Trajectory-aware Lowest-cost Path Selection: A Summary of Results

Yan Li
lixx4266@umn.edu
Dept. of Computer Science & Eng
University of Minnesota - Twin Cities
Minneapolis, MN

Pratik Kotwal
kotwa007@umn.edu
Dept. of Computer Science & Eng
University of Minnesota - Twin Cities
Minneapolis, MN

Pengyue Wang
wang6609@umn.edu
Dept. of Mechanical Engineering
University of Minnesota - Twin Cities
Minneapolis, MN

Shashi Shekhar
shekhar@umn.edu
Dept. of Computer Science & Eng
University of Minnesota - Twin Cities
Minneapolis, MN

William Northrop
wnorthro@umn.edu
Dept. of Mechanical Engineering
University of Minnesota - Twin Cities
Minneapolis, Minnesota

ABSTRACT

The trajectory-aware lowest-cost path selection problem aims to find the lowest-cost path using trajectory data. Trajectory data is valuable since it carries information about travel cost along paths, and also reflects travelers' routing preference. Path-centric travel cost estimation models using trajectory data grows popular recently, which considers the auto-correlation of the energy consumption on different segments of a path. However, path-centric models are more computationally expensive than edge-centric models. The main challenge of this problem is that the travel cost of every candidate path explored during the process of searching for the lowest-cost path need to be estimated, resulting in high computational cost. The current path selection algorithms that use path-centric cost estimation models still follow the pattern of "path + edge" when exploring candidate paths, which may result in redundant computation. We introduce a trajectory-aware graph model in which each node is a maximal trajectory-aware path. Two nodes in the trajectory-aware graph are linked by an edge if their union forms a trajectory-union path. We then propose a path selection algorithm to find a path in the proposed trajectory-aware graph which corresponds to the lowest-cost path in the input spatial network. We prove theoretically the proposed algorithm is correct and complete. Moreover, we prove theoretically that the proposed path selection algorithm cost much less computational time than the algorithm used in the related work, and validate it through experiments using real-world trajectory data.

CCS CONCEPTS

• **Information systems → Geographic information systems**; **Data mining**;

KEYWORDS

path selection, routing, trajectory, shortest path, path-centric

ACM Reference Format:
Yan Li, Pratik Kotwal, Pengyue Wang, Shashi Shekhar, and William Northrop. 2019. Trajectory-aware Lowest-cost Path Selection: A Summary of Results. In *16th International Symposium on Spatial and Temporal Databases (SSTD '19), August 19–21, 2019, Vienna, Austria.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3340964.3340971

SSTD '19, August 19–21, 2019, Vienna, Austria

ACM ISBN 978-1-4503-6280-1/19/08…$15.00
https://doi.org/10.1145/3340964.3340971

1 INTRODUCTION

The trajectory-aware lowest-cost path selection (TLPS) problem aims to find the path with the lowest travel cost between two locations. Informally, the TLPS problem can be defined as follows: given a spatial network and trajectory data in the network, a cost estimation model using the trajectory data, as well as an origin and a destination, find the path with the lowest estimated cost from the origin to the destination. The TLPS problem is a variant of the shortest path selection (SPS) problem with the main difference being that the travel cost along a path is derived from trajectory data, rather than given as an attribution of the spatial network. Figure 1 illustrates a spatial network composed of eleven nodes ($n1, n2, .., n11$) and twelve edges ($e1, e2, .., e12$), where there are six trajectories ($t1, t2, .., t6$). A candidate result path between $n1$ and $n5$ of a SPS problem is in the form of an ordered sequence of individual edges $[e1, e2, e3, e4]$, while a candidate result path in a TLPS problem is composed of two subpaths defined by $t1$ and $t4$. Our approach takes advantage of the recently introduced path-centric model [20, 27] for travel cost estimation using trajectory data, whose basic spatial unit of cost estimation is a path, but proposes a much more efficient algorithm for the following path selection accordingly.

Telematics devices installed on vehicles are collecting large amounts of trajectory data embedded with rich vehicle information, such as green house gas emissions and fuel consumption. The increasing accessibility of such data is inspiring new approaches to the path selection problem that incorporate trajectory information along with the spatial network. Trajectory data is valuable because it reflects the real-world travel cost and the routing preference of travelers in the past. Trajectory data also contains other information not carried by the spatial network, such as road closure or impassability due to temporary factors [7, 21]. Even though automated

map creation and editing using trajectory data has attracted much attention [4, 16], digital maps still lag reality because the frequency of map updating does not match the changes of cities, especially in rapidly developing cities such as Doha (Qatar) or cities where roads are often occupied for other purposes (e.g., night markets). In this paper, we study path selection using trajectory data directly in this paper.

Despite the importance of path selection using trajectory data, very few studies have been conducted on finding lowest-cost paths. Some studies focus on finding frequently-used paths, which only utilizes the spatial information of trajectories [7, 21]. However, a frequently-used path does not always cost the least. In order to find the path with the least cost, we need to know the estimated cost of traveling along the path first. Trajectory data with rich vehicle information facilitates travel cost estimation. The prediction of expected time cost [19, 26] and energy consumption [6, 17, 28] of travel using trajectory data has been widely studies because of their significant societal importance [18, 25]. However, path selection using these estimation models is rarely discussed. Most path selection algorithms adopt an edge-centric cost estimation model, which treats the cost of a path as the sum of the cost on independent edges. However, it has been shown that when estimating travel cost using trajectory data, these models lose some information, such as the auto-correlation between the cost on segments along a trajectory, because they decompose trajectories into independent segments on edges [27]. Recently, a path-centric view of cost estimation was introduced, which treated a path as a sequence of overlapping subpaths [20, 27]. Each subpath in the path-centric cost estimation models has a certain number of trajectories along it. Since a subpath is the basic unit in cost estimation, it maintains the holism of the trajectories along it. However, the path selection algorithms used in [20, 27] were based on existing path selection algorithms which apply a "path + edge" pattern to explore candidate paths.

The main challenge of our problem is the expensive computation of travel cost estimation, which is needed for every candidate path explored in the process of searching for the lowest-cost path, which results in redundant computation. Take Figure 1 as an example, when exploring candidate paths at $n6$ given the current path as $[e1, e5]$, a path $[e1, e5, e8]$ would be a candidates in a method following the "path + edge" pattern. However, according to the trajectory data, there is no fork at $n9$, and no new trajectory starting from it, which means the estimated cost for $[e5, e8]$ would be the same as the corresponding part of $[e5, e8, e11, e12]$ without losing any information. The travel cost for $[e5, e8]$ is estimated repeatedly when estimating the travel cost of $[e5, e8]$, $[e5, e8, e11]$, and $[e5, e8, e11, e12]$.

In this work, we make the following four contributions. First, we introduce a trajectory-aware graph data model in which each node is a maximal trajectory-aware path, that is, a path along which there are at least a certain number of trajectories. Two nodes in the trajectory-aware graph are linked by an edge if their union forms a path. Second, we propose a trajectory-aware path selection algorithm based on the trajectory-aware graph. Then, we prove the completeness and the correctness of the proposed algorithm. We also show both theoretically and experimentally, that the proposed algorithm has a lower computational time cost than the algorithm in [20].

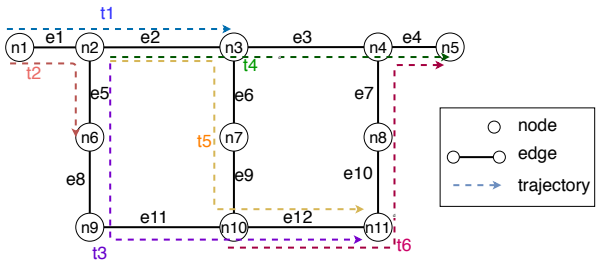


Figure 1: A spatial network with six trajectories.

This paper is organized as follows: In §2, we explain the basic concepts and formally define the trajectory-aware lowest-cost path selection problem. §3 reviews the related literature. §4 presents our data model and algorithm for solving the problem, and an evaluation is given in §5. §6 concludes the paper and presents our future work.

2 BASIC CONCEPTS AND PROBLEM DEFINITION

We introduce the basic concepts in this study, based on which the trajectory-aware lowest-cost path selection problem is formally defined.

2.1 Basic Concepts

A **spatial network** $G = (N, E)$ consists of a **node** set N and an **edge** set E, where each element n in N is a geo-referenced point, while edge set E is a subset of the cross product of N. Each element $e = (n_i, n_j)$ in E is an edge that joins node n_i and node n_j. Fig. 1 shows an example of a spatial network where circles represent nodes (e.g. $n1$, $n2$) and lines represent edges (e.g. $e1$, $e2$). A road system is an example of a spatial network where nodes are road intersections and edges are road segments.

A **path** is a set of edges linking an ordered sequence of nodes. The first and the last nodes are defined as the origin and the destination of the path respectively. A **subpath** of a path is composed of a subset of consecutive edges of the path. In Figure 1, path $[e1, e2]$ is a subpath of path $[e1, e2, e3]$. The union ($\cup$) of two paths $P_\alpha \cup P_\beta$ at a node shared by them is composed of the edges of P_α before the node and those of P_β after the node. For example, in Figure 1, $[e2, e3, e4] \cup [e3, e7]$ at $n4$ is $[e2, e3, e7]$.

A **trajectory** is a log of a vehicle's trip along a path, in the form of a list of pairs, each of which describes an edge and the travel cost on it. Figure 1 shows six trajectories mapped with dashed arrows.

In [20, 27], a **trajectory-aware path** (P_{aware}) was introduced as a path along which there are at least a certain number of trajectories in the same direction. The direction of a P_{aware} is the same as the trajectories along it. A **trajectory-union path** (P_{union}) is the union of several P_{aware}s, and a P_{union} is not a subpath of any P_{aware} that forms it. If we set the minimum number of trajectories along a P_{aware} as 1 in Figure 1, path $[e1, e2]$ is a P_{aware} along which there is a trajectory $t1$. A sample of P_{union} is $[e1, e2, e3, e4]$, which is formed by the union of two P_{aware}s $[e1, e2]$ and $[e2, e3, e4]$ at $n3$.

Table 1: Example trajectory data.

Id	Trajectory Records				
t1	edge	e1	e2		
	cost	2	9		
t2	edge	e1	e5		
	cost	1	1		
t3	edge	e5	e8	e11	e12
	cost	3	2	4	3
t4	edge	e2	e3	e4	
	cost	7	9	2	
t5	edge	e2	e6	e9	e12
	cost	9	2	2	7
t6	edge	e12	e10	e7	e4
	cost	3	2	2	2

2.2 Problem Definition

We formally define the trajectory-aware lowest-cost path selection problem as follows:

Input:

- A spatial network.
- A collection of trajectories in the network.
- A cost estimation model for trajectory-aware paths and trajectory-union paths.
- Two nodes in the spatial network.

Output: A path between the two nodes with the lowest estimated cost.

Objective: Improve the computational efficiency of the path selection algorithm.

Constraints:

- The output path is either a P_{aware} or a P_{union}.
- The cost on any path is positive.

An example of the problem we are solving in this paper is in the following form:

We are given the spatial network shown in Figure 1, a collection of trajectories on it with details shown in Table 1, two nodes $n1$ and $n5$, and a cost estimation model. For simplicity's sake, we adopt a path-centric model simplified from those in [27] and [20]. The model has three features: it requires that the minimum number of trajectories along a P_{aware} be 1; it estimates travel cost on a P_{aware} by the average cost of trajectories along it; and it estimates travel cost on the overlapping edges of two P_{aware}s by the mean of their cost on the edges. For example, the cost on each edge of path $[e1, e2]$ would be $[2, 9]$, while the cost on the edges of path $[e1, e2, e3, e4]$ would be $[2, 8, 9, 2]$. The output of the problem would be the path $[e1, e5, e8, e11, e12, e10, e7, e4]$ with total cost of 18, which is the path from $n1$ to $n5$ with the lowest cost. We will show the details of the path selection procedure in Section 4.

In this paper, we only focus on accelerating the procedure of finding a lowest-cost path with enough trajectory data (i.e., P_{aware} and P_{union}) and using path-centric cost estimation directly [20, 27]. Refinement of cost estimation for paths is outside the scope of this paper. Furthermore, we assume the cost of a path is positive, which is applicable for cases such as time cost and green house gas emissions. In the future, we will generalize our method for negative cost as well.

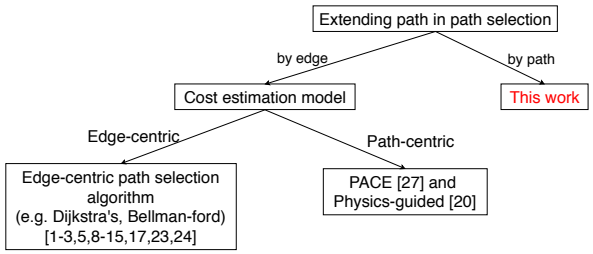


Figure 2: A tree of related works.

3 RELATED WORK AND LIMITATIONS

The trajectory-aware lowest-cost path selection (TLPS) problem is a variant of the shortest path selection (SPS) problem. Based on the basic spatial unit of path cost estimation, the related work on the SPS problem can be categorized into two groups, i.e., edge-centric, and path-centric methods (the left branch in Figure 2).

The basic spatial unit of cost estimation in edge-centric methods is an edge. The travel cost on each edge is assumed to be independent. Most of the edge-centric methods are based on Dijkstra's and Bellman-Ford algorithms which select the shortest path in a static-weighted graph where the cost on each road segment is a constant [10, 15]. Later studies have focused on accelerating computation [1, 3, 8, 24], as well as introducing new constraints (e.g., battery capacity constraint for electric vehicles [2, 12]) and cost metrics (e.g., happiness [23], and bi-objectives metric [11]). In order to represent travel cost more accurately, some work models a road system as a spatio-temporal network, in which the cost on each road segment is a function of time [5, 14]. Other research represents the cost as a stochastic distribution since the factors other than time which affect the travel cost are assumed to be hard to model directly [9, 13]. In order to know the cost of each road segment, which is assumed as a prior in all the aforementioned methods, study has been conducted to utilize trajectory data from vehicles [17]. However, all edge-centric methods suffer from the problem that when decomposing trajectories to estimate the cost of individual road segments, some information, such as the dependence between the costs of adjacent parts along a trajectory, will be lost. For example, when estimating the cost of $e6$ in Figure 1, edge-centric methods treat trajectories along it $(t4, t6)$ the same, without considering the influence of vehicle's movement on the previous road segments, such as the right turn from $e9$ to $e6$.

In recent years, researchers have begun to pay attention to the dependence of segments in a path. In [26], a convolutional neural network is adopted to learn the spatial auto-correlation of nearby trajectory sample points. A path-centric view of path cost estimation introduced by Yang et al. decomposes a path into a sequence of

overlapping subpaths [27]. Inspired by this view, Li et al. propose a physics-guided path-centric method for energy consumption estimation [20]. Since the basic spatial unit in any path-centric method is a subpath, these methods maintain the dependence between the costs of adjacent parts along a trajectory. However, the path selection algorithms used in [27] and [20] still apply a "path + edge" algorithm. In other words, a new candidate path is generated by adding an edge to the end of the current candidate path during the process of searching for the lowest-cost path, which results in redundant computational cost for the cost estimation of candidate paths.

4 APPROACH

We first introduce a trajectory-aware graph data structure to model the spatial road network and the trajectory data on it. Then we present a path selection algorithm using the proposed trajectory-aware graph.

4.1 Trajectory-aware Graph

As introduced in Section 2, a trajectory-aware path (P_{aware}) is defined as a path along which there are more than a certain number of trajectories, which means any subpath of a P_{aware} is a P_{aware}, so we define maximal trajectory-aware path as:

Definition 4.1. A **maximal trajectory-aware path** (P_{max}) is a trajectory-aware path that is not a subpath of any other trajectory-aware path.

Since a P_{max} is a P_{aware}, their union at a node may form a P_{union}. Given a spatial network and trajectory data in it, we can model P_{max}s and the relationships between union-forming P_{max}s using a trajectory-aware graph.

Definition 4.2. A **trajectory-aware graph** is a directed graph whose nodes are P_{max}s. There is an edge between two P_{max}s (P_α and P_β) if $P_\alpha \cup P_\beta$ at a node in the spatial network is a P_{union}.

Since each node in a trajectory-aware graph, P_{max}, represents a path and the travel cost of the path estimated according to trajectory data, a trajectory-aware graph is the combination of a spatial road network and the trajectory data on it.

To clarify the terminology, we name the nodes and edges in a spatial network spatial nodes and spatial edges, while the nodes and edges in a trajectory-aware graph are called trajectory-aware nodes and trajectory-aware edges.

In the condition specified by the example problem in Section 2, the spatial network and trajectories in Figure 1 can be represented by a trajectory-aware graph shown in Figure 3. In this graph, Pi, shown as a square, is a P_{max} along which there is a trajectory ti, and the edge from Pi to Pj indicates the existence of $Pi \cup Pj$ as a P_{union}. For example, two P_{max}s $P1 = [e1, e2]$ and $P4 = [e2, e3, e4]$ can form a union at $n3$, so there is an edge directed from $P1$ to $P4$.

We define a path in a trajectory-aware graph as an ordered sequence of trajectory-aware nodes (P_{max}s) linked by the trajectory-aware edges between them. The union of the P_{max}s of a path in a trajectory-aware graph is a path in the spatial network. If a path in a spatial network is a subpath of the union of the P_{max}s within a path in a trajectory-aware graph, and the origin and the destination are on the first and the last P_{max}s respectively, we say the path

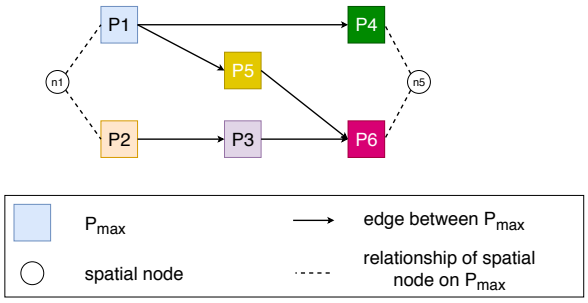


Figure 3: A trajectory-aware graph.

in the spatial network is represented by the path in the trajectory-aware graph. Since a P_{max} may have multiple spatial nodes on it, there may be multiple paths in a spatial network represented by a path in a trajectory-aware graph. We set the cost of a path in a trajectory-aware graph equal to the cost of a P_{union} formed by all but the last P_{max}s in the path except the spatial edges before the origin, so that the cost of a path in a trajectory-aware graph is always less or equal to the path it represents in a spatial network. For example, the cost of a path $[P2, P3, P6]$ with the origin at $n1$ in Figure 3 is equal to the cost of $P_{union} = [e1, e5, e8, e11, e12]$ in Figure 1. After finding a path to the destination in a trajectory-aware graph, the cost of the path in a spatial network from the origin to the destination is estimated by adding the cost on the edges of the last P_{max} to the cost of the path in the trajectory-aware graph.

4.2 Trajectory-aware Path Selection Algorithm

The framework for a stochastic path selection algorithm is shown in Algorithm 1. Given a spatial network, an origin, and a destination, as well as a cost estimation model, the algorithm generates a path satisfying certain criteria. The main steps of the algorithm are as follows. A list of candidate paths CP is initialized in Line 1, typically using the paths consisting of one edge from the origin. Then in each iteration (Lines 2-8), the most promising path in CP is extended, and the result path is added to CP. The iteration ends when the stop criterion is met. The related work implements these steps in different ways. For example, Dijkstra's algorithm's stop criterion is that a path is found between the origin and the destination, while the most promising path in CP is the one with the smallest cost.

In all edge-centric methods, the exploration of candidate paths in Line 4 follows the pattern of "path + edge". In other words, the candidate paths generated from a path are composed of all the edges of the old path and one additional edge linking to the old path's destination. The methods which adopt a path-centric view of cost estimation mainly focus on Line 5, where cost of candidate paths is estimated, but they still use the "path + edge" pattern in Line 4 [20, 27]. In the physics-guided algorithm in [20], the stop criterion is that there is no candidate path that can have cost lower than the already found path. The most promising candidate path is the one with the lowest cost. Table 2 gives the execution trace for the algorithm when solving the example problem in Section 2, which is to find a path between $n1$ and $n5$. Column CP is the set of candidate paths at each step, while column "Cost" is the cost corresponding

Table 2: Execution trace of the algorithm in [20]

step	CP	Cost	p
1	$[e1]$	1.5	$[e1]$
2	$[e1, e5]$,$[e1, e2]$	2,11	$[e1, e5]$
3	$[e1, e5, e8]$,$[e1, e2]$	5,11	$[e1, e5, e8]$
4	$[e1, e5, e8, e11]$, $[e1, e2]$	9,11	$[e1, e5, e8, e11]$
5	$[e1, e5, e8, e11, e12]$, $[e1, e2]$	12,11	$[e1, e2]$
6	$[e1, e5, e8, e11, e12]$, $[e1, e2, e3]$,$[e1, e2, e6]$	12,19,13	$[e1, e5, e8, e11, e12]$
7	$[e1, e5, e8, e11, e12, e10]$, $[e1, e2, e3]$,$[e1, e2, e6]$	14,19,13	$[e1, e2, e6]$
8	$[e1, e5, e8, e11, e12, e10]$,$[e1, e2, e3]$, $[e1, e2, e6, e9]$	14,19,15	$[e1, e5, e8, e11, e12, e10]$
9	$[e1, e5, e8, e11, e12, e10, e7]$,$[e1, e2, e3]$, $[e1, e2, e6, e9]$	16,19,15	$[e1, e2, e6, e9]$
10	$[e1, e5, e8, e11, e12, e10, e7]$,$[e1, e2, e3]$, $[e1, e2, e6, e9, e12]$	16,19,22	$[e1, e5, e8, e11, e12, e10, e7]$
11	$[e1, e5, e8, e11, e12, e10, e7, e4]$,$[e1, e2, e3]$, $[e1, e2, e6, e9, e12]$	18,19,22	

to each candidate path which is in the form of a list of the cost on each edge within a path. The most promising paths in each step are shown in column p.

Algorithm 1 General algorithm framework

Require:
G: A spatial network;
o and d: Two nodes;
model: A cost estimation model.
Ensure: The path between o and d satisfying the criteria.
1: candidate paths $CP \leftarrow$ initialization;
2: **while** stop criteria are not met **do**
3: $p \leftarrow$ the most promising path in CP;
4: **for all** extensions p's of p **do**
5: compute the cost of p';
6: add p' to CP;
7: **end for**
8: **end while**

In our paper, based on the proposed trajectory-aware graph we propose a trajectory-aware path selection algorithm which follows "path + path" pattern in Line 4. The following implementation explains how the framework shown in Algorithm 1 is applied on a trajectory-aware graph. In Line 1, the set of candidate paths CP is initialized with the P_{max}s where the origin is on. The stop criteria in Line 2 is that there is no candidate paths with cost lower than the found path from the origin to the destination. The most promising path in CP is the path with the lowest cost. In Line 4, as candidate

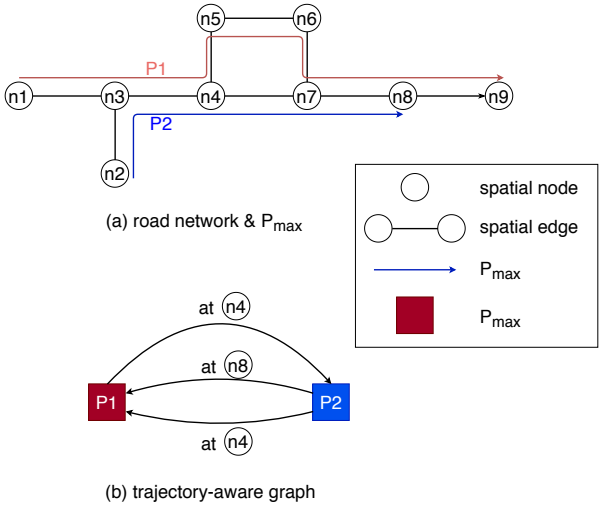


Figure 4: A trajectory-aware graph is a dynamic multigraph.

paths are searched, one P_{max} is added to the currently most promising path if they can form a P_{union}, and P_{max}s which have already been visited can be visited again. Once a path is extended to the destination, we estimate its cost and remove it from the candidate path set. If the estimated cost is lower than the current lowest cost, the result path and the lowest cost are updated. The cost estimation method used in Line 5 is provided as an input.

The main challenge of the algorithm is that a trajectory-aware graph is a dynamic multigraph where there can be multiple trajectory-aware edges between two trajectory-aware nodes, and the existence of a trajectory-aware edge is determined by the previous trajectory-aware nodes and trajectory-aware edges along a path. For example, in Figure 4(a) there are two P_{max}s in a road network. These two P_{max}s form a trajectory-aware graph in 4(b), where $P1$ links to $P2$ at $n4$, while $P2$ links to $P1$ at $n4$ and $n8$. If the trajectory-aware graph is a common multigraph, there may exist paths going back and forth between trajectory-aware node $P1$ and $P2$ multiple times. However, since a trajectory-aware graph represents paths existing in a spatial network, if a path in the spatial network goes from $P1$ to $P2$ at $n4$, it can go back to $P1$ at $n8$ and never go back, while if a path in the spatial network goes from $P2$ to $P1$ at $n8$, it cannot go back. Therefore, a path in a trajectory-aware graph should keep track of not only the trajectory-aware nodes (P_{max}) on the path, but also the trajectory-aware edges (the spatial nodes at which two P_{max}s can form a union).

We can now solve the example problem in Section 2 as follows. Once we transform the input to the trajectory-aware graph shown in Figure 3, the problem of selecting a path between $n1$ and $n5$ in a spatial network with eleven nodes and twelve edges becomes a problem of selecting a path between a set of P_{max}s $\{P1, P2\}$ and another set of P_{max}s $\{P4, P6\}$ in a trajectory-aware graph with 6 nodes and 5 edges. The execution trace is shown in Table 3, and is in the same form as Table 2. At step 1, the candidate paths include the two P_{max}s that $n1$ is on. Since their cost is both 0, we randomly

Table 3: Execution trace of the proposed algorithm

step	*CP*	Cost	*p*
1	[*P*1],[*P*2]	0,0	[*P*1]
2	[*P*1, *P*4],[*P*1, *P*5],[*P*2]	10,11,0	[*P*2]
3	[*P*1, *P*5],[*P*2, *P*3]	11,3	[*P*2, *P*3]
4	[*P*1, *P*5],[*P*2, *P*3, *P*6]	11,12	[*P*1, *P*5]
5	[*P*1, *P*5, *P*6]	20	

choose a promising one. At step 2, we extend [$P1$] by adding one P_{max} to it, which results in two candidate paths. Meanwhile, we find that the path [$P1, P4$] has already reached the destination, so we estimate the cost of traveling along it to the destination, which is 21, and remove it from the candidate path set. At step 4, we find another path to the destination [$P2, P3, P6$], along which the cost to the destination is 18, so we update the result path and the lowest cost. The algorithm terminates at step 5, where there is no candidate path with cost less than the found path.

It is obvious from this example that the trajectory-aware graph simplifies the problem. For example, when generating candidate paths at $n6$, an edge-centric method can only explore the path to $n9$ in one iteration, while the proposed path-centric method explores the path to $n11$ directly since there is no other branch on this path according to the trajectory data. In other words, in the proposed trajectory-aware algorithm, we reduce the number of spatial nodes considered by eliminating the nodes at which no P_{max} form a union, and reduce the number of edges by eliminating the spatial edges within each P_{max}. Since the worst-case time complexity of a stochastic path selection algorithm is $O(|E||V|)$, where $|E|$ is the number of edges and $|V|$ is the number of nodes in a spatial network, the worst-case time complexity of the path selection algorithm in [27] and [20] is linearly correlated to the number of nodes and edges in a spatial network, even though some strategies are introduced to reduce the computational time cost. By reducing the number of spatial nodes and edges considered when selecting paths, the proposed trajectory-aware algorithm will always be faster than the related work, while in the worst case, where at all nodes there exist P_{max}s which can form a union, the trajectory-aware algorithm will be as fast as the related work. Since the acceleration of the proposed algorithm is related to the number of spatial nodes and edges ignored when transferring a spatial network to a trajectory-aware graph, we expect the performance of the proposed algorithm to be affected by the number of trajectories in the input trajectory dataset and the minimum number of trajectories on a P_{aware}, both of which determine whether a spatial node and edge can be ignored. Additionally, the length of the result path (the number of edges within the path) affects the number of iterations needed to reach the termination of the algorithm, we expect that the longer the result path the larger the computational time cost. The experiment results in Section 5 validate our expectation.

4.2.1 Completeness. The completeness of the proposed algorithm means it can explore all possible P_{aware}s and P_{union}s. The physics-guided algorithm in [20] explores all P_{aware}s and P_{union}s in a road network by extending any road segment from each candidate path which can result in a *Paware* or a P_{union}. By contrast, the proposed algorithm explores all subpaths of P_{max}s and unions of P_{max}s. Since a P_{max} is a P_{aware} that is not the subpath of any other P_{aware}s, we have the following lemma.

Lemma 4.1. *Any P_{aware} is a subpath of at least one P_{max}.*

This can be proved by contradiction. Assume that there is a P_{aware} that is not a subpath of any P_{max}. If it is not a subpath of any other P_{aware}s, it is a P_{max} itself. This means it is a subpath of itself, which is a contradiction with the assumption. If it is a subpath of at least one P_{aware} that is not a P_{max}, and because the subpath relation is transitive, there must be an infinite number of P_{aware}s that are not P_{max}s. This contradicts the finite number of trajectories and edges in a road network. Because of Lemma 4.1, for any P_{union} formed by the union of a set of P_{aware}s we can find a set of P_{max}s, each of which has a subpath in the former set of P_{aware}s. Since the origin and the destination of the P_{union} is the origin of the first P_{aware} and the destination of the last P_{aware}, which are in the first and the last P_{aware}s, this gives us the following theorem, which we call the P_{max} union theorem:

Theorem 4.2 (P_{max} union theorem). *Any P_{union} formed by a union of a set of P_{aware}s is a subpath of a P_{union} formed by a union of a set of P_{max}.*

A special case is that of a P_{union} formed by only one P_{aware}. Since there must be a P_{max} having the P_{aware} as a subpath, this case is covered by the P_{max} union theorem as well. Therefore, with the following corollary, we prove the algorithm is complete.

Corollary 4.2.1. *By exploring P_{union}s formed by P_{max}s in the proposed algorithm, we can explore all P_{aware}s and P_{union}s in the road network.*

4.2.2 Correctness. The proposed algorithm is correct if the selected path is the one with the lowest estimated cost from the origin to the destination.

We prove the correctness by contradiction. Assume that there is a path $\hat{P}$ with a lower cost than the path P found by the proposed algorithm. If $\hat{P}$ has been found by the algorithm when the algorithm terminates, the cost of P should be lower than that of $\hat{P}$, since the algorithm always returns the found path with the lowest cost, which is a contradiction with the assumption. If $\hat{P}$ is not found, and since the algorithm explores all P_{aware}s and P_{union}s, and $\hat{P}$ is a P_{aware} or a P_{union}, there must be a subpath $\hat{P}'$ of $\hat{P}$ as a candidate path. Because the cost on paths is assumed to be positive, the cost of $\hat{P}'$ is less than that of both $\hat{P}$ and P, which is contradicted with the stop criterion of the proposed algorithm that there is no candidate paths with cost less than the found path.

5 EVALUATION

We compared the performance of our proposed trajectory-aware algorithm against the baseline physics-guided algorithm in [20] on a real dataset. In order to compare the efficiency of the path selection algorithms, we implemented the cost estimation model in [20] and used it in both our proposed and the baseline algorithms. We also tested our algorithm's sensitivity to different parameter settings. We used computational time cost as the metric to determine efficiency.

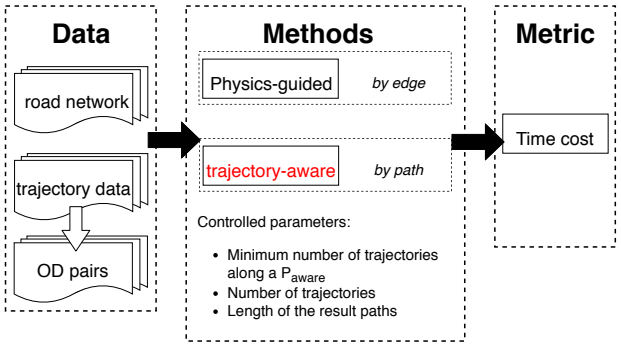


Figure 5: Experiment design.

5.1 Experiment Settings

5.1.1 Dataset. We conducted our experiments on a real dataset containing 920 vehicle trajectories collected from three UPS trucks in Fort Worth, Texas 1/1/2017 - 6/30/2018. Each trajectory represented the trip of one truck in a day. Along with time and location, each record in the trajectories has more than 250 attributes indicating the status of the vehicle's powertrain system (e.g. stop count and energy used). According to the attribute "stop count", we split a trajectory into sub-trajectories between two delivery stops. Each sub-trajectory reflected the routing preference of the UPS truck driver between the stops as well as the mechanical performance of the truck. Then we applied a specially tuned map-matching algorithm derived from [22] to align all sub-trajectories on a digital map covering all road segments traveled by the three trucks which contains 9083 road segments and 5717 road intersections. The digital map data was from OpenStreetMap. After map-matching, each sub-trajectory was represented in the form defined in Section 2. The preprocessed trajectory dataset was composed of 17709 sub-trajectories, whose average length (in terms of the number of road segments) was 54. In all, 5470 road segments were traveled by at least one trajectory. Figure 6 shows the spatial distribution of these road segments in yellow with gradual darkness. The darker the color, the greater the number of trajectories on them. The origin-destination (OD) pairs of each lowest-cost path query in the experiments were the OD pairs of the sub-trajectories, so there were 17709 OD pairs in total. In Figure 6, origins are the red triangles, while destinations are the blue circles. Since the OD pairs in the experiments were the origins and destinations of real-world delivery trips, the experiments show the performance of the proposed algorithm in real cases.

In order to illustrate the sensitivity of the proposed algorithm on the number of trajectories in the input data, we generated two subsets of the original trajectory data by randomly sampling without replacement with equal probability. One subset had 75% (13282) of the trajectories in the original dataset, while the other had 50% (8855).

5.1.2 Experiment environment. All experiments were performed on a single server with a quad-core Intel(R) Xeon(R) CPU E5-2623 v3 (3.00GHz) and 64GB memory. All algorithms were implemented in C#, and version of the Mono runtime was 4.6.2.

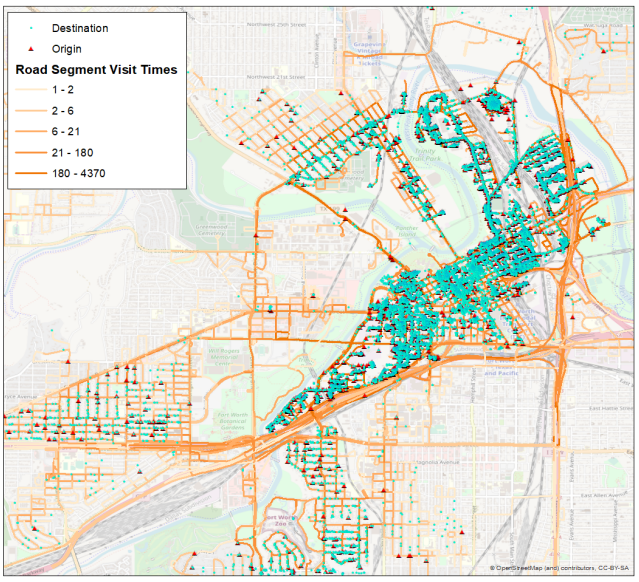


Figure 6: A map of the road segments traveled by trajectories and OD pairs.

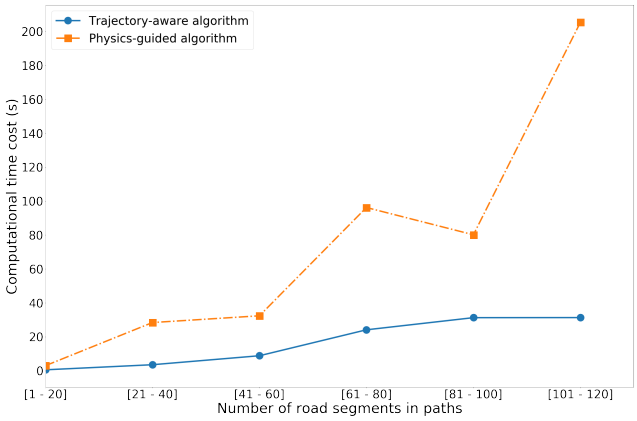
5.1.3 Questions to be answered. The experiments were designed to answer the following questions:

- Is the proposed algorithm computationally more efficient than the baseline algorithm?
- What is the effect of the minimum number of trajectories along a P_{aware} on the proposed algorithm?
- What is the effect of the number of road segments in the lowest-cost path on the proposed algorithm?
- What is the effect of the number of trajectories in the input data on the proposed algorithm?

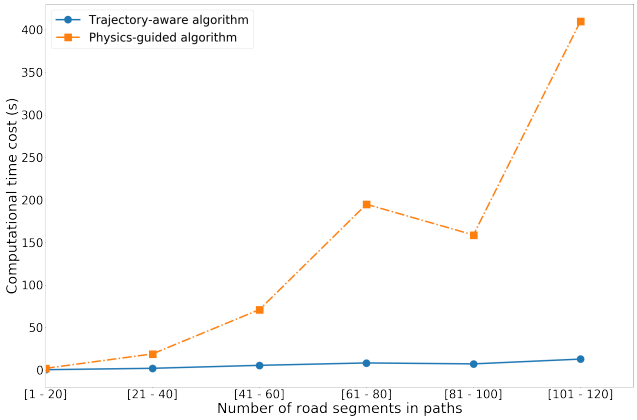
5.2 Experiment Results

As explained in Section 4, both the minimum number of trajectories on a P_{aware} and the number of trajectories in the input data determine the total number of P_{aware}s and P_{union}s considered in path selection, which affects the enumeration space of both the proposed and the baseline algorithm. Additionally, in each lowest-cost path query, the number of road segments in the result path affects the number of iterations in both algorithms. Therefore, we compared the computational performance of both algorithms under conditions with three controlled parameters: β, α, the length of the result path.

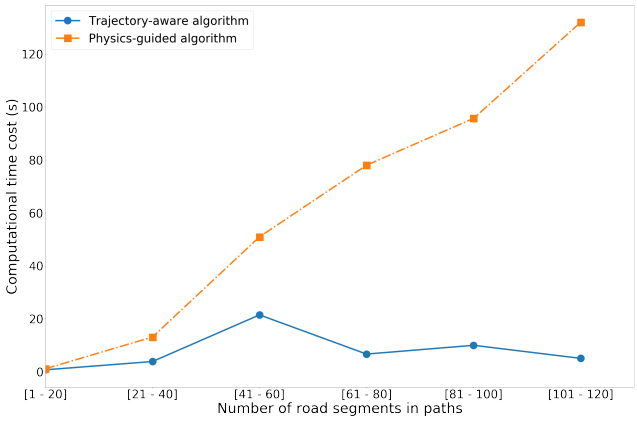
First, we set $\alpha = 17709$ and varied β from 20 to 50. Figure 7 shows the results. As can be seen, the trajectory-aware algorithm always has a lower time cost than the physics-guided algorithm. Furthermore, the gap increases and indeed becomes overwhelming with increasing result path length. We also see that the computational time cost for the trajectory-aware algorithm does not change much as β increase. This means it is less sensitive to differences in



(a) $\beta = 20$.

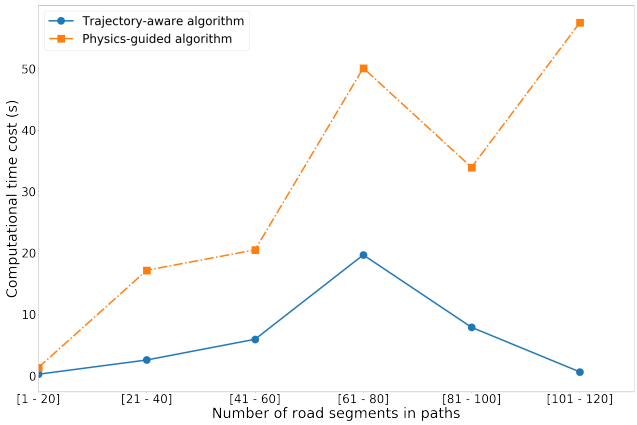


(b) $\beta = 35$.

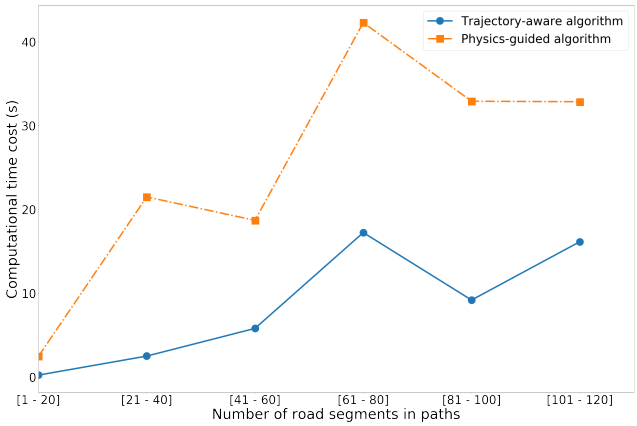


(c) $\beta = 50$.

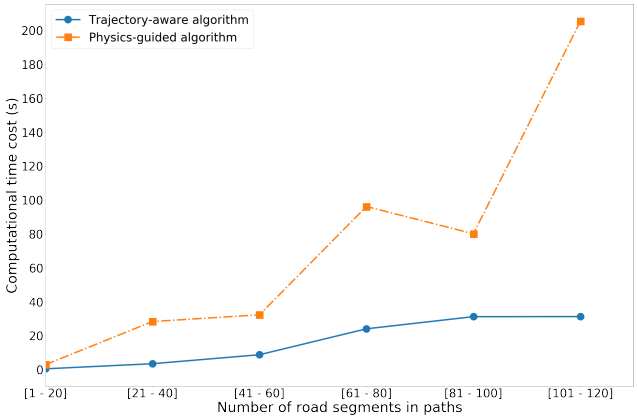
Figure 7: Experiment results with 17709 trajectories and changing minimum number of trajectories on a P_{aware} (β).



(a) $\alpha = 8855$.



(b) $\alpha = 13282$.



(c) $\alpha = 17709$.

Figure 8: Experiment results with fixed minimum number of trajectories on a P_{aware} as 20 and changing number of input trajectories (α).

the minimum number of trajectories on a P_{aware} compared to the baseline.

We fixed $\beta = 50$ and varied α from 8855 to 17709. Results for the two algorithms are shown in Figure 8. Again, the trajectory-aware algorithm outperforms the baseline at all input settings. In addition,

while time cost increases for both algorithms with increasing α, the variance is smaller for the proposed algorithm. This indicates it is more stable than the baseline.

6 CONCLUSION

We study the problem of finding the lowest-cost path through a path-centric algorithm using trajectory data, which follows the pattern of "path + path" when exploring candidate paths instead of "path + edge" in the edge-centric path selection algorithms. We proposed a novel data model, trajectory-aware graph, which combines the road network and trajectory data. We then used the graph to develop a trajectory-aware algorithm. Theoretical analysis proved that the algorithm is correct and complete, and that its computational time cost is less than that of the related works. Experiment results with real-world trajectory and road network data show that the proposed trajectory-aware algorithm is more efficient than the baseline physics-guided algorithm [20], and that the proposed algorithm is less sensitive than the baseline to various input settings.

In the future, we plan to investigate the adjustments of the proposed algorithm needed to handle possibly negative travel cost. Precomputation of the trajectory-aware graph will be studied to further accelerate the computation.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grants No. 1541876, 1029711 , IIS-1320580 , IIS-0940818, and IIS-1218168 , the USDOD under Grants No. HM1582-08-1-0017 and HM0210-13-1-0005, the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy under Award No. DE-AR0000795, the NIH under Grant No. UL1 TR002494, KL2 TR002492, and TL1 TR002493, the USDA under Grant No. 2017-51181-27222, and the OVPR Infrastructure Investment Initiative and Minnesota Supercomputing Institute (MSI) at the University of Minnesota. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. The authors would like to thank Kim Koffolt and the University of Minnesota Spatial Computing Research Group for their comments.

REFERENCES

[1] Sabeur Aridhi, Philippe Lacomme, Libo Ren, and Benjamin Vincent. 2015. A MapReduce-based approach for shortest path problem in large-scale networks. *Engineering Applications of Artificial Intelligence* 41 (May 2015), 151–165.

[2] Andreas Artmeier, Julian Haselmayr, Martin Leucker, and Martin Sachenbacher. 2010. The Shortest Path Problem Revisited: Optimal Routing for Electric Vehicles. In *KI 2010: Advances in Artificial Intelligence (Lecture Notes in Computer Science)*. Springer, Berlin, Heidelberg, 309–316.

[3] JÃ¿rgen Bang-Jensen and Gregory Z. Gutin. 2008. *Digraphs: Theory, Algorithms and Applications*. Springer.

[4] Favyen Bastani, Songtao He, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, and Sam Madden. 2018. Machine-assisted Map Editing. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '18)*. ACM, New York, NY, USA, 23–32. https://doi.org/10.1145/3274895.3274927 event-place: Seattle, Washington.

[5] B. Y. Chen, W. H. K. Lam, Q. Li, A. Sumalee, and K. Yan. 2013. Shortest Path Finding Problem in Stochastic Time-Dependent Road Networks With Stochastic First-In-First-Out Property. *IEEE Transactions on Intelligent Transportation Systems* 14, 4 (Dec. 2013), 1907–1917.

[6] Yuche Chen, Lei Zhu, Jeffrey Gonder, Stanley Young, and Kevin Walkowicz. 2017. Data-driven fuel consumption estimation: A multivariate adaptive regression spline approach. *Transportation Research Part C: Emerging Technologies* 83, 0 (Oct. 2017). https://trid.trb.org/view/1482146

[7] J. Dai, B. Yang, C. Guo, and Z. Ding. 2015. Personalized route recommendation using big trajectory data. In *2015 IEEE 31st International Conference on Data Engineering*. 543–554. https://doi.org/10.1109/ICDE.2015.7113313

[8] Daniel Delling, Andrew V. Goldberg, Andreas Nowatzyk, and Renato F. Werneck. 2013. PHAST: Hardware-accelerated shortest path trees. *J. Parallel and Distrib. Comput.* 73, 7 (July 2013), 940–952.

[9] Yong Deng, Yuxin Chen, Yajuan Zhang, and Sankaran Mahadevan. 2012. Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment. *Applied Soft Computing* 12, 3 (March 2012), 1231–1237.

[10] E. W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numer. Math.* 1, 1 (Dec. 1959), 269–271.

[11] Daniel Duque, Leonardo Lozano, and AndrÃĺs L. Medaglia. 2015. An exact method for the biobjective shortest path problem for large-scale road networks. *European Journal of Operational Research* 242, 3 (May 2015), 788–797.

[12] Jochen Eisner, Stefan Funke, and Sabine Storandt. 2011. Optimal Route Planning for Electric Vehicles in Large Networks. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI'11)*. AAAI Press, San Francisco, California, 1108–1113. http://dl.acm.org/citation.cfm?id=2900423.2900599

[13] Yuan Gao. 2011. Shortest path problem with uncertain arc lengths. *Computers & Mathematics with Applications* 62, 6 (Sept. 2011), 2591–2600.

[14] V. M. V. Gunturi, S. Shekhar, and K. Yang. 2015. A Critical-Time-Point Approach to All-Departure-Time Lagrangian Shortest Paths. *IEEE Transactions on Knowledge and Data Engineering* 27, 10 (Oct. 2015), 2591–2603.

[15] P. E. Hart, N. J. Nilsson, and B. Raphael. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4, 2 (July 1968), 100–107.

[16] Songtao He, Favyen Bastani, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, and Sam Madden. 2018. RoadRunner: Improving the Precision of Road Network Inference from GPS Trajectories. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '18)*. ACM, New York, NY, USA, 3–12. https://doi.org/10.1145/3274895.3274974 event-place: Seattle, Washington.

[17] Xianan Huang and Huei Peng. 2018. Eco-Routing based on a Data Driven Fuel Consumption Model. *arXiv:1801.08602 [stat]* (Jan. 2018). arXiv: 1801.08602.

[18] Phil LeBeau. 2019. Traffic jams cost the US an estimated $87 billion in lost productivity. https://www.cnbc.com/2019/02/11/americas-87-billion-traffic-jam-ranks-boston-and-dc-as-worst-in-us.html

[19] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-task Representation Learning for Travel Time Estimation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. ACM, New York, NY, USA, 1695–1704. https://doi.org/10.1145/3219819.3220033 event-place: London, United Kingdom.

[20] Yan Li, Shashi Shekhar, Pengyue Wang, and William Northrop. 2018. Physics-guided Energy-efficient Path Selection: A Summary of Results. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '18)*. ACM, New York, NY, USA, 99–108. https://doi.org/10.1145/3274895.3274933

[21] Wuman Luo, Haoyu Tan, Lei Chen, and Lionel M. Ni. 2013. Finding Time Period-based Most Frequent Path in Big Trajectory Data. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13)*. ACM, New York, NY, USA, 713–724. https://doi.org/10.1145/2463676.2465287

[22] Paul Newson and John Krumm. 2009. Hidden Markov Map Matching Through Noise and Sparseness. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '09)*. ACM, New York, NY, USA, 336–343. https://doi.org/10.1145/1653771.1653818 event-place: Seattle, Washington.

[23] Daniele Quercia, Rossano Schifanella, and Luca Maria Aiello. 2014. The Shortest Path to Happiness: Recommending Beautiful, Quiet, and Happy Routes in the City. In *Proceedings of the 25th ACM Conference on Hypertext and Social Media (HT '14)*. ACM, New York, NY, USA, 116–125.

[24] Christian Sommer. 2014. Shortest-path Queries in Static Networks. *ACM Comput. Surv.* 46, 4 (March 2014), 45:1–45:31.

[25] U.S. Energy Information Administration. 2017. Total U.S. energy expenditures in 2015 were the lowest in more than a decade. https://www.eia.gov/todayinenergy/detail.php?id=32432

[26] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16657

[27] Bin Yang, Jian Dai, Chenjuan Guo, Christian S. Jensen, and Jilin Hu. 2018. PACE: a PAth-CEntric paradigm for stochastic path finding. *The VLDB Journal* 27, 2 (April 2018), 153–178.

[28] L. Zhu, J. Holden, E. Wood, and J. Gonder. 2017. Green routing fuel saving opportunity assessment: A case study using large-scale real-world travel data. In *2017 IEEE Intelligent Vehicles Symposium (IV)*. 1242–1248.