulations with these real datasets, we also used the TAPAS (simulated) dataset (Uppoor et al., 2014). The contributions of this paper are as follows:

(1) we quantify the travel-time gain when using real-time data to re-route vehicles, compared to a no re-routing approach that prohibits diverting from the path evaluated at the departure. Our dataset highlights that re-routing is very seldom required real conditions, even during rush hours;

(2) we provide an optimal algorithm to compute a lower bound for the travel time between any pair of locations. The algorithm replays the real dataset to mimic ideal predictions. The gain of using ideal predictions is below 10% compared with a continuous re-routing solution, even for the worst route;
(3) we compare the impact of re-routing based on two types of datasets (a real dataset vs. the simulated TAPAS dataset). Surprisingly, and contrary to current belief, we obtain very different behaviors when studying re-routing strategies. Simulations tend to exacerbate the randomness of travel time. Thus, simulations seem to not accurately capture the complexity of urban road network dynamics, proving the relevance of exploiting real datasets;

The remainder of this paper is organized as follows. Section 2 provides the related work. We then present the routing strategies used in our evaluation in section 3, and detail our models and assumptions. We explain our methodology in section 3.3, as well as the detailed description of our datasets. Section 4 presents our comparative analysis of the travel time gain of each routing strategy. We conclude and discuss future work in section 5.

2. Related Work

Road networks are usually abstracted through graph data structures where vertices and edges represent physical intersections and road segments, respectively. Edge weights are labeled with various metrics, such as distances or travel times required to join two vertices along a given edge.

2.1. Re-routing in road networks

The field of shortest path algorithms flourished in the last decade, with continuous improvement in execution times. Pre-processing consists of transforming the initial graph to reply efficiently to route queries (Bast et al., 2016). However, using real-time information implies that the pre-processing phase has to be re-executed.

Gmira et al. (2019) propose to construct a delivery plan related to the Dynamic Vehicle Routing Problem. Their solution collects speed values in real-time, and update the path for a vehicle only if it becomes infeasible. However, they do not investigate the sub-optimality cost, i.e., a route is not updated if no constraint is violated, even if its travel time is not the smallest one.

Understanding when to re-route vehicles is critical to reducing the re-computation cost. Pan et al. (2013) propose an infrastructure-based approach: when congestion is detected, the system asks nearby vehicles to re-compute their shortest route. Congestion threshold, as well as the set of vehicles to re-route, impact the efficiency of this proposal significantly.

In the Dynamic Shortest Path problem (DSP), a re-routing algorithm tries to update the shortest path to handle multiple edge weight updates (Chan and Yang, 2009). Such an approach is much more efficient than re-executing the shortest path algorithm from scratch.

While these approaches aim to devise the most accurate strategy to trigger route computation, ***the objective of this paper is to instead quantify the gain in travel-time when authorizing redirections or exploiting ideal predictions***.

2.2. Real-time data sources

Collecting travel-time measurements of each road segment in real-time is a challenging objective. We may use the GPS trajectories of a collection of vehicles to deduce the specific travel time of each road segment (Duan et al., 2018; Sanaullah et al., 2016). For this purpose, each trajectory has to be mapped to a set of road segments while minimizing the mismatch ratio (Falek et al., 2018). Ladino et al. (2016) propose to merge heterogeneous data sources (e.g., cameras, induction loops), to create a combined, more accurate dataset. Imputation techniques (Chen et al., 2018) help to reconstruct missing data (e.g., a road segment is not crossed for short time-period).

Recent approaches propose to adopt a synthetic model. Typically, the SUMO simulator (Behrisch et al., 2011) is used to generate the mobility pattern of a large number of devices, using realistic urban points of interest. Travel and Activity PAtterns Simulation (TAPAS) was used to generate the well-known TAPAS-Cologne dataset (Uppoor et al., 2014). However, to the best of our knowledge, no study has been conducted to verify the accuracy of these simulations. In (Kamga et al., 2011), the authors, use simulated data from VISTA (Ziliaskopoulos et al., 1999) to determine the impact of incidents on travel times. Other traffic simulation software such as Dynameq (Dynameq, 2020) and TRANSIMS (Barrett et al., 2002) provide simulated datasets of the travel times in a road network.

In this paper, we highlight in Sections 4 and 4.4 ***the differences obtained between the different approaches when using real (measurement-based) datasets vs. the TAPAS dataset.***

2.3. Traffic prediction

Recently, traffic prediction has received much attention in order to provide *prediction as a service* (Liebig et al., 2017). These techniques try to consider the inherent characteristics of road networks (e.g., flow conservation) to predict future trends accurately. Predictions rely on computationally intensive techniques such as, e.g., bee colony optimization (Dell'Orco et al., 2016), or spatiotemporal random field (Liebig et al., 2017). Alternatively, Wang et al. (2019) propose to predict end-to-end travel times directly but limiting its practical interest for route computation.

However, congestion is highly variable: Coifman and Mallika (2007) highlight that 48% of the congestion is difficult to predict. Li et al. (2014) concludes that accidents are impossible to predict and even complicated to detect early.

Since existing prediction techniques are partly inaccurate and computationally intensive, we aim in this paper to quantify its gain: ***if individual travel times for each road segment*** *can be ideally predicted, is the end-to-end travel time gain significant?*

2.4. Evaluating routing performance on dynamic road networks

Smith et al. (2014) evaluate the impact of accidents on traffic congestion using a vehicular simulation and highlight the need for using actual traffic conditions to predict the travel time.

Wang et al. (2013) provide a performance analysis of different route planning algorithms (i.e., Dijkstra, static A*, dynamic Dijkstra, and dynamic A*) in smart cities. Their experiments use the TAPAS-Cologne dataset (Uppoor et al., 2014), which was built by generating traffic demand using TAPAS and Gawron's algorithm for traffic assignment. They consider a rush hour during a weekday to evaluate the impact of traffic jams.

McArdle et al. (2012) attempt to simulate traffic in the Greater Dublin region. Typically, each vehicle selects its destination according to a radiation model, to model probabilities of interactions between different regions.

Unfortunately, all these performance comparisons use simulations with synthetic models. ***The objective of our paper is instead to use real datasets, with the actual travel time for each segment, at any instant, during very long periods (i.e., a few months).***

3. Methodology

A route planning solution identifies a path to follow for a vehicle. Most solutions can be classified into:

(1) **embedded devices** help to compute a route, sometimes after having downloaded real-time data about the congestion of the road network (Wang et al., 2015);
(2) **cloud**-based infrastructures receive a collection of queries from the vehicles that they have to handle in real-time (Li et al., 2017). Continuously reconsidering the route because of real-time data is also expensive in a cloud serving a large number of customers, where additional resources have to be provisioned.

Each route planning strategy has to solve individual queries, returning a route with minimal travel time. Formally, we define a query as $q = (s, d, t_s) | s, d \in V$ and $t_s \in T$, where s, d and t_s represent the departure location, the destination location and the departure time respectively. Likewise, a route is defined as an ordered list of vertices (road segments) in the graph.

In this paper, we consider the following strategies, ordered by the volume of resources (bandwidth and computation) they require:

(1) **static:** we do not have any knowledge of the actual traffic conditions (i.e., no data is exchanged);
(2) **no re-routing:** we know the travel time of each road segment precisely just before the vehicle leaves its starting point. For the sake of limiting computational cost, the vehicle does not reconsider its decision after departure;
(3) **continuous re-routing:** we have continuous access to the most recent real-time data. A vehicle may be redirected to a different (shorter) route as soon as traffic conditions change;
(4) **prediction based routes:** we are able to predict the traffic conditions perfectly. Consequently, we can identify the shortest ideal route, which constitutes our lower bound;

3.1. Assumptions and Model

A road network is defined by a list of road segments. Each of them consists of a set of consecutive coordinates (latitude and longitude) that define its shape. We use a dynamic directed graph $G_T = (V, E, W_T)$ to represent the road network, where $v \in V$ is a vertex representing a physical intersection of two or more road segments, $e_{ij} \in E$ a directed edge from vertex i to j, and $w_{ij}(t) \in W_T$ the weight assigned to e_{ij}, as a function of time $t \in T$. Table 1 contains definitions of the recurring keywords and

Table 1.: Keywords & symbols definitions.

Keywords & symbols	**Definition**
e_{ij}	Directed edge from vertex i to j
d_{ij}	Length $[m]$ of e_{ij}
$s_{ij}(t)$	Speed $[m/s]$ of e_{ij} at time t
$w_{ij}(t)$	Travel time $[s]$ of e_{ij} at time t
T	Number of timeslots in a given dataset
$\delta t = 60\ sec$	Sampling rate of the datasets
$\Delta t \geq \delta t$	Variable sampling rate
$r_i = \{u, v, .., w\}$	Defines a road segment
$R = q(s, d, t_s)$	Route in reply to the query from vertex s to d at timeslot t_s
$R = \{u, v, w, .., z\}$	A route is an ordered set of vertices
$TTime[q$ or $R]$	Travel time of a query q (or a route R)
$algo$	Designates one of four algorithms: static/no re-routing/continuous re-routing/ideal
$TTime[q, algo]$	Travel time of q using algorithm $algo$
$t \geq 0 \in \mathcal{R}$	Represents the time in seconds
$0 \leq t_k \leq T \in \mathcal{N}$	Timeslot index of a real-time data update

symbols used throughout the paper.

For the sake of clarification, we use the term *dynamic* to refer to a graph whose structure remains the same (no edges are deleted or inserted), but edge weights change over the period T. We consider road networks with traffic congestion, but we neglect the new roads that may appear.

We exploit a dataset where the speed of each road segment is monitored periodically and synchronously. Thus, we denote by *timeslot* the discretized time, during which the speeds for all the road segments remain unchanged. An edge weight $w_{ij}(t) = d_{ij}/s_{ij}(t)$ represents the travel time in seconds required to join the vertices i and j, where d_{ij} and $s_{ij}(t)$ represent the length of the edge e_{ij} and its corresponding speed at time t respectively.

A road segment $r_i = \{u, v, w, .., z\} \in V$ consists of an ordered list of vertices in the graph. When the speed on the road segment r_i changes, it affects the weights of all the edges associated with that road segment. Hence, in our implementation, every edge in the adjacency list points to a specific road segment in the road segments map.

3.2. Route Planning Strategies

We now describe in more detail the route computation algorithms we use in the rest of the paper to evaluate the importance of real-time speed data along with road segments. Real-time information implies that vehicles may change their route when congestion occurs. Practically, drivers may be progressively aware of current incidents because they use different data sources (Kucharski and Gentile, 2019). We neglect here the mutual impact of their decisions, i.e., travel-time may increase if all the drivers take the same decision. We model in the rest of this section different families of routing algorithms, taking into consideration traffic information.

Each strategy corresponds to a given travel time formulation. The here presented route planning strategies (i.e., static, no re-routing, continuous re-routing, and ideal prediction based) respectively deal with travel times that are either time-independent (i.e., no knowledge of traffic conditions), evolution-independent (i.e., knowledge of traffic conditions at the initial computation time only), time-aware (i.e., knowledge of initial traffic conditions and regular updates throughout the journey) or ideal (perfect forecast of traffic conditions).

3.2.1. Static Route Planning

We assume here the system has no knowledge about traffic conditions (e.g., an embedded device disconnected from the Internet). It represents our worst but thrifty strategy and provides a baseline for comparison. Thus, it will use the maximum speed for each road segment to compute the fastest route. The weights are time-independent, and hence w_{ij} represents the travel time from vertex i to j, where s_{ij} is the speed limit in this particular case. For a given query $q(s, d, t_s)$, the departure time t_s is meaningless since the algorithm would always return the same route for a departure from s toward the destination d. We use Dijkstra's algorithm (Dijkstra, 1959) to compute the route with the shortest travel time.

3.2.2. No re-Routing Route Planning

The system has here a complete knowledge of the travel times but never reconsiders its decision. It mimics an embedded navigation system that is disconnected after departure or cloud infrastructure that executes the query only once. Thus, we apply the same strategy as previously, just executing Dijkstra's algorithm, with the travel time of each road segment at departure. This way, we can quantify the gain of using up-to-date information *before* departure.

3.2.3. Continuous re-Routing Route Planning

We continuously reconsider the routing decision by trying to compute a better route with shorter travel time, modeling the approach proposed by Chen et al. (2010). This strategy helps to bypass congested areas when they appear, but it also consumes more resources. If the computation is delocalized, the device has to retrieve all the travel times periodically for its area. In a cloud, this means that the query has to be re-executed continuously, consuming computational resources.

Let Δt be the sampling rate of the real-time traffic data. The route planning algorithm will re-compute the optimal route toward the destination at each crossroad. We use a dynamic version of Dijkstra's algorithm as detailed in algorithm 1:

(1) we use a time-dependent graph model, where the weight of each edge is time-variant. When executing Dijkstra's algorithm, we pick the most recent weights;
(2) the route is reconsidered when a new data sample occurs, i.e., it corresponds to an update of speed data. In that case, the shortest route to the destination from the current position is computed (line 4);
(3) we traverse the graph, following the current route (lines 7-17). We have to verify if we can reach the next crossroad before the next speed sample occurs (line 11);
(4) if we cannot, we have also to consider the upcoming speed updates to reflect the actual travel time (line 14).
(5) when a new sample occurs, we set the upcoming crossroad as the new position (line 18) from which we re-evaluate the route (step 1).

Algorithm 1: Fastest Route with continuous re-Routing.

```
Data: departure vertex ($s$), destination vertex ($d$), departure time ($t_s$), sampling rate ($\Delta t$)
Result: shortest route as an ordered list of vertices ($route$)
/* Set current position to vertex $s$ and current time to $t_s$ */
1  $here \leftarrow s$;
2  $t_{now} \leftarrow t_s$;
3  do
       /* updates the route, from here to the destination */
4      $route \leftarrow dijkstra(here, d, t_{now})$ ;
       /* true if a new data sample occurs */
5      $isNewSample \leftarrow false$;
       /* traverse the route until a new data sample occurs */
6      while !$isNewSample$ do
           /* next edge in shortest route */
7          $e_{vw} \leftarrow getNextEdge(route)$ ;
8          $lg \leftarrow getLength(e_{vw})$          /* get length of edge $e_{vw}$ */
9          while $lg > 0$ do
               /* maximum distance that can be covered by next speed change */
10             $lg_{max} \leftarrow speed(e_{vw}, t_{now}) * (\Delta t - t_{now} \mod \Delta t)$;
11             if $lg < lg_{max}$ then          /* crossroad reached before next sample */
12                 $lg \leftarrow 0$ ;
13                 $t_{now} \leftarrow t_{now} + \frac{lg}{getSpeed(e_{vw}, t_{now})}$;
14             else          /* crossroad not reached before next sample */
15                 $isNewSample \leftarrow true$;
16                 $lg \leftarrow lg - lg_{max}$;
17                 $t_{now} \leftarrow t_{now} + \frac{lg_{max}}{getSpeed(e_{vw}, t_{now})}$;
18         $here \leftarrow getHeadVertex(e_{vw})$;
19 while $here \neq d$;
   /* we reached $d$ */
20 return $(route)$
```

This version accommodates any sampling rate. This way, we can compare the impact of the data accuracy on the travel time.

3.2.4. Ideal Prediction Based Route Planning

As mentioned earlier, traffic congestion forecasting is a technique used by most advanced route planning algorithms. By incorporating predictions, one could predict recurrent traffic jams, and thus, plan the route accordingly at departure time.

We propose here to model such prediction-based routing algorithm (Liu, 2017). More precisely, our goal is to quantify the maximal benefit attainable when using a perfect forecast. Thus, we use an **ideal** prediction algorithm to compute the maximum gain achieved by **any** prediction-based routing algorithm. To do so, we depart a vehicle in the past by *replaying* the recorded measurements *a posteriori*.

We propose to use Algorithm 2:

(1) we first insert the departure vertex into the set of settled vertices (line 1) as the triplet (vertex, parent, arrival time). We insert its neighboring vertices (i.e., head vertices of outgoing edges) into a priority queue keeping the vertex with earlier arrival time at the head;
(2) at each iteration, we poll a vertex v from the queue and insert it into the settled set (line 5);

Algorithm 2: Optimal Route with ideal predicion routing.

```
Data: departure vertex ($s$), destination vertex ($d$), departure time ($t_s$)
Result: shortest route as an ordered list of vertices ($route$)
/* settled is a list containing all vertices for which the shortest route was found */
1  $settled \leftarrow \{(s, s, t_s)\}$; /* (the settled vertex, its parent, arrival time) */
/* priority queue holds vertices that we either did not visit yet ($\infty$) or for which we
   did not find the shortest path */
2  $queue \leftarrow \{(v, -1, \infty) | v \neq s \in V\}$;
3  $route \leftarrow \{d\}$; /* insert destination into route initially */
4  do
       /* extract vertex $v$ with smallest arrival time from the queue and insert it into the
          settled list with departure time $t_v$ */
5      $settled.add((v, u, t_v) \leftarrow queue.poll())$;
       /* iterate over all outgoing edges from v */
6      for $e \in getOutgoingEdges(v)$ do
7          $w \leftarrow getHeadVertex(e_{vw})$;
8          $t_w \leftarrow t_v$;                      /* initialize arrival time at $w$ */
9          $lg \leftarrow getLength(e_{vw})$;          /* get length of $e_{vw}$ as $lg$ */
           /* compute travel time of edge $e_{vw}$ */
10         while $lg > 0$ do
               /* maximum distance which can be covered by next speed change */
11             $lg_{max} \leftarrow speed(e_{vw}, t_w) * (\Delta t - t_w \mod \Delta t)$;
12             if $lg < lg_{max}$ then         /* $w$ is reached before next sample */
13                 $lg \leftarrow 0$ ;
14                 $t_w \leftarrow t_w + \frac{lg}{getSpeed(e_{vw}, t_w)}$;
15             else                      /* next sample occurs before reaching $w$ */
16                 $lg \leftarrow lg - lg_{max}$;
17                 $t_w \leftarrow t_w + \frac{lg_{max}}{getSpeed(e_{vw}, t_w)}$ ;
18         $queue.push((w, v, t_w))$;
19 while $d \notin settled$;
/* browse parent vertices starting at $d$ until $s$ is reached */
20 $parent \leftarrow getParentVertex(d)$;
21 while $parent \neq s$ do
22     $route.add(parent)$; /* insert into head of route */
23     $parent \leftarrow getParentVertex(parent)$;
24 return $(route)$
```

(3) we compute the travel time required to reach each of its neighbors w (lines 7-18). In particular, we traverse each $edge_{vw}$, and we update its speed when a new sample occurs before reaching the next crossroad (lines 15-17);

(4) we re-insert w into the queue with its corresponding parent vertex v and the updated arrival time t_w (line 18);

(5) when the destination vertex is settled, we construct the route by browsing backward all the parent vertices starting at the destination until we reach the departure vertex (lines 20-23).

3.3. Evaluation Workflow

To fairly compare different route planning strategies, we need to compute enough routes using each strategy to cover most roads in the road network. To do so, we generate a massive number of queries and solve each of them with our four routing strategies. Hence, for a given route $R = q(s, d, t_s)$, by varying the departure time t_s, we can track the changes of R as a *vehicle* experiences congestion along the route from s to d, at different times of the day.

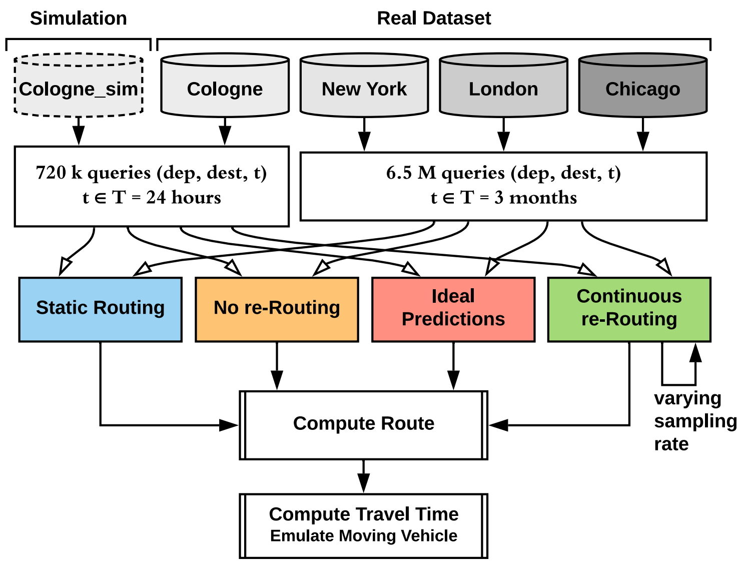


Figure 1.: Workflow for the stretch factor quantification of not using real-time data.

We insist on the fact that we do not simulate the *deployment* of multiple vehicles on the road network at once. Indeed, we do not have any means to incorporate the added congestion due to those vehicles, as one might expect in a mobility simulation tool. Instead, we consider each query as representing a *probe vehicle*, to measure the impact of the measured traffic (from our datasets) on its route (and not the inverse).

We apply here the following workflow to quantify the interest in exploiting real-time data (Figure 1):

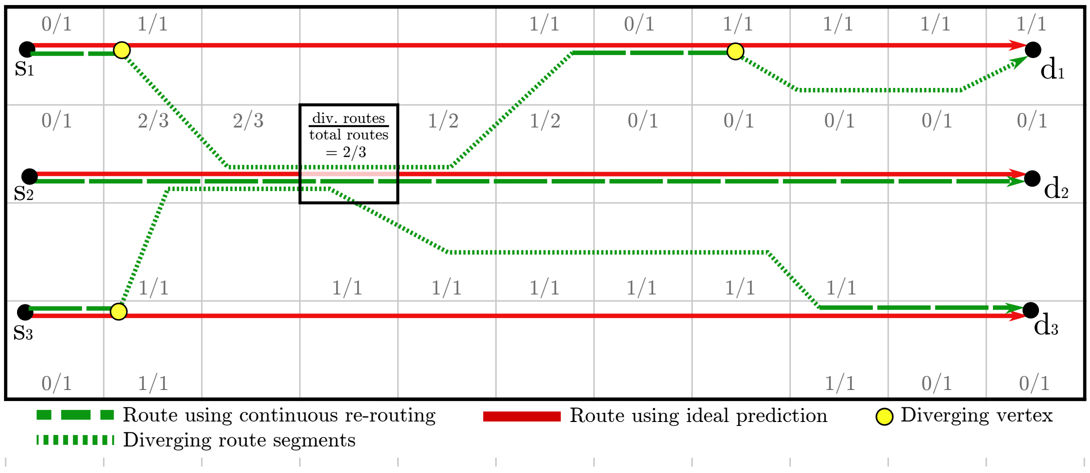
(1) we use real vs. simulated (TAPAS) datasets, and emulate different sampling rates (from 1 to 30 min) by subsampling the datasets;
(2) we randomly select 1000 pairs of source and destination vertices in each road network;
(3) for every (source, destination) pair, we generate a broad set of queries at different timestamps (every 1 min for the simulation and every 10 min on the real dataset);
(4) for each query, we execute each route planning strategy to extract the route to follow;
(5) we then emulate a vehicle moving along the route returned by the algorithm in the previous step. This way, we can accurately evaluate the actual end-to-end travel time for each strategy at different times of the day.

Finally, we use the previous results to compare the static, no re-routing, continuous re-routing and ideal prediction strategies, detailed in sections 3.2.1, 3.2.2, 3.2.3 and 3.2.4 respectively.

We ran all our experiments on the High-Performance Computing (HPC) at the University of Strasbourg. We generated thousands of jobs that were executed on Intel Xeon Sandy-Bridge nodes with 16 cores and 64GO of RAM. The combined computation effort required approximately 50,000 CPU-hours.

3.4. Datasets

To evaluate the impact of the different route planning strategies, we use real travel times for a broad set of road segments. Additionally, we also use a simulated dataset for comparison purposes. We rely upon:



0/1
1/1
2/3
1/2
0/1
div. routes / total routes = 2/3
s1
s2
s3
d1
d2
d3
Route using continuous re-routing
Route using ideal prediction
Diverging vertex
Diverging route segments

Figure 2.: Diagram illustrating the process of computing the divergence ratio of a cell.

(1) **the simulated TAPAS dataset:** (Uppoor et al., 2014) focuses on a small German City (Cologne), during a weekday (24 hours). It relies on the simulator SUMO to simulate the traffic from a large set of emulated vehicles, pseudorandomly selecting pairs of sources and destinations (e.g., home, office). We run the simulation using SUMO to generate a dump file specifying the speed along every road segment at 1-sec intervals. We subsample the dataset to get the same sampling rate used in the real dataset.

(2) **a real dataset:** we use the dataset of three major cities (New York, London, and Chicago) for which HERE (HERE Technologies, 2018) provides a fine-grain estimation of the speeds for the most important road segments. We collected three months (i.e., from September 21st to December 18th 2017) worth of data at a sampling rate of 1 min, which allows an accurate estimation of the actual travel time experienced by users. Additionally, we also use the city of Cologne, for the same geographical area and during the same time window as the TAPAS dataset for a fair comparison.

3.5. Metrics for the Performance Evaluation

We now detail the metrics we used to compare the different routing strategies.

3.5.1. Identification of congestion

To identify the rush hour period, we compute the Congestion Factor (CF) of each query as the ratio of the optimal travel time (using the ideal routing strategy) to the free-flow travel time. In the TAPAS dataset, the free-flow speed of a road segment corresponds to its speed limit. In the real dataset, the free-flow speed was provided as the average measured speed during low-volume periods and depends on the road characteristics such as lane width.

Hence, given a query $R = q(s, d, t_s)$, the congestion factor of the end-to-end route R is defined as:

$$CF[R] = \frac{TTime[R, ideal]}{TTime[R, freeflow]} \tag{1}$$

To analyze more clearly the behavior of each routing strategy, we need to classify the routes according to their congestion level. For this purpose, we define for each route its *traffic flow* (TF) metric, which represents the congestion of its most congested road segment. For each route $R = q(s, d, t_s)$, we measure the level of congestion of each road segment $r_i \in R$. More precisely, the congestion level corresponds to the relative speed decrease compared with free-flow conditions:

$$TF[r] = max\left(\frac{speed(r_i, t)}{speed_{ff}(r_i)}\right)_{r_i \in R} \tag{2}$$

where $speed(r_i, t)$ corresponds to the speed at time t for the road segment r_i, and $speed_{ff}(r_i)$ to its speed in ideal conditions (i.e., free flow). $TF = 1$ corresponds to free-flow conditions, and $TF = 0$ corresponds to a complete halt of all vehicles on one of its road segments.

3.5.2. Travel Time Stretch and Gain Factors

We will use the *stretch factor* in travel time to compare the different strategies. The ideal strategy provides, by definition, the lowest end-to-end travel time and represents our lower bound. The stretch factor for a query $q(s, d, t_s)$ is defined as:

$$SF[q] = \frac{TTime[q, algo]}{TTime[q, ideal])} \tag{3}$$

where $TTime[q, algo]$ represents the end-to-end travel time for the route returned by the algorithm *algo* (static/no re-routing/continuous re-routing/ideal) for the query q. Notice that $SF[q] \geq 1$, and hence, the higher the stretch factor gets, the worse is the performance of algorithm *algo* compared to the ideal.

To specifically focus on the gain achieved by re-routing vehicles after their departure, we also compute the *gain factor*. It corresponds to the relative gain in travel time through the path selected by the prediction-based and the redirecting strategies compared with the static one (i.e., without redirection after the departure).

3.5.3. Identification of divergences

We focus on the continuous re-routing strategy to precisely determine *where* a vehicle is practically re-routed because the congestion has changed. Practically, we identify the *divergence vertices*, i.e., geographical locations where the next edge is different with or without rerouting. Formally, given two routes R_1 and R_2 that share at least the same first and last vertices, we define their diverging vertices as the set $\{j \in V \mid \exists i, k, w \in V \mid (e_{ij}, e_{jk}) \in R_1 \wedge (e_{ij}, e_{jw}) \in R_2 \wedge k \neq w\}$.

We construct a grid overlay where each cell is a 10 $\times$ 10 square meters. We consider a large number of route queries (pairs of sources/destinations). We execute the route computation for each query at different instants covering to cover the whole dataset. For each cell $cell_i$, we consider all the routes that cross this cell. We compute its *divergence ratio* ($0 \leq div[cell_i] \leq 1$) as the ratio of the number of diverging routes to the number of total routes. A route is diverging if the routes with and without the re-routing strategy are different for *at least one instant* of the dataset. Hence, a cell with a high divergence ratio means that more vehicles are re-routed when crossing this cell.

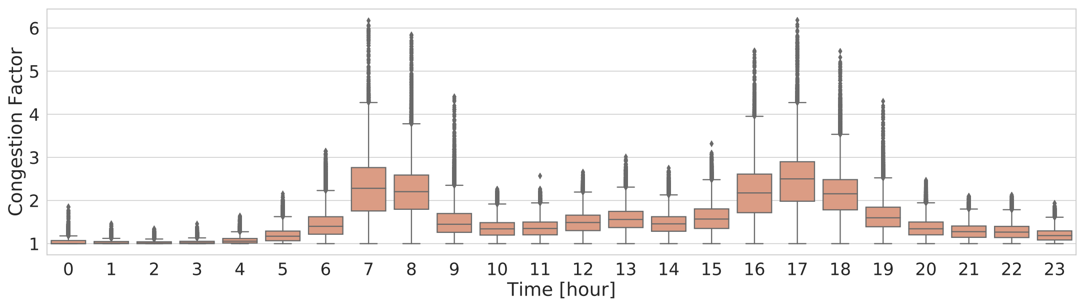


Figure 3.: Congestion Factor distribution in Cologne simulation, Germany, over a period of 24 hours.

Some cells may be traversed by only a few routes, leading to statistically meaningless results. Thus, we only consider cells traversed by a significant number of routes (300 in this case).

4. Results

In a road network with no congestion, all routing strategies should return optimal routes. Hence, our first goal is to distinguish the critical parts of the dataset by identifying the rush hours in each city. The remainder of the paper will solely focus on evaluating the presented routing strategies during rush hours only.

We also provide a visual, interactive interface to showcase a sample of the NYC dataset and obtained results at `http://its-icube.com/`. Each query is represented individually, to visually represent the temporal characteristics of each route all along the week. In particular, we can identify the diverging vertices, as well as the evolution of the congestion.

4.1. Absolute Travel Time and Rush Hours

Figure 3 illustrates the CF in the Cologne TAPAS dataset. As expected, we identify the rush hours in the morning (6:00-9:00) and afternoon (15:00-19:00) of the working day. The optimal travel time is almost six times longer than the free-flow travel time during these two periods.

Figure 4 illustrates the congestion factor for the experimental datasets. We focused uniquely on Thursday, Friday, Saturday, and Sunday to provide readable charts (the remaining weekdays display similar characteristics to Thursday). We can derive the following observations:

(1) some queries benefit from travel times smaller than the free-flow. They correspond to very short distances (a few hundred meters) when streets are empty;
(2) we observe the rise in congestion starting at 6:00 and intensifying in the afternoon with a peak at 17:00.
(3) we can easily make a distinction between working days and weekends, which are much less congested.

Similar observations hold for London and Chicago.

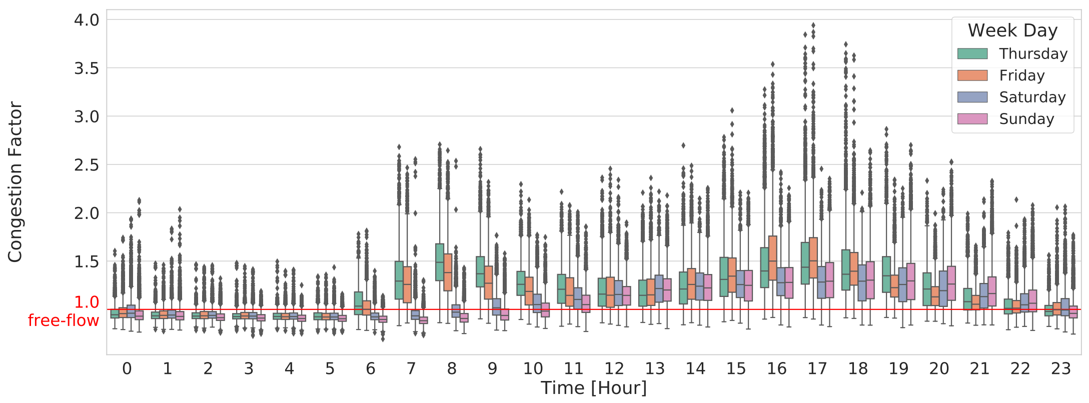


Figure 4.: Congestion Factor distribution in New York, based on weekday and daytime over a period of 3 months.

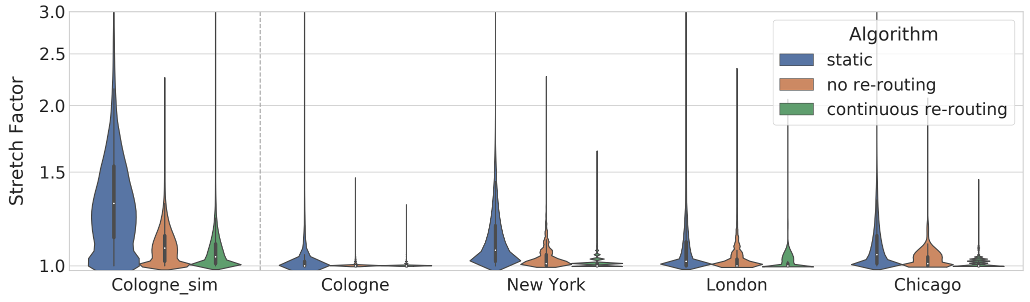


Figure 5.: Stretch factor of the static, no re-routing and continuous re-routing algorithms on both the TAPAS (left) and the real (right) datasets.

4.2. Travel Time Stretch and Gain Factors

We now compare the achieved travel time for each routing strategy. In particular, Fig. 5 illustrates the stretch factor for the TAPAS dataset (left) vs. the real datasets (right). The stretch factor denotes the travel time increase compared with the shortest path, with an ideal dataset (i.e., with ideal predictions). We clipped the plot at $SF = 3$, as it reaches ≈ 6 in the simulation.

The static routing strategy provides, as expected, the longest travel time. However, the distribution is significantly different between the simulation and the real dataset. With TAPAS, 28% of the queries have an SF greater than 1.5. In contrast, we only report 1.6% of the queries in Cologne and approximately 4% in New York, London, and Chicago. In Cologne, the SF using either the no re-routing or continuous re-routing strategies yields optimal travel times most of the time. Only $\approx$ 12% of the queries are characterized with a stretch factor greater than 1. In TAPAS, though, more than 75% of the queries have a $SF > 1$ even when using continuous re-routing. Moreover, for several queries in the simulation, travel time is worse when using continuous re-routing rather than no re-routing. The high variability of congestion profoundly affects re-routing in the simulation as a vehicle might engage in a seemingly faster alternative route only to become more congested than the initial route. In New York, London, and Chicago, approximately 75% of the queries benefit from ideal travel times when using continuous re-routing.

To further comprehend when continuous re-routing is advantageous, we pinpoint

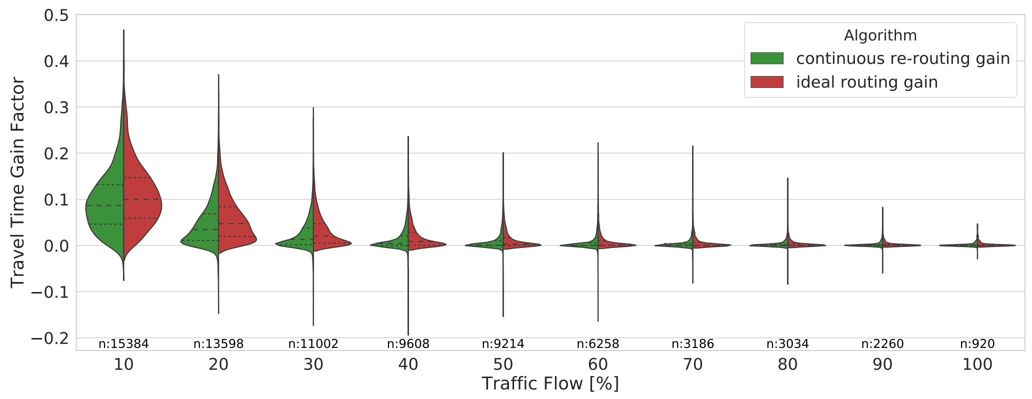


Figure 6.: Travel time gain of continuous re-routing and ideal routing algorithms over the no re-routing algorithm as function of traffic flow in New York

the travel time gain it offers (compared to no re-routing) based on the congestion level (Fig. 6). We normalize the travel time difference of using continuous re-routing (and ideal routing for reference) over the no re-routing strategy.

As expected, re-routing is relevant only when congestion occurs. The gain is, however, often negligible when the actual speed is close to 50% the speed limit. For very congested routes ($TF \leq 10\%$), we reduce the travel time significantly by re-routing the vehicle. Exploiting ideal predictions allows the gain to be even higher, to choose a better route: the best route is selected, before the congestion's increase.

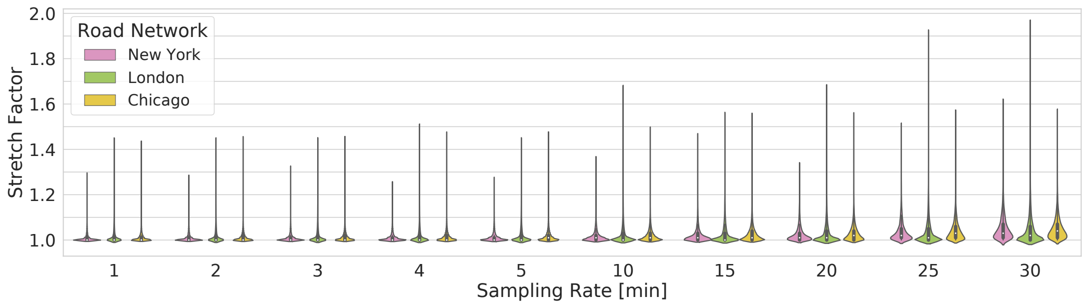
Conclusion: continuous re-routing significantly reduces travel time for most queries. In New York City, we can reduce by 15% (1st quartile), which seems to justify the usage of a smart route planning strategy. However, predictions only marginally decrease the travel time for all the measured datasets. The traffic conditions seem to evolve smoothly, and redirecting the vehicle when the congestion occurs appears as a sufficient strategy. Clearly, continuous re-routing is essential to re-route around highly congested road segments. When the road network is less congested ($TF \geq 40\%$), the gain factor quickly converges to zero.

4.3. Impact of Sampling Rate on Travel Time

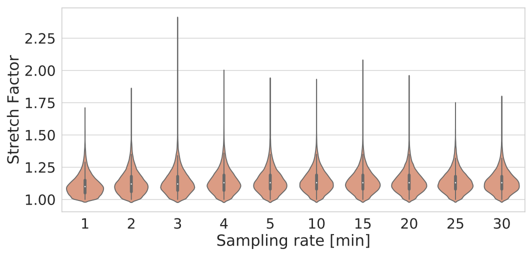
For the continuous re-routing algorithm, the sampling rate Δt impacts both travel and execution times. For fine-grained values, the algorithm is aware of the latest changes in traffic congestion and can re-route accordingly. However, it may also re-compute unnecessarily the routes, increasing execution time significantly. Our goal is thus to determine the right sampling rate for a good travel/execution time trade-off.

By varying the sampling rate Δt from 1 to 30 minutes, we re-compute the queries we generated using continuous re-routing. We only consider queries with a travel time $TTime[q] \geq 60min$ to make sure re-routing has potentially occurred for the largest sampling rate $\Delta t = 30$.

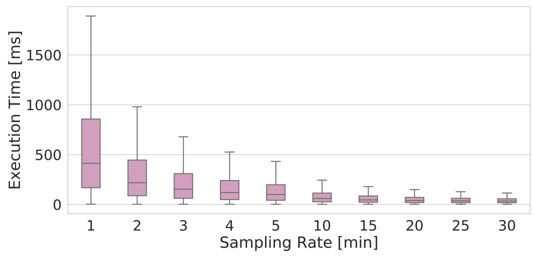
Figure 7a illustrates the distribution of the stretch factor as a function of the sampling rate. Obviously, a higher sampling rate means inaccurate speed values. Thus, the route planning strategy will take sub-optimal decisions. However, the re-routing strategy performs well, even for average sampling rates. Surprisingly, in New York, for instance, the road network variations can be efficiently handled even if measurements are reported every 10 minutes.



(a) Travel time stretch factor in New York, London and Chicago



(b) Cologne simulation stretch factor



(c) New York execution time

Figure 7.: Impact of sampling rate on travel time in Cologne simulation, New York and London (a, c and d) and execution time in Cologne simulation (b) using the continuous re-routing algorithm.

Figure 7c illustrates the execution time for New York. At $\Delta t = 10min$, the execution time drops from $\approx 500ms$ to less than $50ms$, which provides a good trade-off between travel time and execution time. Of course, there exists a myriad of algorithmic techniques in the literature, capable of significantly reducing the execution time. Considering that Δt dictates the number of times we have to re-compute the route, we are only interested here in the rate of the change of execution time as Δt increases.

Figure 7b represents the execution time for the TAPAS dataset. In complete contradiction with the real dataset results, the stretch factor distribution is almost the same regardless of the sampling rate value. Again, we observe that congestion changes too fast throughout the whole road network (even at $\Delta t = 1min$), causing the algorithm to inevitably re-route through highly congested road segments.

Conclusion: we can accommodate average sampling rates when using real-time data. Five minutes provide enough accuracy (for all our road networks) to identify the best routes while reducing the computational or bandwidth cost.

4.4. Route Divergence Patterns

Figure 8 illustrates the distribution of the diverging vertices in each road network, as defined in section 3.5.3. A divergence vertex corresponds to a crossroad where at least one vehicle has been re-routed to reduce travel time (i.e., traffic congestion has changed since its departure). The divergence ratio counts the ratio of routes (source/destination) for which the path diverges when crossing the cell, with and without the re-routing strategy.

The road networks in the real dataset exhibit a small set of diverging vertices with

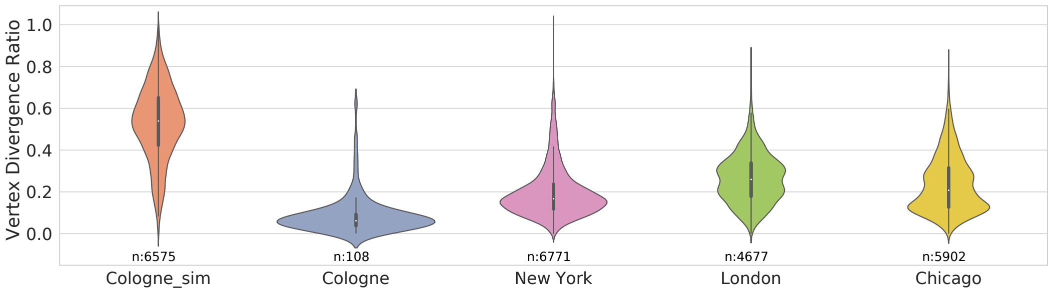
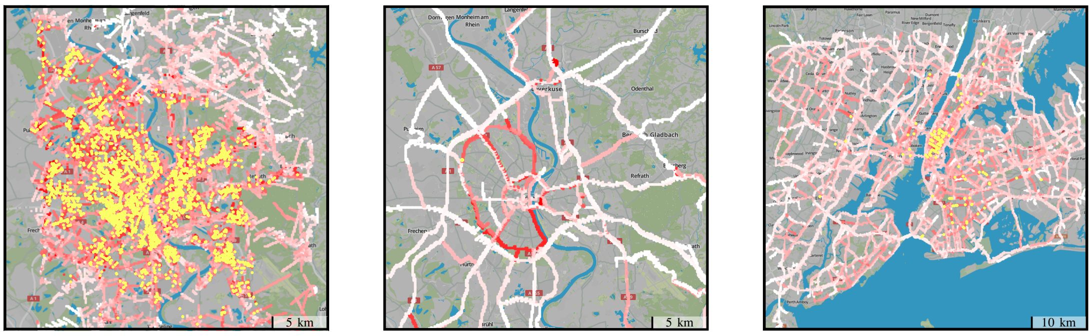


Figure 8.: Diverging vertices divergence ratio distribution.



(a) Cologne simulation (b) Cologne (c) New York

Figure 9.: Diplaying the divergence ratio level (proportional to the red shade intensity). The yellow dots depict diverging vertices with a divergence ratio ≥ 0.5.

a high divergence ratio, typically almost all diverging vertices have a divergence ratio ≤ 0.5. This means that only 50% of the routes that cross these cells are re-routed at least once during the whole duration of the dataset.

Inversely, the simulated dataset (with TAPAS) exhibits a significant number of diverging vertices. TAPAS generates pseudorandomly the traffic and estimates the level of congestion, and the speed for each road segment. It seems that TAPAS exhibits a very different pattern compared with the real datasets.

Figure 9 summarizes the obtained divergence patterns for both the TAPAS and real datasets. We only represent here New-York City since other real datasets behave similarly. Each graph corresponds to a heat map (a red cell corresponds to a cell with a high divergence ratio). We also highlighted the divergence vertices with a divergence ratio ≥ 0.5 (yellow dots).

Remark: while we identify many cells with a high divergence ratio for all the graphs, we can, however, still make a distinction between the TAPAS and the real datasets. In the TAPAS dataset of Cologne, the divergence seems present everywhere. We assume that this behavior comes from the fact that the source/destination of each vehicle is picked pseudorandomly, and the shortest route is used. Individual routing strategies seem more complex, and the trips seem to follow a uniform distribution.

Conclusion: many cells exhibit a large divergence ratio, and we cannot directly use this metric to trigger the route re-computation efficiently. However, we identified only a small number of diverging vertices. Thus, we would be able to execute the computation only when the route crosses these specific points, making the computation much more efficient. We would reduce the processing load without increasing the end-to-end travel time.

5. Conclusion & Perspectives

Modern intelligent systems guide vehicles through the fastest routes to avoid congested areas. However, they need to exploit real-time data, where the speed of each road segment has to be known precisely. Even worse, this real-time feature has a cost, in bandwidth (data collection), and computation (re-computation of the route to the destination). Interestingly, the no re-routing strategy provides close to ideal travel times most of the time. Using the continuous re-routing strategy can further improve travel time by avoiding very congested areas when they appear.

We also used a simulated dataset (TAPAS) that leads to very different results concerning the travel time and the re-rerouting gain. TAPAS seems not able to capture the road network characteristics accurately, and particularly its dynamics. This observation speaks in favor of working with real datasets to model realistic environments.

In future work, we plan to enhance the use of real-time data in our route planning strategies. We considered here that each vehicle triggers a route computation after reaching each crossroad (if the speed has changed). We may improve the re-routing strategy by triggering a computation only when the routes may diverge. We identified specific crossroads where vehicles have a higher probability of being re-routed. We only considered selfish decisions in this study, i.e., each vehicle decides by itself when it should re-compute its route to the destination. To reduce the congestion, we may consider a more collective objective, redirecting *some* of the vehicles . Thus, we expect to propose a strategy that could also consider the gain in travel time, depending on the characteristics of the route and of the local area.

References

Ahsani, V., Amin-Naseri, M., Knickerbocker, S., & Sharma, A. (2019). Quantitative analysis of probe data characteristics: Coverage, speed bias and congestion detection precision. *Journal of Intelligent Transportation Systems*, 23(2):103–119. doi:10.1080/15472450.2018.1502667.

Barrett, C., Bisset, K., Jacob, R., Konjevod, G., & Marathe, M. (2002). Classical and contemporary shortest path problems in road networks: Implementation and experimental analysis of the transims router. In: *European Symposium on Algorithms (ESA)*, pages 126–138. doi:10.1007/3-540-45749-6_15.

Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P., Wagner, D., & Werneck, R. F. (2016). *Algorithm engineering*, chapter Route Planning in Transportation Networks, pages 19–80. Springer.

Behrisch, M., Bieker, L., Erdmann, J., & Krajzewicz, D. (2011). Sumo–simulation of urban mobility. In: *The Third International Conference on Advances in System Simulation (SIMUL 2011), Barcelona, Spain*, volume 42.

Chan, E. P. F. & Yang, Y. (2009). Shortest path tree computation in dynamic graphs. *IEEE Transactions on Computers*, 58(4):541–557. doi:10.1109/TC.2008.198.

Chen, C., Jiao, S., Zhang, S., Liu, W., Feng, L., & Wang, Y. (2018). Tripimputor: Real-time imputing taxi trip purpose leveraging multi-sourced urban data. *IEEE Transactions on Intelligent Transportation Systems*, 19(10):3292–3304. doi:10.1109/TITS.2017.2771231.

Chen, C.-M., Liang, C.-C., & Chu, C.-P. (2019). Long-term travel time prediction using gradient boosting. *Journal of Intelligent Transportation Systems*, 0(0):1–16. doi:10.1080/15472450.2018.1542304.

Chen, Y., Bell, M. G. H., & Bogenberger, K. (2010). Risk-averse autonomous route guidance by a constrained a* search. *Journal of Intelligent Transportation Systems*, 14(3):188–196. doi:10.1080/15472450.2010.484753.

Coifman, B. A. & Mallika, R. (2007). Distributed surveillance on freeways emphasizing incident detection and verification. *Transportation research part A: policy and practice*, 41(8):750–767. doi:10.1016/j.tra.2006.12.001.

Dell'Orco, M., Marinelli, M., & Silgu, M. A. (2016). Bee colony optimization for innovative travel time estimation, based on a mesoscopic traffic assignment model. *Transportation Research Part C: Emerging Technologies*, 66:48 – 60. doi:10.1016/j.trc.2015.10.001, Advanced Network Traffic Management: From dynamic state estimation to traffic control.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271. doi:10.1007/BF01386390.

Duan, Z., Yang, Y., Zhang, K., Ni, Y., & Bajgain, S. (2018). Improved deep hybrid networks for urban traffic flow prediction using trajectory data. *IEEE Access*, 6:31820–31827. doi:10.1109/ACCESS.2018.2845863.

Dynameq (2020). Traffic simulation software your city can plan on. `https://www.inrosoftware.com/en/products/dynameq/`.

Falek, M., Pelsser, C., Gallais, A., Julien, S., & Theoleyre, F. (2018). Unambiguous, real-time and accurate map matching for multiple sensing sources. In: *International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEE. doi:10.1109/WiMOB.2018.8589103.

Flamini, M., Nigro, M., & Pacciarelli, D. (2018). The value of real-time traffic information in urban freight distribution. *Journal of Intelligent Transportation Systems*, 22(1):26–39. doi:10.1080/15472450.2017.1309530.

Gmira, M., Gendreau, M., Lodi, A., & Potvin, J.-Y. (2019). Managing in real-time a vehicle routing plan with time-dependent travel times on a road network. Technical Report 2019-4, CIRRELT.

HERE Technologies (2018). Real-time traffic. `https://www.here.com`. [Online; accessed May 2019].

Hu, W., Yan, L., Wang, H., Du, B., & Tao, D. (2017). Real-time traffic jams prediction inspired by biham, middleton and levine (bml) model. *Information Sciences*, 381:209 – 228. doi:10.1016/j.ins.2016.11.023.

Kamga, C., Mouskos, K., & Paaswell, R. (2011). A methodology to estimate travel time using dynamic traffic assignment (DTA) under incident conditions. *Transportation Research Part C: Emerging Technologies*, 19:1215–1224. doi:10.1016/j.trc.2011.02.004.

Kucharski, R. & Gentile, G. (2019). Simulation of rerouting phenomena in dynamic traffic assignment with the information comply model. *Transportation Research Part B: Methodological*, 126:414 – 441. doi:10.1016/j.trb.2018.12.001.

Ladino, A., Kibangou, A., Fourati, H., & de Wit, C. C. (2016). Travel time forecasting from clustered time series via optimal fusion strategy. In: *European Control Conference (ECC)*, pages 2234–2239. doi:10.1109/ECC.2016.7810623.

Lee, H., Choi, S., Jung, H., Park, B. B., & Son, S. H. (2019). A route guidance system considering travel time unreliability. *Journal of Intelligent Transportation Systems*, 23(3):282–299. doi:10.1080/15472450.2018.1542303.

Li, X. & Sun, J.-Q. (2019). Multi-objective optimal predictive control of signals in urban traffic network. *Journal of Intelligent Transportation Systems*, 23(4):370–388. doi:10.1080/15472450.2018.1504294.

Li, Y., Jin, D., Hui, P., Wang, Z., & Chen, S. (2014). Limits of predictability for large-scale urban vehicular mobility. *IEEE Transactions on Intelligent Transportation Systems*, 15(6):2671–2682. doi:10.1109/TITS.2014.2325395.

Li, Z., Kolmanovsky, I. V., Atkins, E. M., Lu, J., Filev, D. P., & Bai, Y. (2017). Road disturbance estimation and cloud-aided comfort-based route planning. *IEEE Transactions on Cybernetics*, 47(11):3879–3891. doi:10.1109/TCYB.2016.2587673.

Liebig, T., Piatkowski, N., Bockermann, C., & Morik, K. (2017). Dynamic route planning with real-time traffic predictions. *Information Systems*, 64:258 – 265. doi:10.1016/j.is.2016.01.007.

Liu, J. (2017). Lstm network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, 11:68–75(7). doi:10.1049/iet-its.2016.0208.

McArdle, G., Lawlor, A., Furey, E., & Pozdnoukhov, A. (2012). City-scale traffic simulation from digital footprints. In: *ACM SIGKDD International Workshop on Urban Computing*, pages 47–54. doi:10.1145/2346496.2346505.

Pan, J., Popa, I. S., Zeitouni, K., & Borcea, C. (2013). Proactive vehicular traffic rerouting for lower travel time. *IEEE Transactions on Vehicular Technology*, 62(8):3551–3568. doi:10.1109/TVT.2013.2260422.

Sanaullah, I., Quddus, M., & Enoch, M. (2016). Developing travel time estimation methods using sparse gps data. *Journal of Intelligent Transportation Systems*, 20(6):532–544. doi:10.1080/15472450.2016.1154764.

Schrank, D., Eisele, B., Lomax, T., & Bak, J. (2015). Scorecard, urban mobility. Technical report, The Texas A&M Transportation Institute and Inrix.

Smith, D., Djahel, S., & Murphy, J. (2014). A SUMO based evaluation of road incidents' impact on traffic congestion level in smart cities. *Proceedings - Conference on Local Computer Networks, LCN*, 2014-Novem(November):702–710. doi:10.1109/LCNW.2014.6927724.

Uppoor, S., Trullols-Cruces, O., Fiore, M., & Barcelo-Ordinas, J. M. (2014). Generation and analysis of a large-scale urban vehicular mobility dataset. *IEEE Transactions on Mobile Computing*, 13(5):1061–1075. doi:10.1109/TMC.2013.27.

Wang, C., Pan, J., Xu, H., Jia, J., & Meng, Z. (2015). An improved a* algorithm for traffic navigation in real-time environment. In: *International Conference on Robot, Vision and Signal Processing (RVSP)*, pages 47–50. doi:10.1109/RVSP.2015.20.

Wang, H., Tang, X., Kuo, Y.-H., Kifer, D., & Li, Z. (2019). A simple baseline for travel time estimation using large-scale trip data. *ACM Trans. Intell. Syst. Technol.*, 10(2):19:1–19:22. doi:10.1145/3293317.

Wang, S., Djahel, S., Mcmanis, J., Mckenna, C., & Murphy, L. (2013). Comprehensive Performance Analysis and Comparison of Vehicles Routing Algorithms in Smart Cities. In: *Global Information Infrastructure Symposium (GIIS)*, pages 1–8. doi:10.1109/GIIS.2013.6684365.

Wu, Y.-J., Chen, F., Lu, C.-T., & Yang, S. (2016). Urban traffic flow prediction using a spatio-temporal random effects model. *Journal of Intelligent Transportation Systems*, 20(3):282–293. doi:10.1080/15472450.2015.1072050.

Ziliaskopoulos, A., Waller, S., & Barrett, C. (1999). VISTA, Visual Interactive System for Transportation Algorithms. Technical report, UC Berkeley Transportation Library.