W-edge: weighing the edges of the road network

Rade Stanojevic, Sofiane Abbar, Mohamed Mokbel
Qatar Computing Research Institute, HBKU, Doha, Qatar

ABSTRACT

Understanding link travel times (LTT) has received significant attention in transportation and spatial computing literature but they often remain behind closed doors, primarily because the data used for capturing them is considered confidential. Consequently, free and open maps such as OpenStreetMap (OSM) or TIGER, while being remarkably accurate in capturing geometry and topology of the road network are oblivious to actual travel times. Without LTTs computing the optimal routes or estimated time of arrival is challenging and prone to substantial errors. In this work we set to enrich the underlying map information with LTT by using a most basic data about urban trajectories, which also becomes increasingly available for public use: set of origin/destination location/timestamp pairs. Our system, W-edge utilizes such basic trip information to calculate LTT to each individual road segment, effectively assigning a weight to individual edges of the underlying road network. We demonstrate that using appropriately trained edge weights, the errors in estimating travel times are up to 60% lower than the errors observed in OSRM or GraphHopper, two prominent OSM-based, traffic-oblivious, routing engines and are comparable to those of the commercial maps services.

ACM Reference format:
In *Proceedings of ACM Conference, , 2018 (Preprint),* 10 pages.

1 INTRODUCTION

Given a road network represented as a set of nodes and *weighted* edges, a source node, and a destination node, the shortest path operation finds a path from the source node to the destination node that minimizes the sum of edge weights. Finding shortest (or quickest) paths is critical for a number of spatial queries such as navigation, route planning or isochrone calculation. The most common way to define edge weights is by link travel time (LTT): an *average* time a vehicle spends on the road between two neighboring nodes in the graph which defines the road network. We use terms LTT and edge weight interchangeably throughout this paper, but note that the methodology proposed here can be seamlessly applied to extract per-edge cost different to duration, such as CO_2 emissions or fuel consumption [33].

The actual travel time between any two points can be highly variable due to variations in traffic demand, traffic lights, uncertain arrivals at the intersections, traffic light dynamics, individual driving behavior, etc. Hence, obtaining such edge weights or LTTs is highly non-trivial and a substantial amount of literature has been devoted to derive such information. Unlike the case for edge length and maximum speed that are mostly publicly available either through governmental or public websites (e.g., OpenStreetMap (OSM) or TIGER), accurate edge weights inferred either through loop detectors [5], ANPR [17], private GPS traces [14] are considered as proprietary information and not accessible to public. This sets apart commercial routing engines (e.g. Google Maps) from those who use publicly available data, like OSMR [21] and Graphhopper [16], where LTTs are inferred using heuristics and available road metadata[1]. To verify this, we performed comprehensive experiments in traffic-oblivious, OSM-based, routing engines OSMR and Graphhopper and found that they exhibit very large errors, greater than 50% in peak hours, in estimating travel times. This demonstrates the need to have publicly accessible accurate edge weights, which will have great impact in developing freely accessible and *accurate* shortest path operations.

In this paper we describe a methodology, W-edge, for enriching the road network with LTTs which utilizes nothing more than origin/destination location and timestamp information. We focus on data in this form as it is the 'least common denominator' of the trajectory datasets available for public use. While a number of cities have already publicly shared such taxi trajectories, we expect many other urban areas to follow. In particular, Freedom of Information Act (FOIA) [12] is a powerful tool in the hands of public [31] to request and obtain data in this form from publicly regulated transport organizations, such as taxi or public buses.

In contrast to the existing work on LTT inference from GPS probe vehicle data, we here strive to reconstruct the LTTs using *only* two points per trajectory. Such extreme sparsity of the input data allows applicability of our approach to extremely wide range of GPS trajectory datasets, but also requires tackling a number of technical challenges which we detail later in Section 4.

We evaluate our approach, W-edge, on three large metropolitan areas: New York, Porto and Doha. Our results indicate that the estimated time of arrival (ETA) median errors using W-edge are virtually indistinguishable to commercial maps and are 50-65% lower than the errors obtained by OSRM and GraphHopper, two prominent OSM-based traffic-oblivious routing engines. Finally, for the three cities we study we publicly share our traffic-aware edge weights which can be seamlessly integrated in the routing engines like OSRM [21] or PgRouting [26] and used by others in academia and industry.

2 TOY EXAMPLE

Here we will briefly introduce the problem we tackle using a very simple example. Consider a road network depicted in Figure 1. Each edge in the network is bidirectional and has unit length. Consider 5 journeys in the network between following pairs of nodes: $0 \rightarrow 5$, $6 \rightarrow 10$, $7 \rightarrow 9$, $8 \rightarrow 12$, $9 \rightarrow 11$, $11 \rightarrow 13$ and assume that for each of these 5 journeys we know their durations to be 11.1, 8.9, 8.2, 9.2, 7.1, 7.9, respectively. The high-level question W-edge aims to answer is: what are the weights of individual road segments which would explain the observed journey durations.

Preprint, 2018,
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1] E.g. speed limit, number of lanes, road type, etc.

Rade Stanojevic, Sofiane Abbar, Mohamed Mokbel
Qatar Computing Research Institute, HBKU, Doha, Qatar

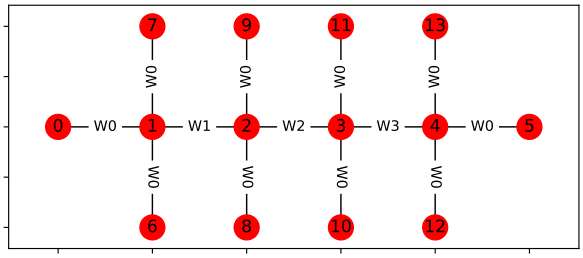

Figure 1: Toy example: road network with 14 nodes and 13 edges with unit length. Three of the 13 edges are heavy: (1,2) (2,3) and (3,4), while the 10 remaining edges are light.

We split the edges in two classes: (1) core or heavy edges:(1,2), (2,3) and (3,4), which are used by many journeys (in this example the threshold is at least 3 journeys), and (2) light edges, are remaining edges which are used by less than 3 of those journeys. For each heavy edge we assign an independent weight variable, while all of the light edges share the same weight W_0. Since the duration of the journey can be approximated by superposition of travel times (or weights) of the edges along the path, we can write 5 equations which relate the edge weights and the travel times of these 5 journeys:

$$\begin{aligned} 2W_0 + W_1 + W_2 + W_3 &= 11.2 \\ 2W_0 + W_1 + W_2 &= 8.9 \\ 2W_0 + W_1 &= 8.2 \\ 2W_0 + W_2 + W_3 &= 9.2 \\ 2W_0 + W_3 &= 7.9 \end{aligned}$$

The above system is over-determined, and hence can only be solved approximately. For example $W_0 = 3$, $W_1 = W_3 = 2$, $W_2 = 1$, closely approximates observed journey durations. However, solving such linear systems in large networks with thousands of variables can lead to unexpected and undesirable results and requires careful consideration. See Section 5.6 for more details.

In the following sections we formalize the problem hinted here, propose a method which can gracefully solve such weight inference in networks with hundreds of thousands of nodes and millions of journeys and demonstrate using extensive data analysis that such traffic-aware weights substantially improve traffic-oblivious routing engines and are comparable in quality to commercial map services.

3 PROBLEM FORMULATION

As we explain in Section 1 our focus is on inferring edge weights using very sparse journey data, containing nothing but origin/destination location and timestamp, together with journey mileage. Such data is becoming increasingly available for public use [9, 10, 24, 31, 35] and is also collected by countless transportation and logistics companies. Using more detailed GPS trajectory data in `W-edge` would be straightforward and could only improve its accuracy.

The road network is represented as a graph, where each node is a unique $(latitude, longitude)$ pair and an edge between two nodes exists if and only if there is a road directly connecting them. Nodes lying in the middle of two-way streets often have in-degree and out-degree equal to 2; nodes lying on the highways have in-degree and out-degree 1; nodes at the intersection of two roads would have

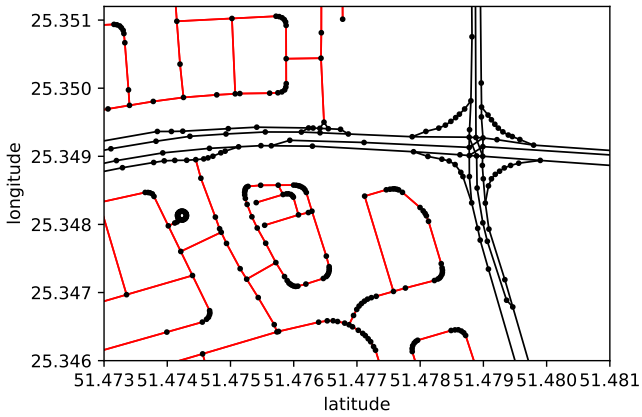

Figure 2: Graph representation of a neighborhood in Doha. Black edges: one way streets; red edges: two-way streets

either in-degree or out-degree greater than 1. See Figure 2 for an illustration of the graph in a neighborhood of Doha. While we work with OSM graph in this paper, weighing any other road network base map using `W-edge` would be straightforward.

A journey τ is represented by the available information: $I_\tau = [(lat, lon)_{orig}, tstamp_{orig}, (lat, lon)_{dest}, tstamp_{dest}, mileage]$. With each journey we associate a path in the road network given by a list of edges in the graph representing the road network $P_\tau = [e_0, \ldots, e_l]$. Effectively we want to do the map-matching using only origin and destination of the trajectory. To that end we query OSRM, an OSM-based routing engine, and denote by P_τ the result of the shortest path query using the journey τ origin and destination [20]. Obviously, P_τ does not necessarily correspond to the actual route traveled and here the mileage information can be used to eliminate those journeys for which the length of P_τ does not match *mileage*. Namely, from the set of available journeys, we consider only those 'reliable' journeys that satisfy:

$$mileage(1-\epsilon) < Length(P_\tau) < mileage(1+\epsilon), \qquad (1)$$

for a small ϵ. In the rest of paper we use $\epsilon = 0.05$, which allows high level of confidence that actual journey traveled over the corresponding path, and keeps a substantial fraction (around 40% of the total) of reliable journeys for training the model. See Section 5 for more details on journey filtering.

We denote by Γ the set of reliable journeys represented by (I_τ, P_τ), for $\tau \in \Gamma$. We aim to find weights of the edges in the underlying road graph that minimize

$$\sum_{\tau \in \Gamma} \Big(\sum_{e \in P_\tau} W_e l_e - \delta_\tau \Big)^2, \qquad (2)$$

where l_e is the length, in meters, of the edge e. Note that for given journey τ, $\sum_{e \in P_\tau} W_e l_e$ represents an estimate of travel time (in seconds), while δ_τ represents the observed travel time. The above sum effectively measures the sum of squares of the difference between the estimated and actual travel times for the set of journeys Γ. In other words, we look for weighing of the edges of the road network such that for every journey which follows a path P_τ its duration is accurately approximated by the sum of the weights of the edges on the path P_τ.

Table 1: Symbol map.

Symbol	meaning
τ	journey
e	edge in road network
P_τ	path of journey τ - list of edges
ϵ	mileage matching threshold
Γ	set of reliable journeys
W_e	weight of edge e (in sec/m)
l_e	length of edge e (in m)
δ_τ	duration of journey (in sec)
H	set of heavy edges
h_l	parameter: number of heavy edges
W_0	weight of **all** light edges
r	number of heavy roads
H_g	heavy roads, $g \in \{1, \ldots, r\}$
W_g	weight of heavy road g
$maxspeed_g$	speed limit on road g (in km/h)
σ	inverse of average speed (in sec/m)
α	Ridge regularization strength

An important remark here is that weights which minimize the errors (2) do not necessarily correspond to effective configuration for computing the shortest paths. Namely, allowing negative edge weights, as in [37], can disturb shortest path computation by not allowing convergence. This issue has been recognized in [22] which constrains the weights to be non-negative. However, even with this constraint, the non-negative least square solver LSEC [22] may (and it does) converge to a solution which associates zero weight to a large fraction of edges which allows shortcuts in the shortest-path computation and create sub-optimal routes and inaccurate ETA estimates; see Section 5.6. Hence, we propose an alternative way for computing edge weights which rely on Ridge regression.

4 W-EDGE: WEIGHING THE EDGES OF ROAD NETWORK

In this section we describe W-edge system which takes as input a graph representation of the road network and a cohort of basic journey information and outputs for each of the graph the expected time, in seconds, a vehicle needs to traverse that edge. We refer to that time as weight or link travel time. Computing the edge weight can be oblivious to time of the travel in which it would capture the 'average' travel time, but edge weight can also depend on the time of the day and/or day of the week in which case we would have weight which is not a scalar but rather a function of time.

As we discuss above, minimizing (2) blindly, without taking into account the physical constraints which limit the actual travel time over any given edge, may lead to weights which are ineffective for the most critical routing primitive: shortest path computation. Additionally, the underlying road network can be very large (e.g. graphs which represent the road network of the three metropolitan areas we study in this paper NYC, Porto and Doha have hundreds of thousands of edges) grouping the edges in a structured way can significantly reduce the dimensionality of the problem and allow scaling the solver to millions of trajectories. Finally, to avoid over-fitting the model we focus only on edges which contain a non-trivial fraction of trajectories.

At this point we assume that every journey is map matched using origin/destination pairs and that we have filtered all of the journeys which do not satisfy the mileage constraint, as described above.

We split W-edge into three sequential phases: (1) heavy edge detection (to avoid over-fitting when low-frequency edges are used in the model), (2) heavy road detection (for dimensionality reduction) and (3) constraint-aware linear regression (for ensuring that edge weights correspond to physical constraints).

4.1 Heavy edges inference

Many journeys share large fraction of their trajectory with other journeys, yet they may have a few edges (typically near the origin or destination) which may be shared with a few or none other trajectories. Regression problems of type (2) which allow each edge in the graph, independently of its 'popularity' to act as a regression feature can easily lead to over-fitting. For that reason we focus on edges which support a large number of trajectories, which we call *heavy edges*. For a training set of trajectories, the set of heavy edges H is derived by sorting the edges according the number of trajectories which pass them and taking the top h_l of them. Here, h_l the number of heavy edges, is a configurable parameter which controls the complexity of the model on one hand, and the execution time as well as the accuracy on the other. Throughout the paper we fix $h_l = 10000$, which captures most major roads in the three cities we study, yet resulting models are not computationally excessive.

Now instead of minimizing the sum (2) we focus on

$$\sum_{\tau\in\Gamma}\Big(\sum_{e\in P_\tau\cap H} W_e l_e + W_0 \sum_{e\in P_\tau\setminus H} l_e - \delta_\tau\Big)^2. \tag{3}$$

where W_0 is a unique weight of the light edges which we assign to all non-heavy edges. Above process reduces the complexity of the model (number of regression features) for 1-2 orders of magnitude in the 3 cities we study by effectively assigning the same weight to all light edges. The key insight here is that majority of shortest paths lie mostly on heavy edges, and therefore the weight of light edges has marginal effect on shortest paths or their length(duration).

4.2 Heavy road detection

Many edges appear in the trajectories simultaneously. By ensuring that they have the same weight we can substantially reduce the dimensionality of the problem. To that end we will group heavy edges together if they belong to the same trajectories. More formally, we split the set H of heavy edges into subsets $H_1, \ldots, H_r$ such that:

$$(\forall i)(\forall \tau \in \Gamma)(\forall e_1, e_2 \in H_i) e_1 \in \tau \iff e_2 \in \tau.$$

In simple words if an edge lies on a trajectory, than all the other edges from its group must lie on that trajectory.

We refer to the (disjoint) sets $H_1, \ldots, H_r$ as heavy roads, as they typically group neighboring edges which form a road or part of a road.

Using the heavy road nomenclature we will rewrite the (3) as:

$$\sum_{\tau\in\Gamma}\Big(\sum_{g:P_\tau\cap H_g\neq\emptyset} W_g L_g + W_0 \sum_{e\in P_\tau\setminus H} l_e - \delta_\tau\Big)^2. \tag{4}$$

where L_g is the length of the heavy road H_g:

$$L_g = \sum_{e \in H_g} l_e$$

Thus the number of unknowns in (4) is $r + 1$, where r is the number of heavy roads: one weight for each of r heavy roads and one weight for all other 'light' edges. In the data from three cities we study, the number of heavy roads r is 3-4 times smaller than the number of heavy edges.

4.3 Enforcing physical constraints via Ridge regression

Authors of [22, 37] solve (2) allowing negative or zero weights which can fundamentally harm the shortest path computation. Here we strive to not only solve the appropriate numerical regression problem but also keep weights physically realistic. To that end, note that weight W_g is inverse of the speed on the relevant road segment and is measured in sec/m. However, on each road segment the speed is limited and that information is often captured by the map itself; e.g. in OSM using a tag *maxspeed* (in km/h). Denoting $maxspeed_g$ the speed limit at road segment g, the physical constraint (accounting for appropriate unit change from km/h to sec/m) for weights becomes:

$$W_g \geq 3.6/maxspeed_g \tag{5}$$

Note that we do not enforce upper limit on weights, as certain roads may become extremely congested (in particular periods of the day/week) and we would like to capture and preserve that information via high edge weight. To ensure that constraint (5) holds for all g we propose to use Ridge regression regularization. Namely we add regularization term to (4) that penalizes weights which largely deviate from the average speed:

$$\sum_{\tau \in \Gamma} \Big(\sum_{g: P_\tau \cap H_g \neq \emptyset} W_g L_g + W_0 \sum_{e \in P_\tau \setminus H} l_e - \delta_\tau \Big)^2 + \alpha \sum_g (W_g - \sigma)^2. \tag{6}$$

Here σ is the inverse of the average speed observed by all the journeys in our data. Parameter α, represent the regularization strength. A small α allows large variability between W_g's, a lot of violations of physical constraint (5) and can potentially lead to over-fitting. A very large α may put too much emphasis on the regularization term neglecting errors we strive to minimize. We choose the hyper-parameter α using a grid search over a small validation set we withdraw from the training cohort; see below.

We use scikit-learn Python library to find W's which minimize (6) and can scale to millions of trajectories, thanks to dimensionality reduction described above and sparse matrix representation of the feature matrix.

With an appropriate regularization strength α (see below for details on how we choose α), Ridge regression regularization indeed forces most of W_g's to be close to σ and we empirically observe that a large majority of segment weights satisfy the constraint (5). In Figure 3 we depict the empiric CDF of the ratio between the inferred speed $3.6/W_g$, and the *maxspeed* tag for all the heavy and non-heavy road segments in the three cities we study here; for more details about the data and evaluation see Section 5. As we can see from the figure, the weights derived by minimizing (6) satisfy the speed limits in over 99% of cases. For a small minority of edges which violate the speed limit constraint we hard code the weights to be exactly equal to $3.6/maxspeed_g$. Thus weight $\bar{W}_g$ at road segment g is

$$\bar{W}_g = max(3.6/maxspeed_g, W_g),$$

where W_g are obtained by minimizing (6). This last step, has a minor effect on the overall ETA estimation, measured through cost (4) since it applies to a minor fraction of the road network, but it eliminates artificial shortcuts which may arise by allowing edges with very high speeds, say over $120kmph$.

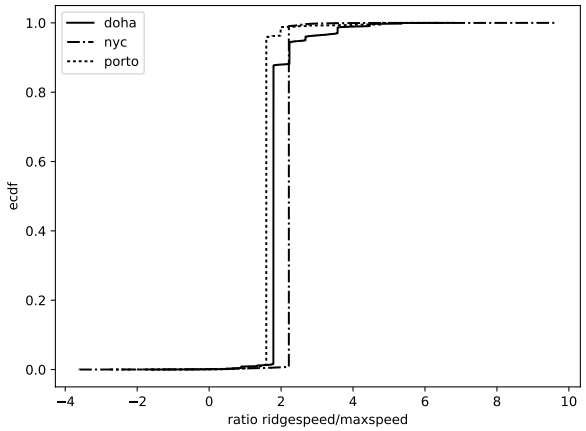

Figure 3: Empiric CDF of $W_g/(3.6/maxspeed_g)$ for three cities we study, where W_g is obtained by minimizing (6). Ratio smaller than 1 implies that W_g allows travel over segment g faster than allowed $maxspeed_g$. In the three cities we study the fraction of such segments is <1%. Note also, that majority of the mass of the CDF is concentrated around 2, which indicates that for most segments average speed is around half of the maximum speed, which was also previously observed by others [29, 33].

4.3.1 Choosing regularization strength α. We do not require α to be manually set, but rather automatically tune it. Our search space is $\alpha \in \{2^k;\ k = 0, 1, \dots\}$. We start by choosing $\alpha = 1$, obtain the weights $\bar{W}_g(\alpha)$ and measure the cost on the validation set (a small set withdrawn from the training cohort) data by substituting those weights in (4) which we denote by $C(\alpha)$. We keep doubling α until $C(\alpha) > C(2\alpha)$, and choose α to be that terminal value, which effectively minimizes the cost $C(\alpha)$ in the above search space.

4.3.2 Number of weights per segment. If our goal is to capture traffic conditions averaged across long time periods, having a single weight per segment would provide reasonably good estimate of the time spent on any given road segment. However it is well known that traffic conditions exhibit periodic behavior on various time-scales: daily, weekly, yearly. For the applications which would require more accurate estimates of the traffic conditions at any given time one may want to train multiple models for capturing the traffic variability in time. For example we can derive one weight per edge for every hour of the day; or one weight per edge for every hour of the week (168 weights) to differentiate between the

Table 2: Basic info about the data

City	# cars	# trips	period	avg duration	# nodes	# edges
NYC	11K	3.4M	1week	12.2 min	299K	596K
Porto	442	1.7M	1year	7.4 min	309K	601K
Doha	4K	1.6M	1month	13.9 min	175K	321K

weekdays and the weekends. The key point here is to train on a subset of trajectories that fall into the appropriate time-slots, and as we shall see in the following section, having time-varying weights can substantially reduce the ETA errors.

5 EVALUATION

5.1 Data

Data our system utilizes to derive weights (or speeds) of individual road segments contains a minimal amount of information about vehicle journeys: origin and destination location, timestamps of the start and the end of the journey as well as the mileage. This succinct journey representation is a fairly common way to publish Taxi/public transportation journey information, as a part of Open Data initiative as well as a result of Freedom of Information Act (FOIA) requests [12, 31]. A number of cities have shared data in this format in the public domain including New York, Chicago, Rome, Porto, Doha, Beijing [9, 10, 24, 31, 35], and we expect many more to do so in the near future. While the format and the actual information shared varies from city to city, the 'least common denominator' of these datasets is the information we use in W-edge.

For the purpose of evaluating W-edge we take advantage of the data generated by taxi fleets in three large cities: New York, Porto and Doha. Basic information about the datasets is shown in Table 2. Each dataset is individually collected by the relevant public transportation authority and contains information about taxi journeys. The data from Porto and Doha we use in this paper is complete data we have for these two cities, yet for New York we take a week-long sample to have a comparable number of trajectories.

The individual edge weights derived by W-edge can be found at: http://ds.qcri.org/people/rstanojevic/wedge/readme.txt. Such traffic-aware edge lengths can be used by researchers and practitioners to improve the quality of their spatial queries.

5.1.1 Preprocessing data: projecting journeys onto road network. As we discuss above, we do not assume that our data contains detailed GPS trace of the trajectory. Hence, understanding the path that vehicle followed from the origin to destination is highly non-trivial. To that end, we use the following heuristic to project the origin/destination/mileage triplet to an actual path. For each origin/destination pair we query OSRM, an OSM-based routing engine, for a shortest path. OSRM uses a number of techniques which prioritize high-speed roads, and for each (origin,destination) pair outputs an actual route it believes is optimal. We consider the journey from our data acceptable if its mileage differs from the OSRM route by at most 5%. All other journeys we filter out as we cannot confidently match their data to a path in the road network. Using above heuristic we filter out about 60% of the journeys, which is a substantial fraction of the dataset. However, the remaining 40% of the journeys still allows to paint a very detailed picture of the road network.

Remark. *One of our datasets, Porto, also contains detailed GPS traces depicting the journey trajectory which allowed us to evaluate the overlap between the matching using heuristic described above with matching using trajectory data and observe a very large (>95%) overlap between the two validating our choice of using mileage-based matching.*

Remark. *Our choice to use OSRM rather than a commercial engine was motivated by two reasons: (1) cost: querying commercial map service for 6.7M trajectories would cost us thousands of USD; and (2) time: with our local OSRM server we could sustain around 100 queries per second or less than a day for 6.7M queries while querying public commercial service for this many trajectories would take one or two orders of magnitude longer.*

5.2 W-edge vs. traffic-oblivious shortest paths

Our first question is: **How does W-edge compare with traffic oblivious OSM-based routing engines in terms of ETA errors?** To that end we choose the two default engines that OSM landing page[2] offers: Open Source Routing Machine (OSRM) [21] and Graphhopper (GH) [16]. Both of those routing engines utilize the underlying OSM data to build the road network graph equipped with weights which are used for computing the optimal routes. The weights OSRM and GH use are derived from the OSM metadata in a way to prioritize routing over motorways and other primary roads. For each (origin,destination) pair both OSRM and GH return a route together with ETA (estimated time of arrival, or route duration) which is derived by solving an appropriate quickest-path query. While both OSRM and GH have a web-based API, OSRM also offers code to set-up a local-server on own premises, which we do, in order to be able to maintain high-throughput. Namely demo OSRM and GH servers with no optimization allow us throughput of around 1 query per second while the local copy of OSRM allows us 100 queries per second.

For each city we first filter out all the trajectories which do not match the length of OSRM path and are left with about 40% of the original trajectories (see previous paragraph). We split these trajectories into training (80%) and test data (20%). And we also withdraw a small fraction (5%) of the training data for validation and tuning the hyper-parameter α.

In the training phase we use the training data (minus validation set) to infer the weights of individual edges as detailed in Section 4. In order to differentiate among different levels of temporal granularity we train three different models:

- W-edge:1. All training trajectories are used to derive a single weight per edge.

- W-edge:24. For each hour of the day ($h = 0 : 23$) we use all the trajectories which originate in the hour h:00:00-h:59:59 to derive weight of the edge at hour h, effectively deriving 24 values for each edge.

[2] https://www.openstreetmap.org/directions

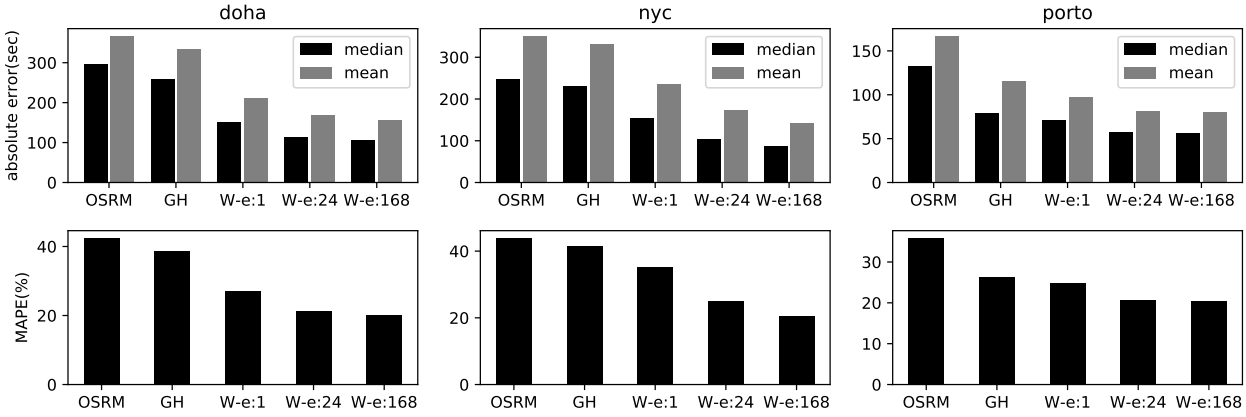

Figure 4: Mean/median absolute error and mean average precision error (MAPE). OSRM and GH have substantial errors. Using `W-edge` reduces all three metrics. Adding temporal granularity in how frequently weights may change additionally reduces errors. E.g. Median absolute errors in Doha using OSRM or GraphHopper are 296 and 258 seconds, respectively. Using `W-edge:168` they go down to 107 seconds, a reduction of 64% and 59%, to OSRM and GH, respectively.

-`W-edge:168`. Similarly we train independent model for each hour of the day for each day of the week, effectively deriving 7*24=168 weight values for each edge.

For each journey τ in the test data we compare the actual travel time with 5 values: OSRM journey duration, Graphhopper journey duration as well as the sum of weight edges over the route P_τ, using `W-edge:1`, `W-edge:24` and `W-edge:168`. We evaluate three different metrics: median absolute error (in seconds), mean absolute error (in seconds) and mean absolute percentage error (MAPE, in %) and report the results in Figure 4.

From the figure we can learn several lessons. First, while querying traffic-oblivious routing engines may offer reasonable routes the actual ETA is very inaccurate. For the cities of Doha and New York the mean absolute error is between 5 and 7 minutes in both OSRM and GH, which is very large, especially given that average trip duration is relatively small 12.2 min in NYC and 13.9 min in Doha (see Table 2). Second, in all three cities and all three metrics GH routing engine outperforms OSRM. Third, adding a single traffic-aware weight per edge can substantially reduce the ETA errors compared to traffic oblivious OSRM and GH. Fourth, `W-edge:24` offers substantial reduction in errors compared to `W-edge:1`, while differentiating between different days of the week in `W-edge:168` does offer improvement to `W-edge:24`, albeit it is quantitatively relatively small.

5.3 When do errors appear?

In this paragraph we ask the question from the title. Throughout the day there is a substantial variability of traffic conditions and we examine how such variability affects the accuracy of traffic-oblivious OSRM or GH as well as traffic-aware `W-edge`. Similarly to the previous section we evaluate the errors on the test journeys using the same three cities and the three different `W-edge` models, as described above. We slice the day in 24 hour-long periods and within each hour we evaluate the errors between the actual and the predicted travel time. In Figure 5 we report the mean absolute error

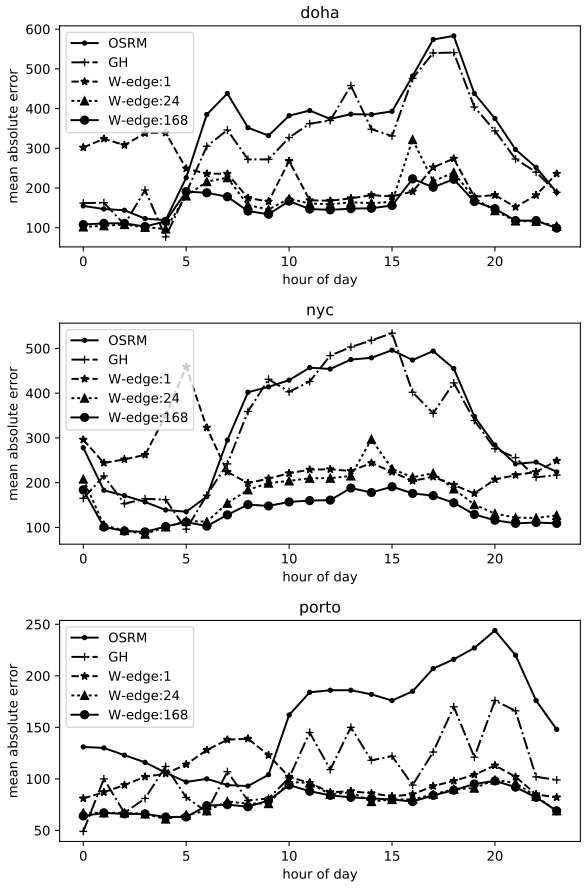

Figure 5: Mean absolute errors throughout the day

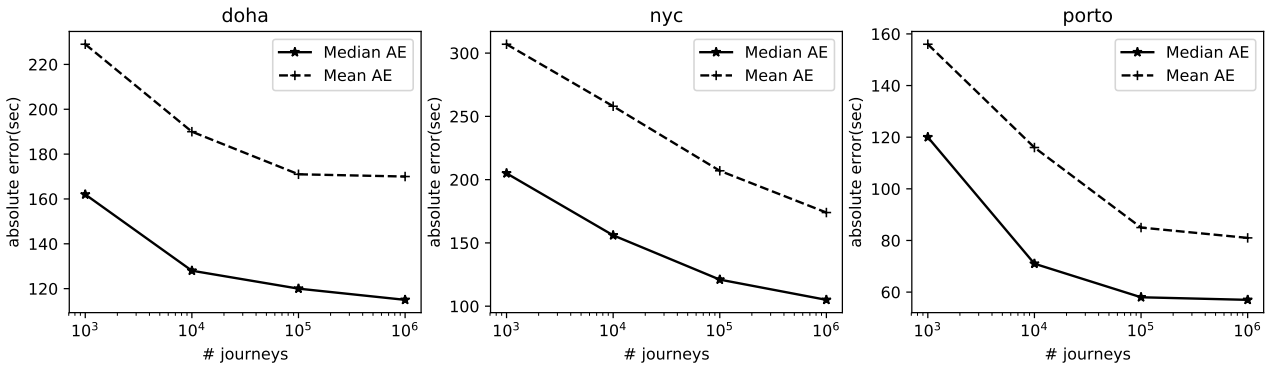

Figure 6: W-edge:24 ETA accuracy gracefully improves with more training journeys up to a point. From 100K trajectories, adding more data results in marginal improvements in accuracy.

across the day; the results for median absolute error and MAPE are qualitatively the same and we omit them here.

From Figure 5 we can observe that in all studied scenarios the errors over night are generally lower than the errors during the day. Mean absolute errors (MAE) of traffic-oblivious OSRM and GH can be as large as 10 minutes in the peak hour in Doha, while traffic-aware W-edge keeps MAE less than 3.5 minutes throughout the day. Note that majority of journeys happen during the day and hence W-edge:1 optimizes for the errors in that period and hence has substantial errors over night. Improvement of W-edge:168 over W-edge:24 is relatively small in Doha and NYC and is virtually indistinguishable in Porto.

5.4 How many journeys?

The fleets which generate data we use in this paper are fairly large, and can produce a large number of journeys. Other fleets may be much smaller, and available data may be of limited size [19]. In this paragraph we examine how the size of training data, measured in terms of available journeys, impacts the ETA errors. To that end we limit the size of the training data to *n_samples* and evaluate the ETA errors on a set of trajectories not used in the training phase. In Figure 6 we report mean and median absolute errors using W-edge:24 granularity for the three cities of interest and varying number of training journeys *n_samples* between 1000 and 1000000; the results for other levels of granularity W-edge:1 and W-edge:168, are qualitatively similar and we omit them here. Intuitively, the larger the training data the smaller the errors. However we observe a law of diminishing returns here, with minor improvements with larger data. Namely, after 100K journeys the accuracy of the W-edge:24 model improves only marginally with more data. Note that training the W-edge:24 model even on a small dataset with several thousand of journeys[3] would outperform traffic oblivious OSRM and GH. However to get most of our models one needs several hundred thousands of journeys.

5.5 Comparison with Google maps

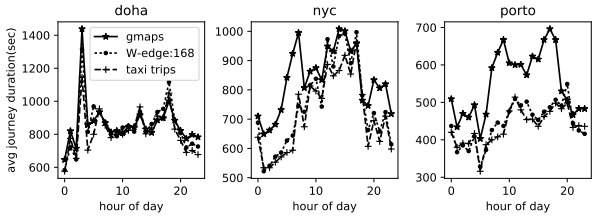

Figure 7: Google maps appears to overestimates the ETA in NYC and Porto, while it is non-biased in Doha. Data from Doha is most recent Feb 2018, while data from NYC and Porto is several years old, collected in 2013 and 2014, respectively.

Table 3: Mean and median absolute errors for Google maps and W-edge:168

City	G.maps median/mean AE	W-e:168 median/mean AE
Doha	112s/149s	106s/159s
NYC	165s/214s	87s/144s
Porto	121s/147s	58s/84s

Comparison with a de facto leader in mapping and navigation services, Google maps (G.maps), is difficult for many reasons. Primarily, G.maps possesses a vast amount of data, in many ways richer than the data we use in our paper. Additionally, G.maps offers a range of routing queries which allow customization and choosing an appropriate comparison is not straightforward. And finally, the data for two of the cities we study (NYC and Porto), is several years old and it appears that traffic in those cities experienced significant shifts which makes comparison with G.maps difficult[4].

We make our best effort to present a fair comparison between W-edge and G.maps in spite of the mentioned challenges. To do so we choose W-edge:168, trained on 80% of the journeys. On the other hand, for a sample[5] journey in the test data with given origin,

[3] A small fleet with, say, a dozen of professional vehicles, can generate 5000 journeys in a matter of weeks.

[4] G.maps allows querying for optimal routes in the future, but does not have the functionality to rewind queries in the past.

[5] We choose a random sample of the test journey to limit the cost of querying the commercial service.

destination and origin-timestamp, we submit a query to G.maps for the optimal route, using the *best_guess* traffic model, between the origin and the destination with the departure time being the time in the second week of May 2018 which coincides with time of the week of origin-timestamp.

For each journey in the G.maps sample (whose origin, destination and timestamp are drawn from the test journeys cohort) we examine the ETA and compare it to the actual journey time using two metrics used above: mean and median absolute errors. Results are reported in Table 3 for G.maps as well as `W-edge:168`. As we can see from the table, in Doha, the ETA errors we observe in G.maps are very close to the ones we get in `W-edge:168`. In contrast for other two cities, NYC and Porto, G.maps shows substantially larger errors that are difficult to be explained. We speculate that a major cause for this mismatch is the fact that our journey data is several years old and that meanwhile substantial change in traffic conditions impacts the results of G.maps which are trained on the current traffic data. In Figure 7 we depict the average journey duration in each hour of the day along with ETA estimates from G.maps and `W-edge:168`. We observe that the three values overlap in Doha throughout the day, but that G.maps shows significantly larger ETA compared to `W-edge:168` and actual travel times in other two cities. Additionally, there could be other factors which may explain the mismatch such as the seasonal effect: traffic in second week of May may be very different to the traffic in NYC in November (when the data was collected), or traffic throughout the year in Porto. Answering this puzzle would merit an interesting study of independent interest, but is out of scope of the present paper.

To conclude, while data for NYC and Porto may be too old to draw a fair comparison between G.maps and `W-edge`, we demonstrate that in Doha, for which we have a very recent dataset (Feb 2018), the observed ETA errors between the two are virtually indistinguishable.

5.6 Impact of zero weights

Recall that we infer the speed/weight of individual edges by observing end-to-end travel times of a set of journeys and minimizing the error (6). Two pieces of work most similar to our work estimate edge weights or link travel times solving a similar system (4) allowing either negative [37] or zero [22] weights.

In this section we shall see that allowing zero (and positive)-weight edges in (4) results in a solution which has comparable ETA errors to `W-edge` when considered errors along a fixed, predefined route as detailed in Section 4. However, zero-weight edges (equivalent to infinite speed roads) create shortcuts which can lead to highly unpredictable effects on shortest path queries.

To understand the impact of zero weights on shortest path queries we solved (4) with a requirement that weights must be nonnegative, which is in essence an implementation of LSEC [22]. We use Python SciPy package non-negative least squares solver to do so.

We evaluate LSEC and `W-edge` as follow. We train both LSEC and `W-edge` using a sample of 100K journeys from Doha dataset. Training LSEC using the whole dataset showed to be computationally challenging, hence we do not utilize the whole datset, but a sample of 100K journeys. For both LSEC and `W-edge` we effectively

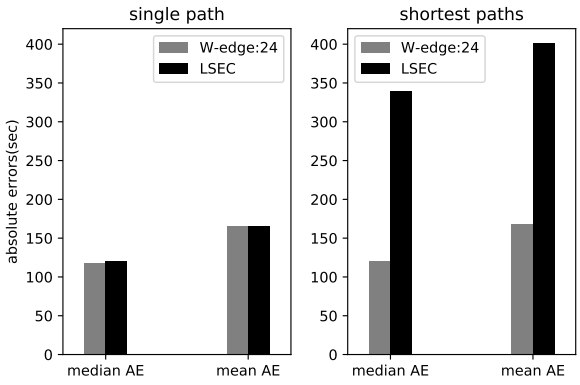

Figure 8: Comparison with LSEC [22]. While ETA errors calculated over fixed path are similar between `W-edge` and LSEC, the ETA errors of the shortest path between origin and destination against actual travel times are $\times 3$ larger in LSEC, due to 'shortcuts' created by zero-weight edges. In contrast `W-edge` shortest paths errors are virtually unchanged.

train 24 models, each corresponding to journeys which start in each of 24 hours of the day. From the journeys not used in training phase we randomly select a thousand journeys which we use for testing. For each journey, defined by (origin/destination/timestamp) triplet we evaluate two errors:

- **single path** ETA error. For given journey $\tau \in \Gamma$ (characterized by origin/destination/timestamp) and weight model (LSEC or `W-edge:24`) we detect the OSRM route P_τ, evaluate the ETA using the weights along that route and compare it to actual taxi journey time.
- **shortest path** ETA error. For given (origin/destination/timestamp) and weight model we evaluate shortest path between the origin and destination and compare ETA defined over such shortest path to actual taxi journey time.

In Figure 8 we report ETA errors for both single path and shortest paths in LSEC and `W-edge:24` models on the test data. When evaluating ETA errors over single paths, LSEC and `W-edge:24` are virtually indistinguishable: numerically solving the least squares problem with non-negativity constraint or using Ridge regularizer, results to similar numerical errors. In contrast, when comparing ETA errors over shortest paths `W-edge:24` marginally changes (for about 2%), while LSEC errors grow by a 250-300%. Such extreme deterioration in the performance of LSEC when using shortest paths ETA is mainly caused by shortcuts created by zero-weight edges.

Given the importance of shortest path queries in map services, we argue that any LTT inference approach must take into account the physical constraints of the network in evaluating the numerically feasible solutions.

5.7 Large OD distance matrices

In transportation planning models a key factor which determines traffic assignment and traffic model calibration is origin-destination (OD) distance matrix [27]. Such OD distance matrix is normally

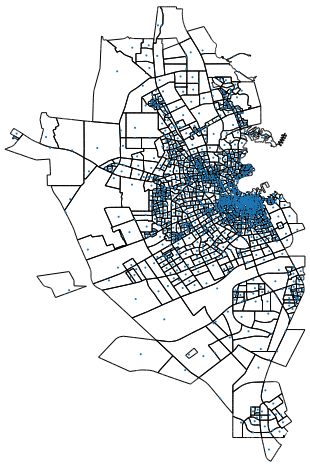

Figure 9: City planning zones in the city of Doha.

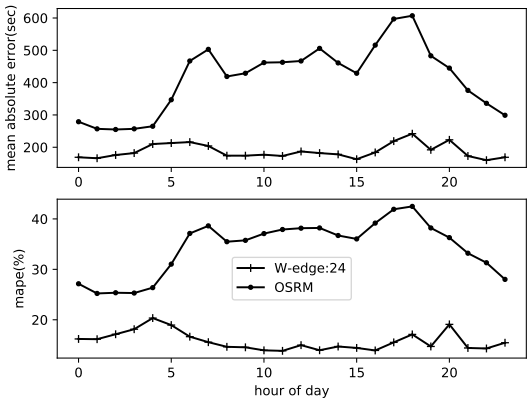

Figure 10: OD distance matrix: errors against Google Maps API. Mean absolute error and MAPE for W-edge:168 and OSRM.

computed using analytical models which map predicted load on a given road segment (measured in number of vehicles per hour) as well as appropriate metadata (such as speed limit and number of lanes) to travel time. In contrast W-edge computes traffic-aware link travel times, and such information can be used to derive data-driven OD distance matrix.

For transportation planning purpose city of Doha is split into 1.3K zones depicted in Figure 9. Each zone covers a region of the city with similar planning parameters (residential, retail, industrial areas, etc.) and inside each zone there is a zone representative point (ZRP) chosen among OSM nodes which lie within the zone. OD distance matrix is effectively distance matrix between ZRPs, where the distance between two ZRPs is denoted as the shortest path length over underlying weighted graph representing the road network. We choose W-edge:168 model, to capture the traffic information in each hour of each day of the week separately, effectively computing 168 OD distance matrices.

Alternatively, one may choose to query a commercial map service to get an estimate of such OD distance matrix. However, the bill for querying such large OD matrices can be fairly large. For example, querying Google Maps for each hour of the week would result in $168 \times 1.3K \times 1.3K \approx 283M$ elements, which would cost $1.4M$\$ - with current Google Maps API pricing of 5\$/1000 elements.

Even though we have a fairly large dataset with over a million journeys, for any given hour of the week over 98% of origin-destination zone pairs have no journey which connects them which effectively means that straightforward computation of OD distances from raw journey data is not feasible.

To compare the quality of our OD matrices we use as a source of ground truth the Google Maps API and query a small sample of ZRP pairs: 1000 ZRP pairs per hour spread throughout the week. For each pair of ZRPs for which we have Google Maps distance we compare the derived value (measured in seconds) and the one obtained by W-edge:168 and report two measures of errors split across 24 hour slots: mean absolute error (in seconds) and mean absolute percentage error (MAPE, measured in percentage points) in Figure 10.

We observe mean absolute errors of about $200sec$ and mean absolute percentage errors of about 15% when using W-edge. In contrast, the traffic oblivious OSRM, exhibits absolute errors in the peak hour as large as $600sec$ and MAPE of over 40%.

6 RELATED WORK

A substantial effort in our community has been devoted to optimize shortest path computation [4, 7, 13]. We argue that the competition among various shortest path algorithms is no longer how computationally efficient they are. Instead, it is more about how accurate edge weights they have. A shortest path algorithm with an *acceptable* processing time and accurate edge weights would be favored over another algorithm that has faster processing time but uses inaccurate edge weights.

Indeed, the wide spread of GPS-enabled devices and subsequently the availability of trajectory data they generate has made it possible for the community to approach the problem of edge weight estimation, also known as link travel time (LTT) problem.

An influential work in this space is the one by Idé and Kato [15] in which they aimed at predicting travel time for any pair of origin destination on a map. However, they focus on inferring overall travel time of a route, as opposed to accurately inferring individual edge weights. Two pieces of work close to W-edge are [29, 36]. Both of those works, rely under the assumption that weights (speeds) of spatially close road segments are similar, and hence add to the regression cost a regularizer which enforces similar weights to spatially close edges. While such approach delivers reasonably accurate results in simulated networks, it is unclear how it would perform in real life settings where spatially close roads do not necessarily have similar speeds.

The problem of static weights has led to a series of papers that allow the expression of time-dependent attributes in transportation systems in general [3], and road networks in particular [23, 33]

For instance, Zheng and Ni proposed a time-dependent trajectory regression on road networks [36].

Several papers use trajectory data to find fastest routes in a road network which often requires the computation of edge travel times as an intermediate step [1, 6, 18, 32, 35]. Yuan et al. tackle the problem of finding the fastest route in a city for any triplet of source, destination, and departure time [35]. Authors use dense trajectory data generated by over 33K taxis to create a dynamic road network in which the time needed to traverse each edge is time-dependent and location-variant, i.e., the temporal dynamics of edges depend on their location. Unlike [28, 36] where edge travel times are considered as constants function of time of the day, the approach in [35] deceives different time intervals for different edges, and within each interval, the travel time is modeled as a distribution rather than a single value. Inference of future travel times on similar dynamic networks is enabled using Markov chains in [34].

Yang et al. propose to use the "incomplete" trip information for complete weight annotation of road networks [33]. The objective here is not to accurately capture trip travel times, but to infer travel costs of all edges of the road network when only few edges are covered by the trajectory data. This is a serious problem that many approaches tackle by simply focusing on "hot" (a.k.a landmarks) segments where substantial data is available [22, 35], and setting the weights of the remaining edges to zero, which is incoherent from a transportation point of view. The solution proposed in [33] consist in modeling the problem as a regression problem with an objection function that incorporates PageRank based spatial regularizers, which is quite in line with earlier works developed in [15, 36]

Many other papers tackled related problems. For instance, [2, 6, 25] are representative of probabilistic (uncertain) edge weights inference. Works in [11, 19, 30] aim at estimating travel time and/or speeds from sparse trajectory data. Instead of inferring edge weights, authors of [8, 32] propose new frameworks to deal directly with paths, avoiding splitting trajectories into small fragments.

7 CONCLUSION

Link travel times are often used for shortest-path computation which is essential in a number of spatial applications such as navigation, route optimization or urban planning. However, most of existing approaches for LTT inference rely on proprietary data, and consequently such LTTs are not publicly available. In this paper we address the problem of link travel time inference from end-point journey data: a common way to publicly share trajectory information. The existing efforts for inference of LTTs from end-point trajectory data, basically search for a numeric solution which explains observed data. However, in our work `W-edge` takes into account physical constraints given by speed limits at each road segment and consequently derives LTTs which are substantially more accurate in capturing shortest paths in road networks. We examined `W-edge` in depth and showed that it substantially improves traffic-oblivious routing engines (like OSRM and Graphhopper), and delivers ETA errors comparable with commercial map services. While in this paper we focus exclusively on journey cost measured in travel time, it would be interesting to extend `W-edge` to capture other relevant cost metrics such as fuel consumption or CO_2 emissions.

REFERENCES

[1] S Aljubayrin and et al. Finding Non-dominated Paths in Uncertain Road Networks. In *Proceedings of ACM SIGSPATIAL 2016.*
[2] M. Asghari and et al. Probabilistic estimation of link travel times in dynamic road networks. In *Proceedings of ACM SIGSPATIAL 2015.*
[3] Petko Bakalov, Erik Hoel, and Wee-Liang Heng. Time dependent transportation network models. In *In Proceedings of IEEE ICDE 2015.*
[4] Lijun Chang, Jeffrey Xu Yu, Lu Qin, Hong Cheng, and Miao Qiao. 2012. The exact distance to destination in undirected world. *The VLDB Journal* (2012).
[5] Benjamin Coifman. 2002. Estimating travel times and vehicle trajectories on freeways using dual loop detectors. *Transportation Research Part A: Policy and Practice* 36, 4 (2002).
[6] U. Demiryurek and et al. Online computation of fastest path in time-dependent spatial networks. In *Proceedings of SSTD 2011.*
[7] J. Dibbelt and et al. Fast exact shortest path and distance queries on road networks with parametrized costs. In *Proceedings ACM SIGSPATIAL 2015.*
[8] J. Dai et al. 2016. Path cost distribution estimation using trajectory data. *Proceedings of the VLDB Endowment* 10, 3 (2016), 85–96.
[9] L. Bracciale et al. 2014. CRAWDAD dataset roma/taxi (v. 2014-07-17). Downloaded from https://crawdad.org/roma/taxi/20140717. (2014).
[10] L. Moreira-Matias et al. 2013. Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems, 14(3)* (2013).
[11] Z. Liu et al. Mining road network correlation for traffic estimation via compressive sensing. In *IEEE Transactions on Intelligent Transportation Systems, 2016.*
[12] H. N. Foerstel. 1999. *Freedom of information and the right to know: The origins and applications of the Freedom of Information Act.* Greenwood Press.
[13] R Geisberger and et al. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *International Workshop on Experimental and Efficient Algorithm, 2008.*
[14] T. Hunter and et al. 2009. Path and travel time inference from GPS probe vehicle data. *NIPS Analyzing Networks and Learning with Graphs* (2009).
[15] Tsuyoshi Idé and Sei Kato. Travel-time prediction using Gaussian process regression: A trajectory-based approach. In *Proceedings of SIAM SDM 2009.*
[16] P. Karich and S Schröder. Graphhopper. In *http://www.graphhopper.com.*
[17] E. Kazagli and H. Koutsopoulos. Arterial travel time estimation from automatic number plate recognition data. In *Proceedings of Annual TRB Meeting 2013.*
[18] L. Li and et al. 2018. Go slow to go fast: minimal on-road time route scheduling with parking facilities using historical trajectory. *The VLDB Journal* (2018).
[19] Y. Li and et al. Urban Travel Time Prediction using a Small Number of GPS Floating Cars. In *Proceedings of ACM SIGSPATIAL 2017.*
[20] Y. et al. Lou. Map-matching for low-sampling-rate GPS trajectories. In *Proceedings ACM SIGSPATIAL 2009.*
[21] D. Luxen and C. Vetter. Real-time routing with OpenStreetMap data. In *Proceedings ACM SIGSPATIAL 2011.*
[22] S. Mridha, N. Ganguly, and S. Bhattacharya. Link Travel Time Prediction from Large Scale Endpoint Data. In *Proceedings ACM SIGSPATIAL 2017.*
[23] T. Nakata and J. Takeuchi. Mining traffic data from probe-car system for travel time prediction. In *Proceedings of ACM SIGKDD 2004.*
[24] City of Chicago. 2018. Taxi Trips. *https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew/data* (2018).
[25] Bei Pan, Ugur Demiryurek, and Cyrus Shahabi. Utilizing real-world transportation data for accurate traffic prediction. In *In Proceedings of IEEE ICDM 2012.*
[26] A Patrushev. Shortest path search for real road networks and dynamic costs with pgrouting. In *Free and Open Source Software for Geospatial Conference 2008.*
[27] R. Pendyala and et al. 2012. Integrated land use-transport model system with dynamic time-dependent activity-travel microsimulation. *Transportation Research Record: Journal of the Transportation Research Board* 2303 (2012), 19–27.
[28] D. Pfoser and et al. Dynamic Travel Time Provision for Road Networks. In *Proceedings of ACM SIGSPATIAL 2008.*
[29] T. Idé M. Sugiyama. Trajectory Regression on Road Networks.. In *AAAI 2011.*
[30] Yilun Wang, Yu Zheng, and Yexiang Xue. Travel time estimation of a path using sparse trajectories. In *Proceedings of ACM SIGKDD 2014.*
[31] Chris Whong. FOILing NYCs taxi trip data. 2014.
[32] B. Yang and et al. 2018. PACE: a PAth-CEntric paradigm for stochastic path finding. *The VLDB Journal* (2018).
[33] Bin Yang, Manohar Kaul, and Christian S Jensen. 2014. Using incomplete information for complete weight annotation of road networks. *IEEE Transactions on Knowledge and Data Engineering* 26, 5 (2014), 1267–1279.
[34] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with Knowledge from the Physical World. In *Proceedings ACM SIGKDD 2011.*
[35] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2013. T-drive: Enhancing driving directions with taxi drivers' intelligence. *IEEE Transactions on Knowledge and Data Engineering* 25, 1 (2013), 220–232.
[36] Jiangchuan Zheng and Lionel M Ni. Time-Dependent Trajectory Regression on Road Networks via Multi-Task Learning.. In *AAAI 2013.*
[37] F. Zheng H. Van Zuylen. Urban link travel time estimation based on sparse probe vehicle data. In *Transportation Research Part C: Emerging Technologies 2013.*