6-2016

Finding the shortest path in stochastic vehicle routing: A cardinality minimization approach

Zhiguang CAO
Singapore Management University, zgcao@smu.edu.sg

Hongliang GUO

Jie ZHANG

Dusit NIYATO

Ulrich Fastenrath FASTENRATH

Citation

CAO, Zhiguang; GUO, Hongliang; ZHANG, Jie; NIYATO, Dusit; and FASTENRATH, Ulrich Fastenrath. Finding the shortest path in stochastic vehicle routing: A cardinality minimization approach. (2016). *IEEE Transactions on Intelligent Transportation Systems*. 17, (6), 1688-1702.
Available at: https://ink.library.smu.edu.sg/sis_research/8194

Finding the Shortest Path in Stochastic Vehicle Routing: A Cardinality Minimization Approach

Article *in* IEEE Transactions on Intelligent Transportation Systems · December 2015

DOI: 10.1109/TITS.2015.2498160

CITATIONS
89

READS
636

5 authors, including:

Zhiguang Cao
Singapore Management University
79 PUBLICATIONS **2,299** CITATIONS

SEE PROFILE

Hongliang Guo
Agency for Science, Technology and Research (A*STAR)
80 PUBLICATIONS **1,308** CITATIONS

SEE PROFILE

Jie Zhang
Nanyang Technological University
215 PUBLICATIONS **6,546** CITATIONS

SEE PROFILE

Finding the Shortest Path in Stochastic Vehicle Routing: A Cardinality Minimization Approach

Zhiguang Cao, Hongliang Guo, Jie Zhang, Dusit Niyato, and Ulrich Fastenrath

***Abstract*—This paper aims at solving the stochastic shortest path problem in vehicle routing, the objective of which is to determine an optimal path that maximizes the probability of arriving at the destination before a given deadline. To solve this problem, we propose a data-driven approach, which directly explores the big data generated in traffic. Specifically, we first reformulate the original shortest path problem as a cardinality minimization problem directly based on samples of travel time on each road link, which can be obtained from the GPS trajectory of vehicles. Then, we apply an ℓ_1-norm minimization technique and its variants to solve the cardinality problem. Finally, we transform this problem into a mixed-integer linear programming problem, which can be solved using standard solvers. The proposed approach has three advantages over traditional methods. First, it can handle various or even unknown travel time probability distributions, while traditional stochastic routing methods can only work on specified probability distributions. Second, it does not rely on the assumption that travel time on different road segments is independent of each other, which is usually the case in traditional stochastic routing methods. Third, unlike other existing methods which require that deadlines must be larger than certain values, the proposed approach supports more flexible deadlines. We further analyze the influence of important parameters to the performances, i.e., accuracy and time complexity. Finally, we implement the proposed approach and evaluate its performance based on a real road network of Munich city. With real traffic data, the results show that it outperforms traditional methods.**

***Index Terms*—Stochastic shortest path, cardinality minimization, ℓ_1-norm minimization, mixed integer linear programming.**

I. Introduction

With recent increases in vehicles and human populations as well as economic activities, roads in large cities, such as Munich and Singapore, are likely to become ever busier. Intelligent Transportation System (ITS) has therefore been developed to facilitate the traffic. Among all functions of ITS, routing service attracts much broader attention, the objective of which is to determine an optimal path regarding certain cost (i.e., travel time). In reality, a traffic condition is often random because of various uncertainties, such as road work, bad weather, traffic accident and unexpected traffic congestion. The uncertain factors can delay the vehicle from arriving at destination with desirable travel time [1]. Consequently, stochastic shortest path problem is posed to find an optimal route considering the uncertainties.

A. Related Work

In stochastic shortest path problem, least expected travel time (LET) is often used as a routing criterion. A path is optimal if it guarantees least expected travel time. To achieve it, many researchers study this optimization problem. Hooks and Mahmassani [2] provide a thorough review and discussion on a time dependent LET path problem without waiting policy. In [3], Dean solves time-dependent LET path problem with waiting policy. Waller and Ziliaskopoulos [4] address the LET path problem considering correlation between random link travel times. The attractive finding from LET is that the problem can be transformed into a deterministic routing problem and solved efficiently. However, a path with minimum expected travel time may still have a high variance (i.e., risk), which is undesirable for some applications.

To address the risk issue in LET criterion, Nikolova *et al.* [5], [6] propose a mean-risk model. In this model, they seek the path which minimizes the weighted combination of expected travel time and travel time standard deviation. The mean-risk model can also be solved efficiently by converting it into a deterministic shortest-path problem with respect to certain weight on each road link, which is a linear combination of corresponding mean and variance. This model alleviates the risk issue, but still has some limitations. In particular, there is no direct physical meaning of the weighted combination of expectation and standard deviation. Therefore, drivers may not be able to understand and gain actual expectation from solution of the optimization.

The stochastic shortest path problem can also be formulated based on probabilistic comparison as described in [7] and [8], where optimal path has highest probability of being faster than all alternatives. The studies conclude that it is more desirable for drivers to achieve probabilistic fastest path instead of LET path. To determine the optimal path, Sigal *et al.* [7] propose to compute for each path the probability of being faster than all

Manuscript received October 26, 2014; revised March 16, 2015 and September 20, 2015; accepted September 21, 2015. Date of publication December 3, 2015; date of current version May 26, 2016. This work was supported by the BMW Group, Germany. The work of J. Zhang was supported in part by the MOE AcRF Tier 1 under Grant M4011261.020. The Associate Editor for this paper was W.-H. Lin. *(Corresponding author: Hongliang Guo.)*

Z. Cao is with the School of Computer Engineering, Nanyang Technological University, Singapore 639798, and also with the Energy Research Institute @ NTU (ERI@N), Interdisciplinary Graduate School, Nanyang Technological University, Singapore 637141 (e-mail: caoz0005@e.ntu.edu.sg).

H. Guo, J. Zhang, and D. Niyato are with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (e-mail: GuoHL@ntu.edu.sg; ZhangJ@ntu.edu.sg; Dniyato@ntu.edu.sg).

U. Fastenrath is with the Traffic Information Management and Routing Optimization, BMW Group, 80788 Munich, Germany (e-mail: Ulrich.Fastenrath@bmw.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2015.2498160

alternatives. On the other hand, Fastenrath and Becker [8] iteratively perform pair comparisons until the best path is obtained. This probabilistic criterion is much more robust and reliable, but its computation complexity may become prohibitively time-consuming as the road network scales up, because those solutions are derived based on enumeration.

To overcome all above disadvantages, a probability tail model, which incorporates both expected travel time and reliability, is proposed as an optimal criterion in [9]. It defines the optimal path as the one that maximizes the probability of arriving at destination before deadline. This criterion is reasonable in that it is consistent with people's travel planning behavior. One common query could be that "I want to reach hospital in 40 minutes. Please find a path with maximum chance." The key objective of routing in such a circumstance is to reduce the risk of arriving late rather than to minimize the expected travel time. Unfortunately, the solution to get such an path has not been laid out in [9]. Consequently, many subsequent studies investigate this problem and various solutions have been proposed in [5], [10]–[12].

Among those solutions to the probability tail model, three seminal works are [5], [10], and [12]. An adaptive method is developed in [10] to achieve the maximal probability of arriving on time, which provides an optimal policy to select next road junction rather than a prior path. In that solution, a further road junction will be determined only when vehicle arrives at the preceding one, which is inconvenient and not applicable, especially for a pre-planning scenario. By contrast, the works in [5] and [12] aim to search a complete path. In [5], Nikolova *et al.* show that for a large range of deadlines, solving the problem requires maximization of a quasi-convex function over the path polytope. Due to a specific form of its quasi-convex objective, the optimal path can be obtained at an extreme point of the dominant, which is the projection of path polytope onto a two-dimensional plane. Lim *et al.* [12] propose to efficiently search the optimal path by examining probe points, which can eliminate futile searching space. This method improves the performance in terms of computation complexity compared with that of [5].

B. Limitations of Existing Work

The methods in [5] and [12] are desirable in that they output complete path instead of road link to next enter. Moreover, the two methods are significantly faster than enumeration method in terms of computation efficiency. However, they both rely on three common assumptions: 1) travel time for each road link should follow a normal distribution; 2) travel time distributions on different road links are independent; 3) deadline should be longer than the least expected travel time.

The first two assumptions are made to simplify computation. However, in reality, travel time on road link does not always follow an independent normal distribution. And there are literatures challenging these assumptions: 1) in [13], travel time is shown to follow a skewed distribution; 2) in [14], travel time is best fitted with a gamma distribution; 3) the works in [15]–[18] derive a log-normal fit to travel times; 4) in [8], a bi-normal distribution is claimed, where the first normal distribution is for congested traffic and the other is for non-congestion; 5) the works in [16] and [17] show the correlation for travel time on adjacent road links. The third assumption is made to guarantee the quasi-convexity of the problem so as to facilitate the computation. However, in real travel planning, people may not know whether their preferred deadlines are large enough, especially when they are traveling on an unfamiliar routes.

C. Our Contributions

To address aforementioned problems and limitations, we propose a data-driven approach. This approach does not need to assume a fixed distribution of travel time on road link. It can also work well when travel times on different road links are correlated with each other. Moreover, it can handle different deadlines. Although the proposed approach only provides an approximated solution, it is shown by simulation to be satisfactorily close to an optimal one.

More specifically, to obtain the optimal path, we formulate the problem of guaranteeing maximum probability of arriving on time as a cardinality minimization problem. The general model-driven optimization methods [19] cannot solve this problem efficiently since this optimization is neither convex nor quasi-convex if we do not make necessary assumptions on the travel time distribution, correlation and deadlines. However, the cardinality minimization, which is data-driven in this context, can avoid these assumptions, because it is directly applied on the sampled travel time data set to approximate the real probability by frequency. To solve this cardinality minimization problem, we relax it as ℓ_1-norm minimization and its variants, which in turn can be efficiently solved as a mixed integer linear programming problem. Furthermore, we analyze important parameters that may affect the performance.

The data-driven approach has been widely used in many applications, e.g., [20] and [21]. Especially in machine learning, a classifier is always built based on a known data set and prediction is conducted upon coming unknown data. The goal of data-driven approach in machine learning is to minimize the chance of mis-classification [20]. In a similar nature, the data-driven approach in this paper is to minimize the chance of being later than deadline on a known data set, which is expressed as cardinality minimization. To our best knowledge, it is the first time that cardinality minimization is used to solve stochastic shortest path problem whose objective is to find the path that maximizes the probability of arriving at destination before deadline. In addition, we point out that our data-driven approach largely relies on the travel time samples of all road links. Generally, we cannot directly obtain travel time samples, but they can be easily achieved by using big data technique to analyze the GPS trajectories of vehicles, which have traversed those road links [22].

The remainder of this paper is organized as follows. In Section II, we formulate the stochastic shortest path problem to find the path that maximizes the probability of arriving on time. Then we transform it into a cardinality minimization problem. In Section III, we use ℓ_1-norm minimization to relax and solve the cardinality problem, whose two variants are also proposed. Then they all are further formulated as a mixed

integer linear programming problem. In Section IV, we perform extensive experiments on a simple network to justify three important advantages of our approach. In Section V, we provide analysis on important parameters that affect the performance. In Section VI, we further test our approach in an actual road network with real traffic data. Section VII states the conclusion and our future work.

II. Problem Formulation and Reformulation for Stochastic Shortest Path

In this section, we first define a road network as a graph, based on which, we formulate the stochastic shortest path problem. We would like to note that, to find the optimal path, our approach mainly adopts travel time samples of road links, e.g., a vehicle took 100 seconds to traverse a road link, and another vehicle took 120 seconds to traverse the same road link. Generally, it is not easy to directly obtain those travel time samples. Nonetheless, they can be achieved by analyzing the GPS trajectory data when vehicles travel on those road links [22].

A. Original Problem Formulation

We model a road network as a graph. Let $G = (V, A_r)$ be a directed graph, where $V = \{1, 2, \ldots, n\}$ represents the set of nodes and $A_r \subseteq \{(v, w) : v, w \in V, v \neq w\}$ represents the set of arcs, which also refer to the road links. More specifically, (v, w) means an arc from node v to node w. Then the stochastic shortest path problem to maximize the probability of arriving at the destination d from the origin o not later than deadline T can be formulated as follows:

$$\begin{aligned} \max_{\vec{\mathbf{x}}} \quad & \text{Prob}(\vec{\mathbf{w}}^\top \vec{\mathbf{x}} \leq T) \\ \text{s.t.} \quad & \forall v \in V : \sum_{w \in V, (v,w) \in A_r} x(v,w) - \sum_{w \in V, (w,v) \in A_r} x(w,v) \\ & = \begin{cases} 1, & \text{if } v = o \\ -1, & \text{if } v = d \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

where $\vec{\mathbf{w}}$ denotes travel time samples for each arc, which can be obtained from GPS trajectories of vehicles [22]; $\vec{\mathbf{x}} \in \{0, 1\}^{|A_r|}$, and its each component refers to an arc, e.g., $(w, v) \in A_r$ is on the concerned path if $x(v, w) = 1$, and not on it if $x(v, w) = 0$. Then this problem can be compactly written as follows [23]:

$$\max_{\vec{\mathbf{x}}} \text{Prob}(\vec{\mathbf{w}}^\top \vec{\mathbf{x}} \leq T) \left| \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}}; \vec{\mathbf{x}} \in \{0, 1\}^{|A_r|} \right. \tag{2}$$

where $\mathbf{M} \in \mathbb{R}^{n \times |A_r|}$ is node-arc incidence matrix; $b \in \mathbb{R}^n$, whose elements are zeros except the sth and tth element, which are 1 and -1 (i.e., for o and d, respectively).

In general, the optimization problem in Eq. (2) is not convex or quasi-convex if we do not make further assumptions on travel time distribution, correlation and deadlines. This means that there is no efficient method to solve this problem optimally. Alternatively, we seek to approximate the probability by using frequency based on the cardinality minimization problem. We first rewrite the "maximizing" problem in Eq. (2) as "minimizing," which is expressed as follows:

$$\min_{\vec{\mathbf{x}}} \text{Prob}(\vec{\mathbf{w}}^\top \vec{\mathbf{x}} > T) \left| \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}}; \vec{\mathbf{x}} \in \{0, 1\}^{|A_r|} \right. . \tag{3}$$

B. Problem Reformulation as Cardinality Minimization

Definition 1: Cardinality is the number of non-zero elements in a vector or matrix [24], e.g., if $\vec{\mathbf{x}} = [x_1, x_2, x_3] = [0, 0, 4]$, then the cardinality of $\vec{\mathbf{x}}$ is 1.

In our routing problem, the objective is to minimize probability of arriving at destination later than deadline (i.e., as in Eq. (3)). Therefore, it is equivalent to minimizing number of times of arriving at destination later than deadline if size of travel time samples for each path is sufficiently large. For instance, if we travel 1000 times on two paths (e.g., path 1 and path 2) from o to d, and the frequency of arriving at d after deadline is 20 on path 1 and 10 on path 2, then path 2 is considered to be optimal. Thus, we formulate original problem as a cardinality minimization problem, which aims to minimize the cardinality of vector i.e., $C(\cdot)$, defined as follows:

$$\begin{aligned} C(\vec{\mathbf{x}}) &= (c_1, c_2, \ldots, c_S) \\ &= \left(\left[\vec{\mathbf{w}}_1^\top \vec{\mathbf{x}} - T\right]^+, \left[\vec{\mathbf{w}}_2^\top \vec{\mathbf{x}} - T\right]^+, \ldots, \left[\vec{\mathbf{w}}_S^\top \vec{\mathbf{x}} - T\right]^+ \right) \end{aligned} \tag{4}$$

where $[\cdot]^+ = \max\{0, \cdot\}$; each component in $\vec{\mathbf{w}}_i$ denotes the ith travel time sample on corresponding arc; S is size of travel time samples on each arc. Note that the cardinality minimization approach does not require any specific travel time distribution, correlation or deadline. Consequently, our objective to minimize probability of arriving at destination later than deadline can be reformulated as follows:

$$\min_{\vec{\mathbf{x}}} \text{Card}\left(C(\vec{\mathbf{x}})\right) \left| \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}}; \vec{\mathbf{x}} \in \{0, 1\}^{|A_r|} \right. \tag{5}$$

where $\vec{\mathbf{x}}$ is decision variable, denoting the optimal path. Thus, original problem to find an optimal path that maximizes probability of arriving at destination not later than deadline becomes a cardinality minimization problem. It is important to highlight that in this problem, we adopt travel time samples on arcs instead of probability distribution. When sampling size S is large enough, the frequency of not being late can be used to accurately approximate the probability of arriving on time. Therefore, solution to this cardinality optimization can be considered as optimal path in Eq. (1).

III. Solution Method

In this section, we present ℓ_1-norm minimization (and its variants) and mixed integer linear programming. The former is used to transform the cardinality minimization problem while the latter is used to solve the transformed problem.

A. ℓ_1-Norm Minimization to Solve Cardinality Minimization

Generally, cardinality minimization is neither classified as convex nor quasi-convex optimization, and a typical approach to solve it is to relax the problem by ℓ_1-norm. The relaxed problem can be solved efficiently, where ℓ_1-norm is usually known as convex envelop of the function $\text{Card}(\vec{\mathbf{x}})$ [25].

ℓ_1-norm of a vector is denoted by $\|\cdot\|_1$, which is absolute sum of all its elements. For instance, ℓ_1-norm of the vector $\vec{\mathbf{x}}$ can be expressed as follows:

$$\|\vec{\mathbf{x}}\|_1 = |x_1| + \cdots + |x_n| \tag{6}$$

where n is size of $\vec{\mathbf{x}}$. Accordingly, minimization of ℓ_1-norm for $\vec{\mathbf{x}}$ can be expressed as follows:

$$\min_{\vec{\mathbf{x}}} \|\vec{\mathbf{x}}\|_1 \quad | \quad \vec{\mathbf{x}} \in \mathcal{F} \tag{7}$$

where $\mathcal{F}$ is a feasible set.

B. Iterated Weighted ℓ_1-Norm Algorithm

Inspired by the typical ℓ_1-norm minimization, another approximation of $\text{Card}(\vec{\mathbf{x}})$ can be expressed as follows [26]:

$$\text{Card}(\vec{\mathbf{x}}) = \lim_{\epsilon \to 0} \sum_{i=1}^{n} \frac{\log\left(1 + \frac{|x_i|}{\epsilon}\right)}{\log\left(1 + \frac{1}{\epsilon}\right)} \tag{8}$$

with $\epsilon \geq 0$ and $\vec{\mathbf{x}} = [x_1 \quad \cdots \quad x_n]^\top$. Without loss of generality, we decompose $\vec{\mathbf{x}}$ and write it as follows:

$$\vec{\mathbf{x}} = \vec{\mathbf{x}}_+ - \vec{\mathbf{x}}_-, \text{ where } \vec{\mathbf{x}}_+, \ \vec{\mathbf{x}}_- \geq 0. \tag{9}$$

Thus we have $\text{Card}(\vec{\mathbf{x}}) = \text{Card}(\vec{\mathbf{x}}_+) + \text{Card}(\vec{\mathbf{x}}_-)$. Then, we reformulate the cardinality minimization as follows:

$$\min_{\vec{\mathbf{x}}} \sum_{i=1}^{n} \log\left(1 + \frac{x_i}{\epsilon}\right) \mid \vec{\mathbf{x}} \in \mathcal{F}, \quad \vec{\mathbf{x}} \geq 0. \tag{10}$$

This approximation is closer to the real cardinality. Although Eq. (10) is still not a convex problem, it can be solved by 'difference of convex' programming [27]. Therefore, we further linearize the objective function at current $x_i^{(k)}$:

$$\sum_{i=1}^{n} \log\left(1 + \frac{x_i}{\epsilon}\right) \approx \sum_{i=1}^{n} \log\left(1 + \frac{x_i^{(k)}}{\epsilon}\right) + \sum_{i=1}^{n} \frac{x_i - x_i^{(k)}}{\epsilon + x_i^{(k)}}. \tag{11}$$

This objective function can be solved in an iterative manner, and can be further expressed as follows [26]:

$$\min_{\vec{\mathbf{x}}} \sum_{i=1}^{n} w_i x_i \mid \vec{\mathbf{x}} \in \mathcal{F}, \quad \vec{\mathbf{x}} \geq 0 \tag{12}$$

where $w_i = 1/(\epsilon + x_i)$, which will be used as weight in next iteration until it converges to a local solution.

C. Reweighted ℓ_1-Norm Algorithm

In Eq. (12), weight is updated in each iteration. Alternatively, to obtain a better weight for ℓ_1-norm, we can use the merit function [28].

Definition 2: For any $\epsilon > 0$, a merit function $F_\epsilon(\vec{\mathbf{x}})$: $R_n \to R$ is separable and coercive of the form $F_\epsilon(\vec{\mathbf{x}}) = \sum_{i=1}^{n} \psi_i(|x_i| + \epsilon)$ for approximating the cardinality, where the functions ψ_i are strictly concave, i.e., strictly increasing and twice differentiable.

Accordingly, the cardinality minimization problem can be more closely approximated as follows:

$$\min_{\vec{\mathbf{x}}} F_\epsilon(\vec{\mathbf{x}}) \mid \vec{\mathbf{x}} \in \mathcal{F}. \tag{13}$$

The merit functions are not unique. A typical merit function to approximate the cardinality is defined as follows [28]:

$$F_\epsilon(\vec{\mathbf{x}}) = \sum_{i=1}^{n} \log\left(|x_i| + \epsilon\right) + \sum_{i=1}^{n} \log\left(|x_i| + \epsilon\right)^p \tag{14}$$

where $0 < p < 1$. Similarly, to solve this merit function, we can still use the linearization technique. Then, we can reach the same formulation as in Eq. (12). The only difference is the way to compute the weight, which is described as follows [28]:

$$w_i = \frac{1 + (|x_i| + \epsilon)^p \, p}{|x_i| + \epsilon}, \quad i = 1, \ldots, n. \tag{15}$$

D. Mixed Integer Linear Programming

The three algorithms in Section III-A–C can be further reformulated as typical mathematical programming. We only take ℓ_1-norm minimization as an example. For iterated weighted ℓ_1-norm algorithm and reweighted ℓ_1-norm algorithm, the only difference is that they are computed in an iterative manner with different weights as stated in Eq. (12) and Eq. (15), respectively.

We aim to solve the optimization problem in Eq. (5), whose cardinality is defined in Eq. (4). Since ℓ_1-norm can be used to minimize the cardinality, it means that we can minimize the sum of $([\vec{\mathbf{w}}_1^\top \vec{\mathbf{x}} - T]^+, [\vec{\mathbf{w}}_2^\top \vec{\mathbf{x}} - T]^+, \ldots, [\vec{\mathbf{w}}_S^\top \vec{\mathbf{x}} - T]^+)$ instead. To make this objective function more compact, we introduce a non-negative intermediate variable ξ_i and some inequality constraints. Incorporating all above considerations, Eq. (5) can be reformulated as follows:

$$\min_{\vec{\mathbf{x}}} \sum_{i=1}^{S} \xi_i \left| \begin{array}{l} \vec{\mathbf{w}}_i^\top \vec{\mathbf{x}} - T \leq \xi_i; \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}}; \\ \xi_i \geq 0; \vec{\mathbf{x}} \in \{0,1\}^{|A_r|} \end{array} \right. \tag{16}$$

where ξ_i is the intermediate variable associated with $[\vec{\mathbf{w}}_i^\top \vec{\mathbf{x}} - T]^+$ in Eq. (4), and $\vec{\mathbf{x}}$ is decision variable which refers to optimal path. By analyzing the optimization in Eq. (16), we find that the ℓ_1-norm minimization can be easily transformed to a mixed integer linear programming (MILP) problem, which is expressed as follows:

$$\min_{\vec{\mathbf{y}}} \vec{\mathbf{c}}^\top \vec{\mathbf{y}} \left| \begin{array}{l} \mathbf{A}\vec{\mathbf{y}} \leq \mathbf{B}; \mathbf{A}_e \vec{\mathbf{y}} = \mathbf{B}_e; \\ \vec{\mathbf{l}} \leq \vec{\mathbf{y}} \leq \vec{\mathbf{u}}; \vec{\mathbf{y}}_{(1:|A_r|)} \in \mathbb{Z}^{|A_r|} \end{array} \right. \tag{17}$$

where

$$\vec{\mathbf{y}} = \begin{bmatrix} x_1 & \cdots & x_{|A_r|} & \xi_1 & \cdots & \xi_S \end{bmatrix}^\top \tag{18}$$

$$\vec{\mathbf{c}} = \begin{bmatrix} 0_1 & \cdots & 0_{|A_r|} & 1_1 & \cdots & 1_S \end{bmatrix}^\top \tag{19}$$

$$A = \begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1|A_r|} & -1 & 0 & \cdots & 0 \\ W_{21} & W_{22} & \cdots & W_{2|A_r|} & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ W_{S1} & W_{S2} & \cdots & W_{S|A_r|} & 0 & 0 & \cdots & -1 \end{bmatrix} \tag{20}$$

$$\mathbf{B} = \begin{bmatrix} T & \cdots & T \end{bmatrix}^\top \tag{21}$$

$$\mathbf{A}_{\mathrm{e}} = \begin{bmatrix} \mathbf{M} & \mathbf{0}_{|V|\times S} \end{bmatrix}^\top \tag{22}$$

$$\mathbf{B}_{\mathrm{e}} = \vec{\mathbf{b}} \tag{23}$$

$$\vec{\mathbf{l}} = \mathbf{0}_{(|A_r|+S)\times 1} \tag{24}$$

$$\vec{\mathbf{u}} = \begin{bmatrix} \mathbf{1}_{1\times|A_r|} & \boldsymbol{\infty}_{1\times S} \end{bmatrix}^\top. \tag{25}$$

Note that W_{ij} in Eq. (20) is the ith travel time sample on arc j and $\mathbf{B}$ in Eq. (21) has the size of $S \times 1$.

In this MILP problem, $\vec{\mathbf{y}}$ is decision variable, and $\vec{\mathbf{x}}$ in Eq. (18) is optimal path. To solve this MILP problem efficiently, we use the *intlinprog* function in Matlab (version: R2014a), which is based on the branch and bound algorithm. The steps are stated as follows [29]:

1) *Initially reduce the problem size:* Linear program pre-processing is performed on dual problem, the purpose of which is to eliminate redundant variables or constraints.
2) *Solve the relaxed linear programming:* Interior point method [30] is employed to determine an optimal solution to the relaxed problem, which guarantees polynomial complexity.
3) *Tighten the LP relaxation of the mixed-integer problem:* Mixed-integer program pre-processing is conducted to analyze linear inequalities and determine whether some bounds can be tightened.
4) *Further tighten the LP relaxation:* Cut generation is implemented to restrict feasible region of the linear programming relaxations, to ensure that solutions are closer to integers.
5) *Compute the integer-feasible solutions:* Heuristics are used to find feasible points for the branch and bound step below so that an upper bound on the objective functions can be determined [31].
6) *Systematically search for the optimal solution:* Branch and bound method [32] is constructed as a sequence of sub-problems that attempt to converge to a solution of the MILP, where sub-problems give a sequence of upper and lower bounds on the solution.

The *intlinprog* function can solve the MILP problem in any of the steps. In that case, *intlinprog* does not execute the latter steps once it finds optimal solution. More details about these steps can be found in [29].

E. Analysis of Accuracy and Computation Complexity

Before analyzing performance of our approach, we summarize its flow as follows: We originally aim to determine the path that maximizes probability of arriving at destination before deadline, but that optimization problem is neither convex nor quasi-convex without strong assumption, i.e., independent normal distribution for the travel time, which is usually contrary to the conclusions by many researchers. To solve this problem without such strong assumption, we use frequency of arriving at destination before deadline to approximate corresponding probability. However, the frequency can be exactly represented by cardinality. Further, the typical way to solve cardinality minimization is ℓ_1-norm minimization or its variants. We point out that, they only provide approximated solutions, because they are trying to approximate cardinality by ℓ_1-norm, and different variants may have different degrees of approximation. Despite these differences, each of the ℓ_1-norm based algorithms can be further exactly reformulated as a MILP problem, which can be solved by standard solvers.

Based on this summary, there are two critical issues that we should discuss with emphasis because they impact the performance of our approach, i.e., accuracy and computation complexity. We use ℓ_1-norm based algorithms to approximately solve cardinality minimization problem, which results in the accuracy issue. Take iterated weighted ℓ_1-norm algorithm as an example. When ϵ is infinitely close to 0, Eq. (8) can provide 100%-accuracy solution to cardinality minimization. However, Eq. (8) cannot be solved directly. Thus, we linearize it in Eq. (11), which can be solved iteratively. Moreover, it may converge to a sub-optimal solution as iteration number becomes large [33], which also depends on the input data. However, higher accuracy usually comes at the cost of more iterations. Thus, in an actual application, we can seek the satisfactory trade-off between them.

The other important issue comes from MILP problem, which mainly affects time complexity. As stated in Section III-D, we employ branch-and-bound framework to solve the MILP problem. In this framework, it iteratively branches at a new point, and the worst complexity to finish the branching is $\Theta(2^{|A_r|})$ [34], where $|A_r|$ is number of arcs. In each branch, there is a bounding problem, which involves a relaxed linear programming, and its averaged computation complexity is $\Theta((|A_r|+S)^3)$ [35], where S is number of inequality constraints in Eq. (17). Therefore, the worst complexity for the whole algorithm will be $\Theta(2^{|A_r|}(|A_r|+S)^3)$. However, the real time complexity also depends on input data because it may affect the number of branching in MILP problem.

Since both accuracy and time complexity depend on input data (e.g., travel time samples on each road link) and those data are always random in traffic, in the following sections, we first test our approach on an artificial road network with artificial data. Then, we analyze various parameters that may influence the accuracy and time complexity. Finally, we further test our approach in the real road network of Munich city with real traffic data and compare it with the state of the art approach.

IV. Simulation Results

In this section, we evaluate performance of the proposed approach through simulation. Particularly, we first divide travel time samples on all road links into training data set and testing data. Then we use our approach to compute an optimal path

TABLE I
CASE 1: ACCURACY FOR INDEPENDENT SINGLE DISTRIBUTION (%)

	α	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
N	0.2	95.8, 95.5, 96.3, 97.2	96.8, 96.6, 96.9, 97.7	98.6, 98.3, 98.7, 99.1
	0.4	97.6, 97.3, 97.9, 98.1	99.5, 99.3, 99.5, 99.8	98.8, 98.6, 98.9, 99.3
	0.6	97.0, 96.9, 97.2, 97.9	98.1, 97.8, 97.9, 98.5	98.9, 98.7, **99.4**, **99.2**
	0.8	96.7, 96.3, 96.9, 98.4	97.7, 97.6, 98.2, 98.9	99.0, 98.5, 98.9, 99.3
	1.0	95.9, 95.6, 97.7, 98.2	96.3, 95.7, 97.9, 98.4	95.9, 95.7, 96.9, 98.4
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
Bi	0.2	92.9, 92.7, 93,8, 96.1	96.0, 95.7, 96.3, 97.5	97.5, 97.1, 97.4, 97.9
	0.4	95.5, 95.1, 95.8, 96.3	98.9, 98.3, 98.6, 99.2	98.9, 98.5, 98.6, 98.9
	0.6	96.0, 95.4, 96.2, 97.3	98.6, 98.1, 98.7, 99.1	98.8, 98.3, 98.8, 99.1
	0.8	93.9, 93.0, 94.5, 96.2	97.6, 97.0, 98.1, 98.6	97.3, 96.8, 97.1, 97.5
	1.0	93.2, 92.7, 93.8, 95.3	94.1, 93.3, 95.1, 95.8	96.2, 95.0, 95.8, 98.3
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
G	0.2	96.0, 95.7, 96.2, 97.1	97.8, 97.4, 97.9, 98.2	99.1, 98.7, 98.9, 99.4
	0.4	97.8, 97.4, 97.6, 98.2	99.3, **99.5**, **99.2**, **99.1**	99.5, **99.4**, **99.2**, 99.7
	0.6	97.8, 97.2, 97.5, 98.2	98.5, 98.2, 98.7, 99.1	99.1, 99.3, 99.5, 99.7
	0.8	95.1, 94.8, 95.3, 96.2	97.9, 97.3, 97.9, 98.2	98.1, 97.9, 98.4, 98.9
	1.0	96.0, 95.8, 96.4 96.9	96.2, 95.9, 96.5, 97.3	96.4, 96.0, 96.7, 97.3
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
L	0.2	95.9, 95.6, 96.3, 98.1	97.8, 97.1, 97.9, 98.4	98.1, 97.5, 97.9, 98.4
	0.4	95.4, 95.0, 95.8, 97.2	98.3, 98.1, 98.3, 98.9	99.3, 99.1, 99.3, 99.6
	0.6	97.2, 96.8, 97.2, 97.8	98.5, 98.3, 98.7, 99.0	98.9, 98.6, 98.8, 99.3
	0.8	94.5, 94.0, 95.5, 96.2	97.2 96.9, 97.1, 97.8	98.6, 98.0, 98.4, 99.0
	1.0	94.2, 93.9, 94.6, 96.7	94.8, 94.1, 94.9, 97.1	95.3, 94.6, 95.4, 96.7
	1.2	100, 100, 100, 100	100, 100 , 100, 100	100, 100, 100, 100

TABLE II
CASE 2: ACCURACY FOR COMBINED INDEPENDENT PROBABILITY DISTRIBUTION (%)

	α	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
N + Bi	0.2	94.1, 93.6, 94.4, 95.2	97.3, 97.0, 97.6, 98.1	96.5, 96.2, 97.1, 97.8
	0.4	95.6, 95.0, 95.4, 95.9	99.1, 98.8, 99.1, 99.4	98.6, 98.2, 98.3, 99.1
	0.6	96.4, 95.9, 96.8, 97.2	98.9, 98.8, 98.9, 99.2	98.9 **98.9**, **99.2**, **98.9**
	0.8	96.1, 95.4, 96.3, 97.0	97.2, 96.4, 96.9, 98.0	97.6, 97.1, 97.4, 98.9
	1.0	96.2, 95.5, 96.2, 96.9	95.2, 94.7, 95.2, 96.7	96.3 95.5, 96.3, 97.1
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
N + G	0.2	93.1, 91.1, 92.6, 94.7	95.1, 94.6, 95.4, 96.2	96.0, 95.5, 96.3, 97.8
	0.4	93.4, 92.8, 93.8, 94.9	98.3, 97.6, 98.1, 98.5	98.2, 98.4, 98.9, 99.3
	0.6	95.1, 94.0, 95.5, 96.3	96.9, 96.5, 97.2, 97.9	97.9, 97.5, 97.8, 98.5
	0.8	91.2, 89.4, 92.4, 94.7	95.2, 94.8, 95.4, 96.1	96.3, 95.8, 96.2, 97.3
	1.0	92.1, 89.5, **92.9**, **92.4**	91.3, 90.4, 91.9, 93.4	91.5, 90.6, 91.3, 92.7
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
N + L	0.2	94.7, 94.2, 95.1, 96.0	97.2, 96.6, 97.1, 98.0	98.0, 97.3, 97.8, 98.1
	0.4	96.1, 95.7, 96.6, 97.2	98.3, 97.8, 98.3, 98.4	98.1, 97.7, 98.2, 99.0
	0.6	96.0, 95.6, 96.3, 97.1	98.0, 97.5, 97.8, 98.9	98.5, 98.0, 98.6, 98.9
	0.8	93.1, 92.5, 93.2, 94.7	96.3, 95.7, 96.0, 96.7	95.5, 95.4, 96.1, 96.8
	1.0	92.0, 91.4, 92.7, 93.3	93.9, 93.2, 94.3, 95.0	91.4, 90.9, 92.1, 94.2
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100

TABLE III
CASE 3: ACCURACY FOR CORRELATED PROBABILITY DISTRIBUTION (%)

	α	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
N	0.2	98.0, 97.6, 98.2, 98.7	97.9, 97.9, 98.3, 98.9	99.3, 98.8, 99.2, 99.5
	0.4	98.6, 98.1, 98.4, 99.2	99.3, 99.0, 99.2, 99.6	99.2, 99.0, 99.2, 99.7
	0.6	98.3, 98.0, 98.2, 99.0	99.5, 99.3, 99.6, 99.8	99.0, 98.8, 99.0, 99.4
	0.8	97.6, 97.1, 97.5, 98.2	98.6, 98.1, 98.4, 99.0	99.3, 99.0, 99.2, 99.5
	1.0	97.2, 96.6, 96.8, 97.4	97.2, 96.9, 97.2, 98.1	96.9, 96.5, 97.1, 98.2
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
Bi	0.2	96.8, 96.2, 96.8, 97.5	97.3, 96.8, 97.3, 98.5	98.1, 97.7, 98.0, 98.6
	0.4	97.6, 97.4, 97.8, 98.7	99.4, 99.2, 99.3, 99.6	99.5, **99.3**, **99.0**, **99.1**
	0.6	96.5, 96.2, 96.9, 97.5	98.4, 98.1, 98.5, 99.1	99.3, 99.5, 99.7, 99.7
	0.8	97.2, 96.9, 97.4, 98.3	98.9, 98.3, 98.9, 99.2	99.4, 99.3, 99.5, 99.5
	1.0	96.2, 95.9, 96.1, 97.4	96.4, 96.1, 96.7, 97.9	98.0, 96.9, 97.2, 98.1
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
G	0.2	98.2, 97.6, 98.2, 98.7	98.5, 98.2, 98.4, 98.9	98.9, 98.5, 98.8, 99.5
	0.4	98.6,**98.4**, **98.1**, **98.3**	99.5, 99.3, 99.4, 99.8	99.7, **99.7**, **99.3**, **99.4**
	0.6	98.7, 98.5, 98.7, 99.2	99.3, 99.2, 99.4, 99.5	99.5, 99.5, **99.7**, **99.6**
	0.8	97.5, 97.2, 97.8, 98.3	98.6, 98.2, 98.7, 99.3	98.9, 98.5, 98.9, 99.3
	1.0	96.5, 95.9, 96.2, 97.3	98.1, 97.4, 98.1, 98.7	97.8, 97.4, 98.2, 99.0
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
L	0.2	97.0, 96.6, 96.7, 97.2	98.4, 98.1, 98.3, 98.8	98.7, 98.1, 98.5, 98.9
	0.4	97.6, 97.4, 97.7, 98.1	98.9, 98.6, 99.1, 99.3	99.3, 99.2, 99.3, 99.7
	0.6	97.9, 97.3, 97.5, 97.9	99.3, 99.1, 99.3, 99.4	98.9, 98.8, 98.9, 99.3
	0.8	96.2, 95.8, 96.4, 97.5	96.5, 95.9, 96.2, 97.3	98.2, 97.8, 98.2, 98.8
	1.0	95.3, 94.6, 95.3, 95.9	96.1, 95.3, 95.7, 96.5	96.3, 95.7, 96.1, 96.8
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100

based on training data, and compare it with exact solution. The exact solution is obtained by enumerating all possible paths and computing corresponding probabilities on testing data. The accuracy is then measured by comparing the exact solution with solutions derived from our approach.

We should remark that, by "training data," we refer to the historic travel time samples on road links, e.g., collected from 5 days backwards till the current moment, and by "testing data," we refer to the future traffic data, e.g., 24 hours from the current moment onwards. For a typical prediction problem in machine learning, it always uses training data to build a classifier and measures the accuracy based on the ground truth label of testing data. Similarly, in our stochastic shortest path problem, we plan to predict an optimal path for the upcoming driving. However, the metric of this optimal solution is the probability rather than a class label. In real applications, it is more reasonable to find the ground truth optimal path out of both the training data and testing data, instead of only testing data, and then compare it with the path obtained out of training data by our approach. Moreover, we should also note that the accuracy obtained based only on testing data is usually not higher than that of on whole data set. This is the case in this paper, i.e., results in Tables I–III. However, in order to show the lower bound accuracy of our approach, we only use testing data to find the ground truth optimal path and compare it with the solution computed by our approach.

Moreover, we evaluate the impacts of probability distributions, correlation and different deadlines, which constitute three major advantages of the proposed approach over existing algorithms. In particular, we classify the experiment into three cases with respect to travel time on arc. In Case 1, we consider independent probability distributions. In Case 2, we consider combination of independent probability distributions. Then, in Case 3, we consider correlated probability distributions. While Case 1 and Case 2 help to show that our approach is able to handle diverse or even unknown distributions, Case 3 can show that our approach is able to address the correlation issue with respect to travel time. In all cases, we perform the tests with various of deadlines to show that our approach works well for different deadlines.

We also would like to highlight that, for the sake of convenience of justifying our approach, all experiments in Sections IV–VI are performed on a general PC with Intel Core i7-3540M processor and 8.00 GB RAM. However, in real use cases, computation of the optimal path would be done by a powerful central routing engine server, which means that the computation speed in real use cases can be faster than that of the experiments in Sections IV–VI. Additionally, all the optimal paths would be computed in an online manner because the traffic data on each arc may change dynamically.

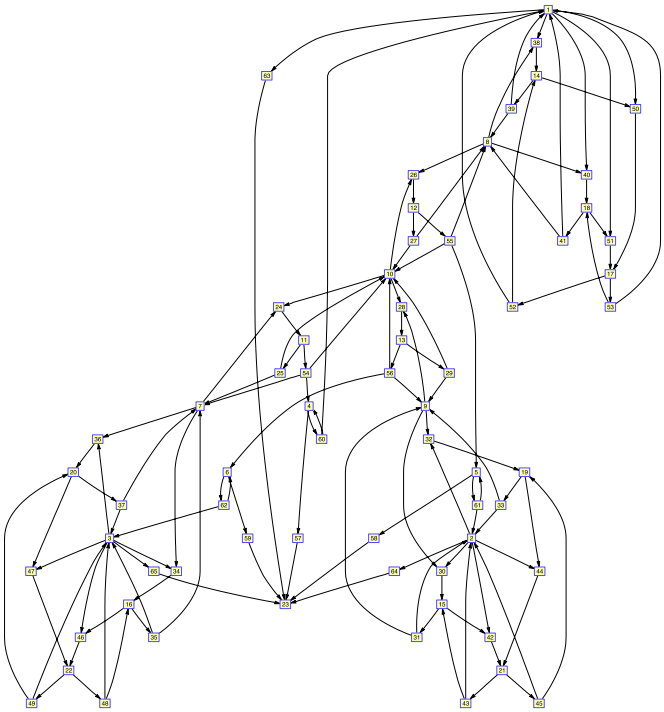


Fig. 1. A 65-node, 123-arc road network.

A. Test Scenario: A Simple Network

We first test our approach on a simple road network to validate accuracy of the solution and to gain useful insights of the solution. Consider the 65-node, 123-arc network in Fig. 1, which is a directed graph. This is a fairly representative spatial network with the following features. Firstly, the graph contains cycles, and some arcs between two nodes are bi-directional. Secondly, there are some clusters in this graph which are connected with each other. These two features make the graph closely imitating actual traffic networks.

Our approach is data-driven, which needs travel time samples on each arc as input (i.e., W_{ij} in Eq. (20)), and we adopt some random distribution functions to generate them. In the following experiments, we consider Normal, Bi-Normal, Gamma and Log-normal distribution and their combinations. We randomly select an origin-destination pair out of ten: {(3, 7), (7, 3), (1, 8), (8, 1), (10, 2), (2, 10), (1, 10), (10, 1), (3, 10), (10, 3)}.

B. Description of Symbols and Settings

- S—size of training data for travel time samples on each arc. We take values of 100, 500, and 1000.
- S_1—size of testing data for travel time samples on each arc. We take values of 40, 200, and 400.
- α—deadline coefficient with respect to T: $T = T_1 + \alpha * (T_2 - T_1)$, where T is deadline, T_2 is the minimum longest travel time for all paths connecting origin and destination, and T_1 is the shortest travel time with respect to the same path. We take $\alpha = 0.2, 0.4, 0.6, 0.8, 1.0$, and 1.2.
- N_E—number of times to run our approach to reach target accuracy, which is 1000 in this experiment.
- N—denotes Normal distribution.
- Bi—denotes Bi-normal distribution.
- G—denotes Gamma distribution.
- L—denotes Log-normal distribution.
- $N + Bi$—denotes Normal distribution combined with Bi-normal distribution.
- $N + G$—denotes Normal distribution combined with Gamma distribution.
- $N + L$—denotes Normal distribution combined with Log-normal distribution.
- O-D pair—denotes origin and destination pair.
- rtsots—denotes ratio of training size over testing size.
- ℓ_1-*norm*—denotes ℓ_1-norm minimization algorithm.
- *iterated* ℓ_1-*norm*—denotes iterated weighted ℓ_1-norm minimization algorithm.
- *reweighted* ℓ_1-*norm*—denotes reweighted ℓ_1-norm minimization algorithm.
- Both *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm* algorithms apply 10 iterations.

C. Case Study 1: Single Independent Distribution

For each arc, we generate S training data and S_1 testing data according to the four probability distributions. For each distribution, the data are generated with different parameters, and independent from each other. Then the S training data for each arc will be incorporated into Eq. (20) to compute the optimal path using three ℓ_1-norm based algorithms. Then, this path will be compared with the real optimal path which is obtained based on the S_1 testing data for each arc. To show that this result is only the lower bound accuracy, we also compute the accuracy measured on both training and testing data together for the ℓ_1-*norm* algorithm, as stated previously at the beginning of Section IV. Moreover, this process will repeat N_E times and the four corresponding accuracy results are shown in Table I, structure of which is stated as follows:

1) 1st column stands for types of distributions;
2) 2nd column stands for values of deadline coefficient α;
3) In each of the 3rd–5th columns, there are four sub-columns. The first sub-column is accuracy for the ℓ_1-*norm* based on both training and testing data, the second to the fourth sub-column are respectively accuracy for ℓ_1-*norm*, *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm* based on only testing data.

From Table I, we observe that the minimum accuracy for three ℓ_1-norm based algorithms is above 92%, and most of them are higher than 95%. The overall accuracy is sufficiently high considering the following facts. Firstly, optimal path is computed from the training data, while we evaluate the optimal path by testing data, and higher accuracy means better prediction. Secondly, ℓ_1-norm based algorithms are only approximations of cardinality minimization problem.

Comparing three ℓ_1-norm based algorithms with each other, we observe that *reweighted* ℓ_1-*norm* always has higher accuracy than that of *iterated* ℓ_1-*norm*. Furthermore, *iterated* ℓ_1-*norm* has higher accuracy than that of ℓ_1-*norm*. Although there are some exceptions, which are highlighted with bold font in Table I, they are already sufficiently high. Therefore, we

can conclude that *reweighted* ℓ_1*-norm* is better than *iterated* ℓ_1*-norm*, and *iterated* ℓ_1*-norm* is better than ℓ_1*-norm* in terms of accuracy. It is straightforward to understand this because of their different degrees of cardinality approximation.

We observe that when deadline coefficient α is larger than 1, i.e., 1.2, which corresponds to the situation that there exists at least one path guaranteeing arriving on time with 100% probability, our approach can always correctly find the actual optimal path. It is expected since ℓ_1-norm minimization in our approach is to minimize total delay with respect to deadline T. If T is large enough, there always exists at least one path with zero probability of being late. When α is not larger than 1, i.e., 0.2, 0.4, 0.6, 0.8, 1.0, which corresponds to the situation that there is no path that can guarantee arriving on time with 100% probability, the accuracy always falls between 92%–100%.

Regarding data size, we observe that accuracy increases when data size S increases from 100 to 500. This is because that more data we sampled, the closer frequency is to real probability. However, we observe limited similar increase when S increases from 500 to 1000. One possible reason is that data size of 500 is already large enough and further more data does not enhance the accuracy.

Another notable result is that four different probability distributions share a similar pattern for the accuracy under different deadlines and data sizes. The main reason is that the proposed approach is directly based on data, and Eq. (17) only takes the sampled data into account, and it is not affected by the probability distribution used.

Last but not least, comparing the first sub-column with the second sub-column of the 3rd–5th columns, we see that, most of the accuracy results measured on both training data and testing data together are higher than that of only on testing data. This is because in-sample test usually achieves higher accuracy than the out-of-sample test [36]. Although there are some exceptions, which are highlighted with underlines, they are still acceptable since accuracy in first sub-column and second sub-column are both high and close to each other.

Based on above analysis, we can basically conclude that our approach is able to handle different independent distributions with different deadlines. Especially, when data size is large (i.e., $S = 500$), our approach can achieve high accuracy.

D. Case Study 2: Blended Distributions

For each arc, we adopt combinations of probability distributions to generate S training data and S_1 testing data. We first use the sequence in incidence matrix $\mathbf{M}$ to order the arcs. Then, at each time, odd arcs use a probability distribution, and even arcs will use a different distribution. The combinations of probability distributions are set as follows: N + Bi, N + G, and N + L. We also assume that data on different arcs is independent from each other. Similar with Case Study 1, four different accuracy results are shown in Table II.

From Table II, we observe that the minimum accuracy for three ℓ_1-norm based algorithms is above 89%, and most of them are higher than 93%. It is sufficiently high considering that the traffic data are mixture of different distributions. Besides, we observe that *reweighted* ℓ_1*-norm* always has higher accuracy than that of *iterated* ℓ_1*-norm*, and *iterated* ℓ_1*-norm* has higher accuracy than that of ℓ_1*-norm*. There are also very few exceptions highlighted with bold font, which are acceptable. Similarly, the accuracy results for ℓ_1*-norm* measured on both training and testing data together are higher than that of only on testing data. Although there is exception highlighted with underline, it is also acceptable.

Comparing the results with Case 1, accuracy for Case 2 shares similar pattern under different deadlines and data size. From Table II, our approach can accommodate blended distributions with different deadlines because our approach is data driven, which is not affected by distribution types.

TABLE IV
TIME COMPLEXITY FOR THE ENUMERATION METHOD AND ℓ_1-NORM BASED ALGORITHMS

	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
enumeration method	0.6419	0.6389	0.6124
ℓ_1*-norm*	0.0546	0.2042	0.4052
iterated ℓ_1*-norm*	0.3412	0.7308	1.7123
reweighted ℓ_1*-norm*	0.4020	0.8736	1.9052

E. Case Study 3: Correlated Distributions

For each arc, we first generate S training data and S_1 testing data according to the four probability distributions. Then, we randomly choose some adjacent arc pairs, the travel time on which are correlated with each other, i.e., the data on an arc is proportional to the other. Similar with the first two cases, the four different accuracy results are shown in Table III.

From Table III, we observe that the minimum accuracy for three ℓ_1-norm based algorithms is above 94%, and most of them are higher than 96%. The overall accuracy is sufficiently high considering that existing methods cannot address the correlation in probability distribution. Comparing the results with Case 1 and Case 2, Case 3 share similar pattern with them. However, accuracy for Case 3 is slightly higher than that of Case 2 on average. This is because, in general, compared with correlated distribution, the blended distributions always generate larger variance for travel time samples on a path. Since accuracy results in Case 2 and Case 3 are obtained based on a learning scheme, training data and testing data might not be similar with each other if the variance is too large, which generally is not desirable to achieve high accuracy. Thus, we observe higher accuracy for Case 3 compared with Case 2. Similar explanation can also be applied in analysis of distribution parameters in Section V-A. From Table III, our approach can address correlation issue well.

F. Time Complexity

In all previous cases, to determine whether our approach can achieve an optimal path, we use enumeration method to compute the actual optimal path. To evaluate the overall computation complexity, we record all the running time for the above experiments. Additionally, the average running times for different sizes of travel time data and different algorithms are shown in Table IV.

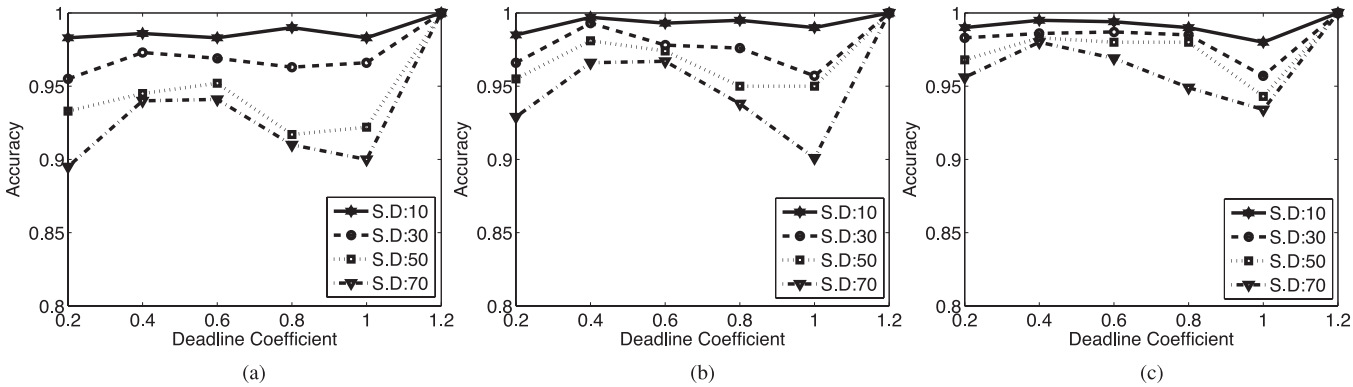


Fig. 2. Accuracy with respect to different standard deviations. (a) Data size: 100. (b) Data size: 500. (c) Data size: 1000.

From Table IV, we observe that the average running time for ℓ_1-*norm* is always shorter than that of enumeration method under different sizes of travel time data. The important reason is that MILP can be solved more efficiently than enumeration method. In this table, time complexity for enumeration method is relatively constant because it depends mainly on the network size. Although running time always increases with size of travel time data for ℓ_1-*norm* algorithm, we do not need a very large size of travel time data to obtain satisfactory solutions based on conclusions for the 3 cases that we studied. This means that we can obtain an optimal path faster than enumeration method.

Comparing *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm* algorithms with enumeration method, we observe that when size of training data is 100, the running times for *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm* algorithms are shorter than that of enumeration method. When data size increases to 500 and 1000, it has a reverse effect. However, one fact is that when network size increases, the running time and storage space will become prohibitively large for enumeration method (see Section V-D). By contrast, the time complexity for *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm* algorithms will also increase as network scales up. Nevertheless, they are still acceptable since the two algorithms are still more efficient with respect to time complexity in that case (see Section V-D).

Comparing three ℓ_1-norm based algorithms with each other, we observe that *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm* algorithms have higher time complexity. This is because the two algorithms involve 10 iterations for each optimal path. For ℓ_1-*norm*, it inherently solves only one MILP problem. For *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm*, they compute 9 relaxed MILP (linear programming in nature) problems before the last real MILP. Since the time complexity for linear programming is much less than that of MILP, it is much less than 10 times of time complexity of ℓ_1-*norm* algorithm.

The accuracy of ℓ_1-*norm* algorithm in all 3 cases is sufficiently high (although not the highest) and *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm* algorithms have much higher time complexity, although they have comparatively higher accuracy. Therefore, we conclude that ℓ_1-*norm* has better overall performance.

V. Parameter Analysis

In Section IV, we perform simulations to evaluate accuracy and time complexity of the proposed approach, which involve various parameters. In this section, we individually investigate their influences on the performance of our approach, i.e., accuracy and time complexity. The *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm* algorithms are comparatively complicated, but they are both developed based on ℓ_1-*norm*. Therefore, we only take the latter as an example. For all experiments in this section, at each time, we randomly generate travel time samples for each arc. To obtain accuracy and averaged running time, we repeat the experiment for 1000 times. Moreover, we remark that in this section, the accuracy are measured only based on testing data, which usually results in the lower bound, as justified in Section IV.

TABLE V
Averaged Running Time for Different Standard Deviations

	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
10	0.0403	0.1993	0.4018
30	0.0546	0.2042	0.4052
50	0.0599	0.2436	0.4872
70	0.0571	0.2630	0.4659

A. Distribution Parameters

Generally, probability distributions have two parameters, i.e., mean and standard deviation. To investigate their influence on the performance, we only take normal distribution as an example. We vary standard deviation, i.e., 10, 30, 50, and 70, while keeping other parameters fixed (e.g., O-D pair and ratio between the sizes of training and testing data). The accuracy results are shown in Fig. 2.

From Fig. 2, we observe that most of accuracy is still above 90% (except one instance at 89%) for standard deviation of 70. We also observe that as standard deviation increases, accuracy decreases under same deadline coefficient and data size. It is not difficult to understand that if standard deviation is 0, the accuracy by ℓ_1-*norm* will be 100% because it becomes a deterministic problem. If standard deviation is large, the similarity between training data and testing data greatly decreases, and accordingly, the accuracy will be detrimental. Additionally, the averaged running time is shown in Table V for different standard deviations and data sizes.

From Table V, we observe that time complexity increases as standard deviation increases from 10 to 50. This is due to the fact that when standard deviation is 0, there is only one effective

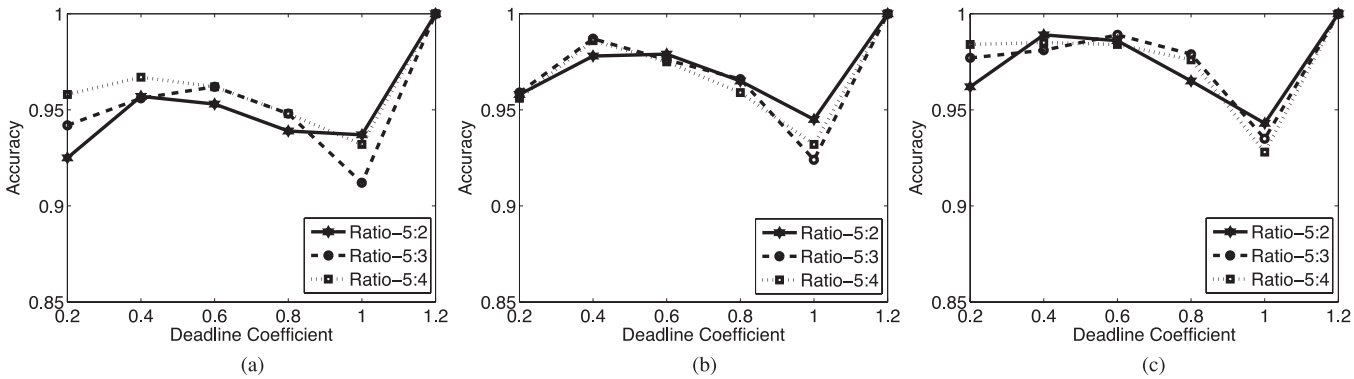


Fig. 3. Accuracy under different values of Rtsots. (a) Data size: 100. (b) Data size: 500. (c) Data size: 1000.

TABLE VI
AVERAGED RUNNING TIME FOR DIFFERENT RTSOTS

	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
5:2	0.0395	0.1836	0.3769
5:3	0.0382	0.1963	0.3721
5:4	0.0419	0.1880	0.3694

inequality constraint in the MILP problem. Thus, computation complexity is small. However, computation complexity may not always increase as standard deviation continues to increase, which can be observed when standard deviation changes from 50 to 70. This is because that the input data also affects time complexity, but there is an upper bound for the complexity to solve MILP problem, which was analyzed in Section III-E.

B. Ratio of Training Size Over Testing Size

In our approach, accuracy is computed based on prediction, which is similar to the prediction problem in machine learning. In machine learning, ratio of training size over testing size (*rtsots*) is an important parameter to evaluate the accuracy. Accordingly, we investigate its influence on our approach. We vary the value of *rtsots* from 5:2 to 5:4 while keeping other parameters fixed. The results are shown in Fig. 3.

From Fig. 3, we observe that there is no clear trend that accuracy will be affected by *rtsots*. We explain this by combining two points. Firstly, our problem is similar to the prediction in machine learning, and according to the machine learning theory, more training data and less testing data usually lead to higher accuracy. This means that, in our problem, higher *rtsots* may achieve better accuracy. Secondly, on the other hand, our question is not exactly same as the machine learning problem, because metric of this stochastic optimal solution is probability. When testing data size is comparatively large, it approximates the probability better and share a similar pattern with training data (i.e., similar probability of not being later than deadline). It means that lower *rtsots* may achieve comparatively higher accuracy, especially when traffic data size is large enough. When these two points compromise with each other, we may not observe a clear pattern regarding accuracy and *rtsots*. Meanwhile, we also record the averaged running time with respect to different *rtsots* in Table VI.

As expected, we do not observe obvious pattern between *rtsots* and time complexity in Table VI, because optimal path is obtained from training data while we validate it on testing data. Thus, the running time will not be influenced by *rtsots*.

C. O-D Pair

The locations of O-D pair are also important parameters that may impact accuracy. We investigate the influence by changing O-D pair while keeping other parameters unchanged. We choose following O-D pairs: 1-8, 1-10, 1-3, and 1-48, and there are respectively 8, 1168, 2384, and 2784 different connected paths between them. The results are shown in Fig. 4.

From Fig. 4, we observe that for most cases, as number of connected paths increases, the accuracy decreases except deadline coefficient of 0.8 in Fig. 4(b) and 0.6 in Fig. 4(c). It is reasonable considering that if there is only one possible path between O-D pair, the accuracy will always be 100%. Since our approach is an approximation and traffic data is all random, if there are a large number of candidate paths, the chance that our solution is the actual optimal path will decrease. However, in real world road networks, there may not be too many connected paths between the concerned O-D pair for driver to choose. Therefore, accuracy will not be degraded significantly. Moreover, we record the averaged running time for different O-D pairs in Table VII.

From the first three O-D pairs in Table VII, we observe that when number of connected paths between O-D pair becomes large, averaged running time accordingly increases. It is reasonable because if there is only one path between the pair, running time should be very short. If the number is large, it will take longer time to search the best path. However, time complexity may not constantly increase, because MILP solver adopts efficient algorithms to find optimal solution although the worst case (exponential computation complexity) is possible to happen.

D. Graph Scale and Data Size

To analyze the influence brought by graph scale and data size, we implement our approach on another comparatively large artificial network with 100 nodes and 220 arcs. For this

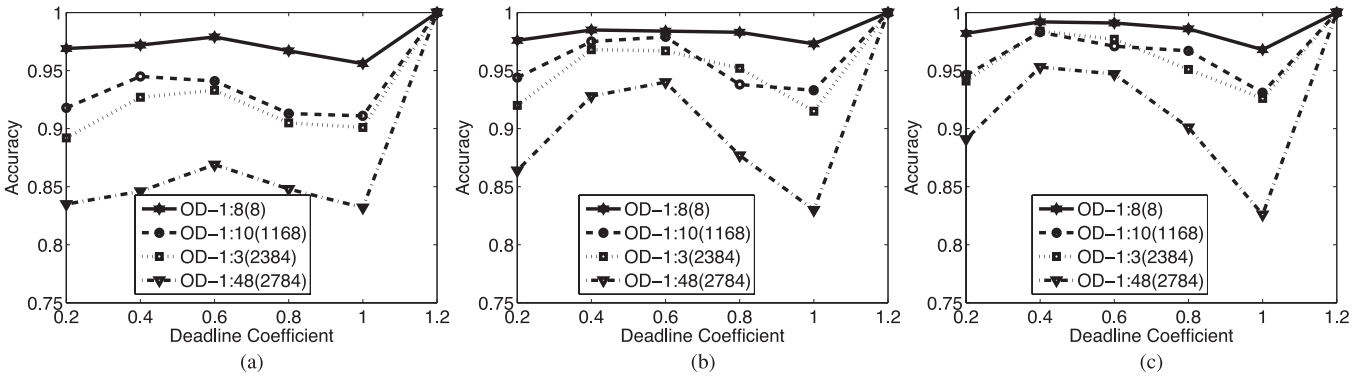


Fig. 4. Accuracy with respect to different O-D pairs. (a) Data size: 100. (b) Data size: 500. (c) Data size: 1000.

TABLE VII
AVERAGED RUNNING TIME FOR DIFFERENT O-D PAIRS

	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
1-8 (8)	0.0280	0.1352	0.3102
1-10 (1168)	0.0390	0.1958	0.3798
1-3 (2384)	0.0452	0.2210	0.3941
1-48 (2784)	0.0443	0.2324	0.3886

large network, as shown in Fig. 5, we randomly select O-D pairs out of (40, 43) and (40, 51), whose numbers of connected paths are respectively 2 403 060 and 4 529 052. Regarding other settings, they are the same as previous simulations for the small road network in Fig. 1. Accordingly, the accuracy for these two graphs are plotted in Fig. 6.

From Fig. 6, we can see that graph scale affects accuracy and small graph can achieve comparatively higher accuracy. We note that the average number of connected paths for small graph is around 2000 while large graph is around 3 000 000. Moreover, the reason for different accuracy can be explained by the same analysis for O-D pair. As for the influence caused by data size alone, it has been analyzed previously, and the accuracy could improve as data size becomes large, i.e., from 100 to 500. However, it becomes saturated, i.e. from 500 to 1000. Additionally, more obvious impact on our approach caused by graph scale and data size is time complexity, which is recorded in Table VIII.

From Table VIII we can see that enumeration method becomes prohibitively time consuming as graph scales up, which uses more than 4,000 seconds to compute optimal path. The obvious reason is that the number of connected paths between O-D pair is huge on this large graph and enumeration does not work efficiently. By contrast, time complexity of ℓ_1-*norm* on large graph is only around 1 second, which is considerably efficient because it uses smart optimization techniques to search the desired path. However, considering ℓ_1-*norm* alone, time complexity also increases as graph and data size scale up. This is reasonable as we have analyzed in Section III-E, our approach is based on MILP, which is solved by branch-and-bound method. Additionally, the worst complexity for branch-and-bound method is $\Theta(2^{|A_r|}(|A_r|+S)^3)$, which increases as network and data size scale up. However, as we observe from all previous experiments, the worst case seldom happens.

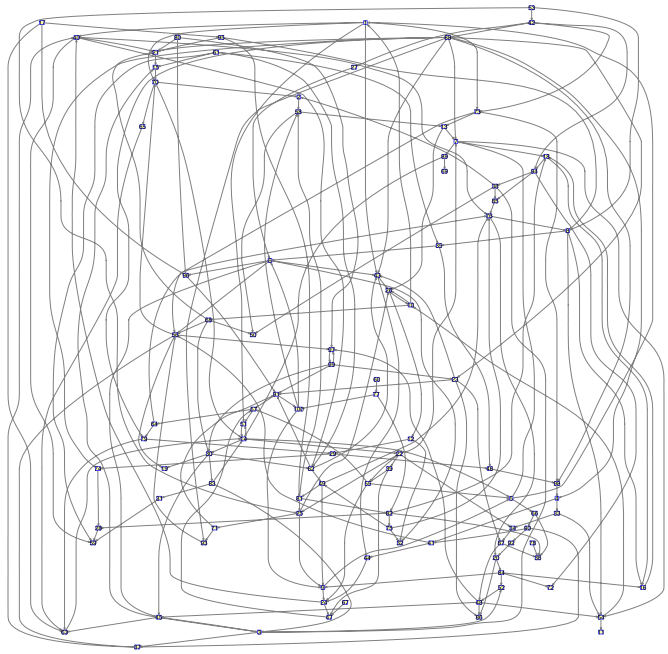


Fig. 5. A 100-node, 220-arc road network.

E. Deadline Coefficients

Deadline coefficient impacts accuracy, which can be justified from Figs. 2–4, and 6. Especially, for coefficient larger than 1, i.e., 1.2, accuracy always reaches 100%. The detailed reason can be found in the case studies presented in Section IV. As to coefficients between 0 and 1, there is no regular pattern for the accuracy. Moreover, to show the impact on time complexity, we also record the averaged running time in Table IX for different deadline coefficients while keeping other parameters unchanged. From Table IX we observe that, time complexity does not change with deadline coefficients regularly. This is reasonable because in the MILP problem, deadline coefficient impacts only inequality constraints, but does not change the number of these constraints. By contrast, it may impact the branching operation in MILP, which can be evaluated only by the worst case scenario.

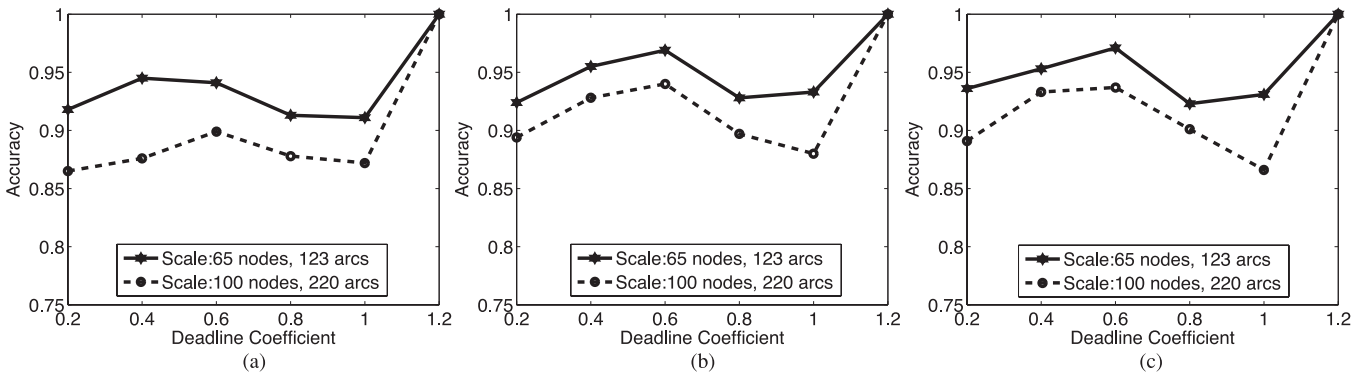


Fig. 6. Accuracy with respect to different graph scale and data size. (a) Data size: 100. (b) Data size: 500. (c) Data size: 1000.

TABLE VIII
AVERAGED RUNNING TIME FOR DIFFERENT GRAPH SCALE AND DATA SIZE

	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
enumeration for large graph	4670.5	4830.1	4781.5
ℓ_1-*norm* for large graph	0.7207	0.9816	1.3198
enumeration for small graph	0.6980	0.6932	0.6970
ℓ_1-*norm* for small graph	0.0539	0.2267	0.3996

TABLE IX
AVERAGED RUNNING TIME FOR DIFFERENT DEADLINE COEFFICIENTS

	100, 40	500, 200	1000, 400
0.2	0.0240	0.1561	0.2983
0.4	0.0367	0.1683	0.2836
0.6	0.0344	0.1625	0.3041
0.8	0.0436	0.1438	0.3249
1.0	0.0359	0.1583	0.3039
1.2	0.0408	0.1469	0.3329

VI. TESTING ON REAL WORLD ROAD NETWORK

In this section, we evaluate our approach using an area of Munich city, which is shown in Fig. 7. The underlying graph includes 170 nodes and 277 arcs. The travel time samples on each road link are extracted from real GPS trajectories of BMW vehicles. The numbers of those samples on each road link are not same, the minimum of which is 880 and the maximum is 41,256. Other settings are similar to that in Section IV. Since the input data is from real GPS trajectories, there are no assumptions of probability distribution or correlation. Moreover, we synthesize the input data as follows:

1) Sort the data for each arc according to the time when they were generated;
2) Divide the data for each arc into two parts: the first 80% forms training data set, and the remaining forms testing data set;
3) For each arc, randomly sample S data from first part as training data and S_1 data from second part as testing data, and set $S : S_1 = 5: 3$;
4) Repeat 1000 times to obtain accuracy measure.

Note that we measure accuracy based only on testing data instead of training data and testing data together, which usually results in lower bound accuracy, as justified in Section IV.

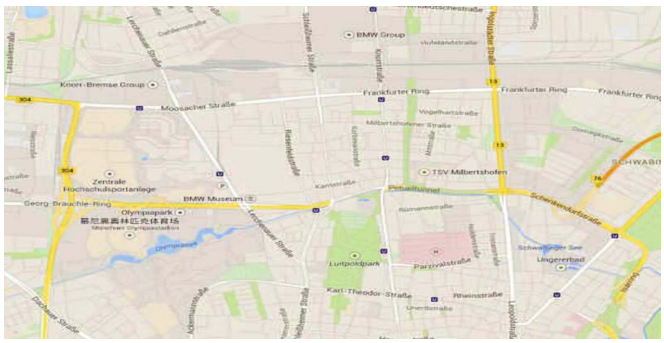


Fig. 7. An area of the Munich city, Germany.

Additionally, we implement the state-of-the-art algorithm in the same network. The algorithm assumes that travel time for each road link follows an independent normal distribution. If there exists at least one path whose expected travel time is no longer than deadline, this problem can be solved by quasi-convex optimization technique efficiently. Otherwise, it can only find an optimal solution by an inefficient enumeration method. Its details are described in [12]. For better comparison, we adopt this method according to the following settings:

1) Fit the best normal distribution for each arc based on the S training data;
2) Use state-of-the-art algorithm to find the optimal path;
3) Validate this optimal path using the testing data, and repeat 1000 times to obtain the accuracy measure.

Based on above settings, we obtain the accuracy results, which are shown in Fig. 8. Comparing the three sub-figures, we observe that, as expected, *reweighted* ℓ_1-*norm* algorithm always achieves higher accuracy than that of *iterated* ℓ_1-*norm* algorithm, and *iterated* ℓ_1-*norm* obtains higher accuracy than that of ℓ_1-*norm* algorithm. The reason has been previous analyzed. Considering the accuracy for data size of 100, ℓ_1-*norm* always has better accuracy than that of the state-of-the-art algorithm. As data size increases to 500 and 1000, accuracy of the state-of-the-art algorithm for some coefficients exceeds that of ℓ_1-*norm* algorithm. This is because the state-of-the-art algorithm can work well only for normal distribution. When size of the data is small, such as 100, the fitting between normal distribution and real data is usually poor. As data size becomes

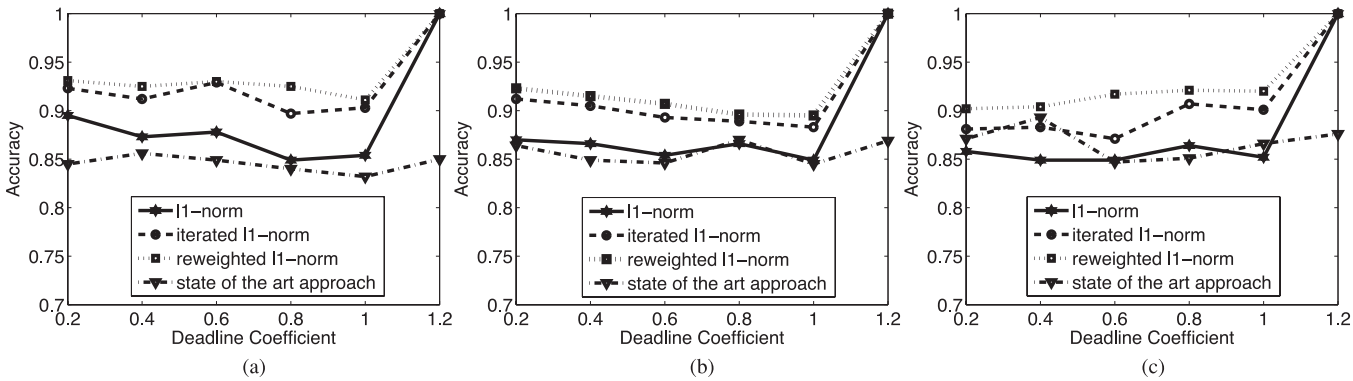


Fig. 8. Accuracy for the ℓ_1-norm based algorithm and the state-of-the-art algorithm. (a) Data size: 100. (b) Data size: 500. (c) Data size: 1000.

TABLE X
TIME COMPLEXITY FOR THE ℓ_1-NORM BASED ALGORITHMS AND THE STATE-OF-THE-ART ALGORITHM

α	0.2	0.4	0.6	0.8	1.0	1.2
ℓ_1-*norm*	0.454	0.435	0.465	0.476	0.472	0.464
iterated ℓ_1-*norm*	1.541	1.730	1.712	1.860	1.687	1.702
reweighted ℓ_1-*norm*	1.602	1.873	1.914	1.937	1.837	1.821
state-of-the-art	0.808	0.721	0.554	0.443	0.453	0.450

large, the sampled data will become close to the distribution. However, its accuracy is significantly affected by probability distribution and data size. By contrast, our ℓ_1-*norm* based algorithms are not strictly impacted by them, which has been previously analyzed. However, regarding the accuracy for large deadline coefficients, i.e., 1.2 or larger, our approach always achieves 100% accuracy, which clearly outperforms the state-of-the-art algorithm. Moreover, it is also important to note that, the accuracy measured for ℓ_1-*norm* algorithm in Fig. 8 is only the lower bound.

One important observation is that, the overall accuracy for three ℓ_1-*norm* based algorithms is not as high as those in Sections IV and V. This is because we made some assumptions there to analyze the factors that may impact the performance of our approach (e.g., keeping the same distribution, standard deviation or O-D pairs). By contrast, the traffic data in this section is collected from real vehicle trajectories, and there are consequently no such assumptions. However, considering that the real traffic data is always random and most of accuracy is above 85%, the ℓ_1-*norm* based algorithms achieve good performance for the real world road network and traffic. Additionally, the deadline is an important parameter for the state-of-the-art algorithm. To show the impact on time complexity caused by deadline coefficients, we record the averaged running time in Table X.

From Table X, we observe that ℓ_1-*norm* algorithm is more efficient than *iterated* ℓ_1-*norm* algorithm, and *iterated* ℓ_1-*norm* algorithm is more efficient than *reweighted* ℓ_1-*norm* algorithm in terms of time complexity. This efficiency is obtained at price of accuracy. Comparing ℓ_1-*norm* and the state-of-the-art algorithm, we observe that, when deadline coefficient is small, averaged running time for the state-of-the-art algorithm is comparatively longer. The underlying reason is that only if there exists the path whose expected travel time is not longer than deadline, the state-of-the-art algorithm can work efficiently. Otherwise, it will employ an enumeration method to determine the optimal path, which is inefficient. Since at each time, the travel time is randomly sampled from the real data, if deadline coefficient is small, the chance that the smallest expected travel time is larger than deadline increases. Consequently, we observe higher running time for small coefficient. As coefficient becomes larger, that chance will decrease. Consequently, the corresponding running time will also become short. However, it is not always becoming shorter for the state-of-the-art algorithm, because once there exists the path whose expected travel time is not longer than deadline, the time complexity will not change any more. And this result can be observed from Table X.

Combining both accuracy and time complexity together, we found that ℓ_1-*norm* algorithm and the state-of-the-art algorithm have their own advantages. When data size is comparatively small, i.e., 100 and 500, the accuracy of ℓ_1-*norm* algorithm is always higher (at least not lower) than that of the state-of-the-art algorithm. When data size is large, i.e., 1000, and deadline coefficient is small, i.e., 0.2 and 0.4, accuracy for the state-of-the-art algorithm is higher. However, for all different data sizes, as long as deadline coefficient is large, i.e., not less than 1.2, accuracy for the ℓ_1-*norm* algorithm is always 100%, which is much higher than that of the state-of-the-art algorithm. We also highlight that, the deadline coefficient is determined by driver, and it can be any positive values. We only consider the range of 0.2–1.2 for the sake of conveniently analyzing the factors that may affect accuracy. More importantly, as stated in Sections IV and V, accuracy obtained for ℓ_1-*norm* algorithm in Fig. 8 is only the lower bound. By contrast, time complexity of the state-of-the-art algorithm is obviously affected by deadline coefficient. When deadline coefficient is small, its time complexity is much higher than that of ℓ_1-*norm* algorithm. Furthermore, as deadline coefficient increases, its time complexity approximately converges, which is slightly lower than that of ℓ_1-*norm* algorithm. Taking all these into account, we can conclude that ℓ_1-*norm* algorithm is better than the state-of-the-art algorithm. Additionally, comparing ℓ_1-*norm* algorithm with *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm* algorithms, we conclude that ℓ_1-*norm* algorithm is better considering both accuracy and time complexity.

We would like to highlight that our approach is data-driven. It largely relies on travel time samples of each road link, which can be obtained by processing the GPS trajectories of vehicles. In a metropolitan city, daily traffic may involve millions of vehicles, almost all of which are equipped with GPS devices. Therefore, it is easy and feasible to deploy our approach into realistic vehicle routing by exploring the big traffic data. Generally, as analyzed previously, if our approach adopts huge amount of the traffic data, accuracy is expected to be higher, but running time is also accordingly longer. On the other hand, if the size of traffic data is small in our approach, accuracy may decrease although running time is shorter. Therefore, we need to seek an optimal balance between them.

VII. Conclusion and Discussion

This paper aims at solving a stochastic shortest path problem in vehicle routing. The objective is to determine an optimal path that maximizes probability of a vehicle arriving at destination not later than deadline. This problem is usually difficult to solve unless we apply some assumptions on travel time probability distribution, correlation or deadline. To address these issues, we have transformed the problem into a cardinality minimization problem, and further used an ℓ_1-norm technique and its variants to solve the problem. The simulation results on a road network have shown that the algorithm works well under a variety of probability distributions. The performance is not affected even when we consider travel time dependencies. Moreover, it can solve the problem with various deadlines. By analyzing important parameters, we have found that to some extent, the accuracy and time complexity of the approach are subject to standard deviation of travel time, ratio between the sizes of training and testing data sets, origin and destination pair, deadline coefficient, graph scale and data size. When tested on real world road network with real traffic data, the results show that ℓ_1-*norm* algorithm outperforms the state-of-the-art algorithm. Comparing ℓ_1-*norm* algorithm with its two variants, we have concluded that ℓ_1-*norm* algorithm is a better choice in terms of both accuracy and time complexity.

We also would like to note that, with ℓ_1-*norm* algorithm, searching the stochastic shortest path in this paper finally becomes a MILP problem. In this MILP problem, decision variable size is the sum of arc amount in the graph and traffic data size for each arc. Moreover, the traffic data size is also the number of inequality constraints, and node amount in the graph is the number of equality constraints. To our best knowledge, the worst computation complexity of this MILP problem will exponentially increase as graph or traffic data size scale up, and currently there is no efficient (i.e., polynomial computation complexity) method to solve it. Although the central routing engine server is powerful, the computation may still become prohibitively time-consuming because usually real world map scale or traffic data size is very large. Accordingly, in the future, we try to improve the computation complexity of this MILP problem:

1) We will use the approximate/heuristic algorithms, because ℓ_1-*norm* algorithm only approximately solves the cardinality minimization problem, and in this sense solution to the MILP problem does not need to be accurate as long as it is acceptable;
2) Different from general MILP problem, the equality constraint in our problem satisfies total unimodularity, and we will utilize this characteristic to solve the problem efficiently. In addition to the computation complexity, we will also study cooperative stochastic shortest path among multiple vehicles.

References

[1] H. Guo, Z. Cao, J. Zhang, D. Niyato, and U. Fastenrath, "Routing multiple cars in large scale networks: Minimizing road network breakdown probability," in *Proc. 17th IEEE ITSC*, 2014, pp. 2180–2187.

[2] E. D. Miller-Hooks and H. S. Mahmassani, "Least expected time paths in stochastic, time-varying transportation networks," *Transp. Sci.*, vol. 34, no. 2, pp. 198–215, May 2000.

[3] B. C. Dean, "Algorithms for minimum-cost paths in time-dependent networks with waiting policies," *Networks*, vol. 44, no. 1, pp. 41–46, Aug. 2004.

[4] S. T. Waller and A. K. Ziliaskopoulos, "On the online shortest path problem with limited arc cost dependencies," *Networks*, vol. 40, no. 4, pp. 216–227, Dec. 2002.

[5] E. Nikolova, J. A. Kelner, M. Brand, and M. Mitzenmacher, "Stochastic shortest paths via quasi-convex maximization," in *Algorithms-ESA 2006*. Berlin, Germany: Springer-Verlag, 2006, pp. 552–563.

[6] E. Nikolova, "Approximation algorithms for reliable stochastic combinatorial optimization," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Berlin, Germany: Springer-Verlag, 2010, pp. 338–351.

[7] C. E. Sigal, A. A. B. Pritsker, and J. J. Solberg, "The stochastic shortest route problem," *Oper. Res.*, vol. 28, no. 5, pp. 1122–1129, Sep./Oct. 1980.

[8] M. D. Becker and U. D. Fastenrath, "Verfahren zur Auswahl einer Route für eine dynamische Navigation von Individualverkehr," DE Patent App. DE200 710 060 590, Jun. 18, 2009.

[9] H. Frank, "Shortest paths in probabilistic graphs," *Oper. Res.*, vol. 17, no. 4, pp. 583–599, Jul./Aug. 1969.

[10] Y. Fan, R. Kalaba, and J. Moore, II, "Arriving on time," *J. Optim. Theory Appl.*, vol. 127, no. 3, pp. 497–513, Dec. 2005.

[11] Y. M. Nie and X. Wu, "Shortest path problem considering on-time arrival probability," *Transp. Res. B, Methodol.*, vol. 43, no. 6, pp. 597–613, Jul. 2009.

[12] S. Lim, H. Balakrishnan, D. Gifford, S. Madden, and D. Rus, "Stochastic motion planning and applications to traffic," in *Algorithmic Foundation of Robotics VIII*. Berlin, Germany: Springer-Verlag, 2009, pp. 483–500.

[13] J. G. Wardrop, "Road paper. Some theoretical aspects of road traffic research," *Proc. Inst. Civil Eng.*, vol. 1, no. 3, pp. 325–362, May 1952.

[14] A. Polus, "A study of travel time and reliability on arterial routes," *Transportation*, vol. 8, no. 2, pp. 141–151, Jun. 1979.

[15] M. Mogridge and S. Fry, "Variability of car journey times on a particular route in central London," *Traffic Eng. Control*, vol. 25, no. 10, pp. 510–511, Oct. 1984.

[16] F. Montgromery and A. May, "Factors affecting travel times on urban radial routes," *Traffic Eng. Control*, vol. 28, no. 9, pp. 452–458, Sep. 1987.

[17] H. Rakha, I. El-Shawarby, M. Arafeh, and F. Dion, "Estimating path travel-time reliability," in *Proc. IEEE ITSC*, 2006, pp. 236–241.

[18] K. Chen, L. Yu, J. Guo, and H. Wen, "Characteristics analysis of road network reliability in Beijing based on the data logs from taxis," in *Proc. Transp. Res. Board 86th Annu. Meet.*, 2007, pp. 1–19.

[19] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[20] N. Kallus, "From predictions to data-driven decisions using machine learning," *ArXiv Preprint ArXiv:1402.5481*, 2014.

[21] D. Solomatine, "Applications of data-driven modelling and machine learning in control of water resources," in *Computational Intelligence in Control*. Hershey, PA, USA: IGI Global, 2002, pp. 197–217.

[22] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 25–34.

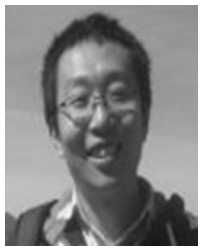
[23] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "A data-driven method for stochastic shortest path problem," in *Proc. 17th IEEE ITSC*, 2014, pp. 1045–1052.

[24] M. J. Abdi, "Cardinality optimization problems," Ph.D. dissertation, School Math., Univ. Birmingham, Birmingham, U.K., 2013.

[25] M. Fazel, H. Hindi, and S. P. Boyd, "A rank minimization heuristic with application to minimum order system approximation," in *Proc. Amer. Control Conf.*, 2001, vol. 6, pp. 4734–4739.
[26] S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky, "ℓ_1 trend filtering," *SIAM Rev.*, vol. 51, no. 2, pp. 339–360, May 2009.
[27] P. D. Tao and L. T. H. An, "The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems," *Ann. Oper. Res.*, vol. 133, no. 1–4, pp. 23–46, Jan. 2005.
[28] Y.-B. Zhao and D. Li, "Reweighted ℓ_1 minimization for sparse solutions to underdetermined linear systems," *SIAM J. Optim.*, vol. 22, no. 3, pp. 1065–1088, 2012.
[29] Mixed-Integer Linear Programming Algorithms, MathWorks, Natick, MA, USA, 2014. [Online]. Available: http://www.mathworks.com/help/optim/ug/mixed-integer-linear-programming-algorithms.html
[30] Y. Zhang, "Solving large-scale linear programs by interior-point methods under the MATLAB environment," *Optim. Methods Softw.*, vol. 10, no. 1, pp. 1–31, Jan. 1998.
[31] E. Danna, E. Rothberg, and C. Le Pape, "Exploring relaxation induced neighborhoods to improve MIP solutions," *Math. Programm.*, vol. 102, no. 1, pp. 71–90, Jan. 2005.
[32] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, vol. 18. New York, NY, USA: Wiley, 1988.
[33] M. Fazel, H. Hindi, and S. P. Boyd, "Log-det heuristic for matrix rank minimization with applications to Hankel and Euclidean distance matrices," in *Proc. Amer. Control Conf.*, 2003, vol. 3, pp. 2156–2162.
[34] W. Zhang, "Branch-and-bound search algorithms and their computational complexity," USC Inf. Sci. Inst., Marina del Rey, CA, USA, ISI/RR-96-443, 1996.
[35] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. 16th Annu. ACM Symp. Theory Comput.*, 1984, pp. 302–311.
[36] Y. Bengio *et al.*, "Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, vol. 16, pp. 177–184.

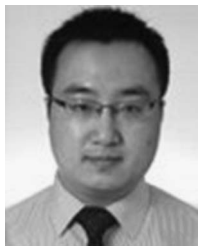
Zhiguang Cao received the B.Eng. degree in automation from Guangdong University of Technology, Guangzhou, China, in 2009 and the M.Sc. degree in signal processing from Nanyang Technological University, Singapore, in 2012. He is currently working toward the Ph.D. degree with the Future Mobility Research Laboratory, Nanyang Technological University.

His research interests include stochastic vehicle routing, convex optimization, and machine learning.



Hongliang Guo received the B.Eng. degree in dynamic engineering and the M.Eng. degree in dynamic control from the Beijing Institute of Technology, Beijing, China, and the Ph.D. degree in electrical and computer engineering from the Stevens Institute of Technology, Hoboken, NJ, USA.

In 2011, he joined Almende, Rotterdam, The Netherlands, as a Postdoctoral Researcher. His research interests include self-organizing systems and agent-based technologies. In 2013, he joined the Nanyang Technological University, Singapore, as a Research Fellow.



Jie Zhang received the Ph.D. degree in computer science from the University of Waterloo, Waterloo, ON, Canada, in 2009.

He is currently an Associate Professor with the School of Computer Engineering, Nanyang Technological University, Singapore. His research has been focused on the design of effective and robust intelligent software agents, through the modeling (trustworthiness and preferences) and simulation of different agents in a wide range of environments, using AI techniques and multi-agent technologies.

Dusit Niyato received the Ph.D. degree in electrical and computer engineering from the University of Manitoba, Winnipeg, MB, Canada, in 2008.

He is currently an Associate Professor with the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include optimization of wireless communication and mobile cloud computing, smart grid systems, and green radio communications.

Ulrich Fastenrath received the Ph.D. degree in theoretical solid state physics from the University of Cologne, Cologne, Germany.

He was a two-year Postdoctoral Fellow with the Technical University of Munich. In May 2012, he joined the BMW Group, Munich, Germany, where he is responsible for research and development activities in traffic and routing space and currently the Head of Traffic Information Management and Routing Optimization.