IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 19, 2019, accepted September 13, 2019, date of publication September 18, 2019, date of current version October 2, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2942282*

# Mobile Crowdsourcing for Intelligent Transportation Systems: Real-Time Navigation in Urban Areas

**XIANGPENG WAN, (Student Member, IEEE), HAKIM GHAZZAI, (Member, IEEE), AND YEHIA MASSOUD, (Fellow, IEEE)**

School of Systems and Enterprises, Stevens Institute of Technology, Hoboken, NJ 07030, USA

Corresponding author: Hakim Ghazzai (hghazzai@stevens.edu)

**ABSTRACT** Modern urbanization is demanding smarter technologies to improve a variety of applications in intelligent transportation systems. Mobile crowdsourcing enabling automatic sensing tasks constitutes an excellent mean to complement existing technologies. In this paper, we exploit the high amount of data that can be collected by on-board and infrastructure-based sensors to evaluate traffic network statuses and improve the navigation of vehicles in urban areas. The objective is to design real-time route planning algorithms that determine fastest trajectories for both single and multiple destinations, in a real-time manner based on the frequent data inputs. We first formulate the routing problems as integer linear programs (ILPs) and then, design iterative approaches levels to iteratively solve the ILPs while considering updated traffic data. Afterwards, lower complexity sub-optimal graph-based algorithms are designed to solve the real-time routing problems. Unlike traditional navigation solutions, the proposed approaches update the vehicle trajectory after a certain period characterized by timely correlated data. Uncertainty and erroneous data inputs are also considered to determine fastest and least risky trajectory. Our results show that crowdsourcing-based real-time navigation outperforms outperform traditional navigation solutions by selecting less congested roads and avoiding blocked streets.

**INDEX TERMS** Intelligent transportation systems, mobile crowdsourcing, uncertainty, real-time navigation, delivery vehicle problem.

## I. INTRODUCTION

With modern urbanization, the increasing amount of vehicular traffic flowing within cities is perpetually threatening to increase congestion dramatically. According to the latest Traffic Index 'TomTom', congestion has increased 23% globally from 2008 to 2016 and now is worse [2]. In New York city, drivers spend up to 35% extra travel time per day on average. It is even up to 62% on average during evening rush hours. Traffic congestion becomes a serious problem in modern urban cities, which does not only affect people's commuting time and temper, but it can also cause health-damaging air pollution and a big waste of energy. To cope with this problem, a lot of technological solutions have been proposed under the scope of smart cities. For instance, using vehicle-infrastructure technology, the city of Columbus, Ohio, controls the traffic signal time by monitoring the traffic flow throughout the day. In Atlanta, a 2.3-mile smart corridor was opened in 2017. The corridor contains tremendous sensors, cameras, and devices to collect data. It solves the issues related to mobility and safety. It improves the overall traffic efficiency, e.g. by controlling smart traffic signals [3]. In San Francisco, local authorities installed sensors alongside the roads to help drivers find on-street parking.

Such solutions effectively contribute to reducing traffic congestion issues and improving urban mobility, thanks to the emergence and spread of on-board and infrastructure-based sensors. With their help, collecting data becomes very common in urban areas [4], which may ease the access to traffic information and help determining road statuses in an instantaneous manner mainly when this data is shared among road users. This has led researchers to investigate an emerging concept, the vehicular social networking (VSN) [5], which effectively exploits the availability of data, especially in urban

The associate editor coordinating the review of this manuscript and approving it for publication was Kuo-Ching Ying.

areas [6], [7]. With VSN, vehicles can establish relationships with their peers and other devices based on common interests. This significantly empowers current intelligent transportation systems (ITS) with extra intelligence and enables new applications associated with traffic safety and traffic efficiency, such as accident reporting or slippery roads warnings [8].

Navigation methods which are currently in use, such as Google maps navigation, Waze navigation, and shortest path solutions are based on historical data, statistical records, and human inputs [9]. Those methods work perfectly in rural areas and highways where traffic is usually smooth and not disrupted by unexpected events. However, in urban areas, the road network is far more complex, with a high pedestrian density and unexpected events such as constructions and accidents that could frequently block entire roads. Hence, current navigation methods will not always provide the best routes. They might unbalance the traffic flow by guiding drivers to select the same routes, which become progressively the slower ones [10]. Moreover, they may guide drivers to blocked roads and indirectly contribute to traffic congestion. Therefore, based on instantaneous feedback of other vehicles and sensors, the proposed real-time route planning framework will efficiently deal with the aforementioned issues and potentially improve the vehicle navigation in urban areas.

### A. RELATED WORK

In recent years, a lot of studies investigating the benefits of data collection for different vehicular and ITS applications have been presented [11]. For instance, in [12], the authors proposed to exploit the collected data to detect on-street parking with mobile sensing that has comparable performance with fixed sensing. The advantage is that mobile sensing is much cheaper and available everywhere. The flaw is that mobile sensing works well for standard on-street parking where vehicles parked alongside streets and detectors need to be moving on the outside lanes. In [13], the authors extracted the road lanes information from different source including the Global Positioning System and taxi data, the accuracy is around 85%. There are studies related to advanced driving assistance systems that help detecting lane markings using deep learning [14], additionally, GPS is also a reliable data source to help improve the accuracy. In [15], the authors proposed the last mile navigation method for public vehicles. They introduced an algorithm to detect the last segment of drivers from the trajectory and take them as landmarks for ant optimization. In [16], the authors predicted short-term traffic conditions with trajectory data using Support Vector Machine, the authors classified the conditions into smooth, basically smooth, mild congestion, moderate congestion, and serious congestion. Those categories are used to predict the future traffic situation, however, the accuracy of this method is not well validated. Also, in [17], the authors introduced a new system that collects accident evidence from vehicular sensors and records it for future investigation to provide sufficient data source about traffic behavior supervision once implemented. Moreover, many crowdsourcing architectures and recruitment policies have been proposed to effectively exploit crowdsourced data to solve urban traffic problems [18], [19].

Apart from those contributions described earlier, there are some studies that worked on the real-time route planning problem in urban cities. In [20], the authors exploited the feedback system in a distributed-opportunistic way to optimize the route planning for drivers, and they evaluated the performance of their model using Markovian agents. Their algorithm outperforms the shortest path algorithm in a case study. In [21], the authors designed a system to detect anomalies with tested vehicles, and then select the most comfortable route based on the data collected about three metrics including: travel time, road roughness, and road remove, using the Dijkstra's algorithm. In [22], the authors refined a route-planning algorithm to improve the navigation in urban areas under traffic congestion based on fuel consumption. In [23], the authors proposed an improved navigation algorithm based on the Dijkstra's algorithm and tested it on a simple graph. In these papers, however, the authors do not take the real-time road information into consideration, where vehicles might be blocked by unexpected events, and they do not consider real-time traffic speed, which potentially guides drivers into congested roads.

On the other hand, there are other studies investigating the navigation problems for the multi-destinations scenario. In [24], the authors applied the k-shortest path method to find shortest trajectory towards one destination. They also extended the solution to the multi-destinations case using k-shortest path trees. They transformed the positions into nodes from the real map, but they did not take the real-time traffic data into consideration and their algorithms are not proven to be optimum. In [25], the authors introduced a novel layout model for generating multi-destination maps, they framed each destination with a square block and considered the distance and angles between each block. They also assigned different weights to roads, streets, and highways before selecting the best segregation according to their definition. However, they only considered the off-line map without real-time traffic data, and the destinations are not precisely reached for navigation. Some studies show that the navigation system can be embedded into autonomous vehicles that routes the vehicle through known or preprogrammed co-ordinates autonomously without any human control [26]. In the latter paper, the authors designed a navigation system that applies low-cost strap-down inertial measurement unit to navigate between the two geographical points based on GPS, but in practice, the breakthrough of both hardware and software are needed. With the crowdsourcing infrastructure, the system should become easier to implement.

### B. CONTRIBUTIONS

In smart urban area, mobile users and vehicles are capable of completing various tasks from different locations at the same time by exploiting VSN and crowdsourcing [27]. For instance, performing location-based query tasks and

automatic sensing tasks could provide VSN members with necessary information about the traffic network in real-time. In the automatic sensing task vehicles may crowdsense the pedestrian flow and/or the road status using on-board sensors, especially with the emergence of autonomous sensing techniques, which makes real-time data collection a practical and cost-effective method compared to traditional techniques.

In this paper, we propose to exploit the emergence of VSN and crowdsourcing to design a real-time navigation solution for vehicles in urban smart cities where automatic sensing tasks are expected to be more frequent. A central cloud server is responsible for continuously collecting the data, treating it, and recommending navigation trajectories to requesters, e.g., vehicles. The objective is to instantly guide a vehicle to reach single or multiple destinations based on real-time feedback of other road network users such as vehicles, infrastructure-based sensors, and pedestrians. Furthermore, the proposed framework is applied to the delivery vehicle problem where the objective is to find the fastest route to reach several destinations, e.g., to drop-off goods before returning to the starting position. In all cases, our framework efficiently determines the fastest route in real-time for the vehicle after providing all the scheduled destinations to the system once without predefining any order preference. The routes and destinations can be updated meanwhile according to their geographical locations and the reported traffic status. Finally, we deal with the case where the input data contains uncertain and erroneous information to determine route with minimum congestion risk. The proposed framework combines off-line and on-line information to devise navigation solutions using reported crowdsourced data.

The contributions of this paper are summarized as follows:

- Integer linear programs (ILPs) are formulated to optimally determine the fastest route for a vehicle in realistic off-line maps with single, multiple destinations, and delivery vehicles, based on the crowdsourced inputs of diverse ITS sensors such as on-board and infrastructure-based sensors, and road users. To address uncertainty and erroneous data issues, robust optimization is also implemented to avoid risky routes.
- Due to the recurrent updates, the ILPs are regularly solved according to a certain window size to determine the best progress according to the recent data. The window size corresponds to the time period during which the input crowdsourced data is assumed to be highly correlated. Two real-time navigation ILP-based algorithms are designed: the window-size algorithm (WSA) where complete routes are updated every window size and the low complexity window size algorithm (LCWSA) where part of the route is updated every window size.
- Heuristic approaches using the Dijkstra's algorithm and solving the navigation problems are also proposed to achieve reasonable results with lower computational complexity.
- Comparisons between the performances of all the proposed approaches are provided in addition to the ones of the traditional off-line shortest path algorithm.

The provided results show that joining on-line and off-line solutions can significantly improve the navigation performance in urban areas where sudden events frequently occur. The performances of proposed real-time navigation algorithms always upper-bound those of the shortest path one and allow the avoidance of blocked streets beforehand instead of identifying them in-person. At the same time, the proposed solution could pave the way towards real-time navigation for autonomous vehicles by complementing their on-board sensors with data traffic inputs shared by their peers.

### C. OUTLINE

The rest of the paper is organized as follows. The proposed system and related parameters are presented in Section II. In Section III, we introduce the decision variable and formulate the ILPs for the single-destination, multi-destinations, delivery vehicle, and uncertainty-based problems. The proposed ILP-based navigation algorithms and the heuristic approaches are presented in Section IV. The simulation results are revealed in Section V. Open challenges are discussed in Section VI and conlcusions are drawn in Section VII.

## II. SYSTEM MODEL

We consider an urban area where a complex traffic network exists. We assume that the traffic network is composed of intersections and roads. In the paper's context, each road $i$, where $i \in \{1, \ldots, N\}$ is connecting at most two intersections, where $N$ is the number of roads in this region. We denote by $\mathcal{I}$ the set including all intersections in the area of interest. Each intersection in $\mathcal{I}$ is indexed by k where $k \in \{1, \cdots, \bar{\mathcal{I}}\}$, where $\bar{\mathcal{I}}$ denotes the cardinality of the set $\mathcal{I}$. Hence, each road starts at one intersection $k$ and ends at another intersection $k'$ where $k, k' \in \mathcal{I}$. We denote the length of a road $i$ by $l_i$.

We also assume that each road $i$ is divided into $M_i$ segments. Their lengths are determined according to a certain distance threshold defined by the operator and denoted by $d_{th}$. The number of segments of road $i$ could be calculated as $M_i = \lceil l_i/d_{th} \rceil$, where $\lceil . \rceil$ denotes the ceiling function. Thus, we compute the length of each segment $j$ of road $i$ as $l_{i,j} = \frac{l_i}{M_i}$. Note that the length of the segment cannot exceed $d_{th}$, and that the lower $d_{th}$, the better accuracy for the navigator, but the higher computational complexity. We define the segments of road $i$ by $(i, j)$, $j \in \{1, \cdots, M_i\}$. Also, we define the geographical values for the middle of the segment $(i, j)$ as $\phi_{i,j}$ and $\psi_{i,j}$, representing the longitude and latitude values, respectively.

We define the set of connections linking two successive segments by $\mathcal{C}$. A connection in $\mathcal{C}$ is indexed by $q$ where $q \in \{1, \cdots, \bar{\mathcal{C}}\}$. Consequently, for each connection $q$, we denote its input set by $\mathcal{C}_q^{\text{In}}$ and its output set by $\mathcal{C}_q^{\text{Out}}$. In the example shown in Fig. 1, $\mathcal{C}_q^{\text{In}}$ and $\mathcal{C}_q^{\text{Out}}$ are given as follows:

$$\mathcal{C}_q^{\text{In}} = \{(1, 1), (2, 2)\}, \tag{1a}$$
$$\mathcal{C}_q^{\text{Out}} = \{(1, 2), (2, 1)\}. \tag{1b}$$

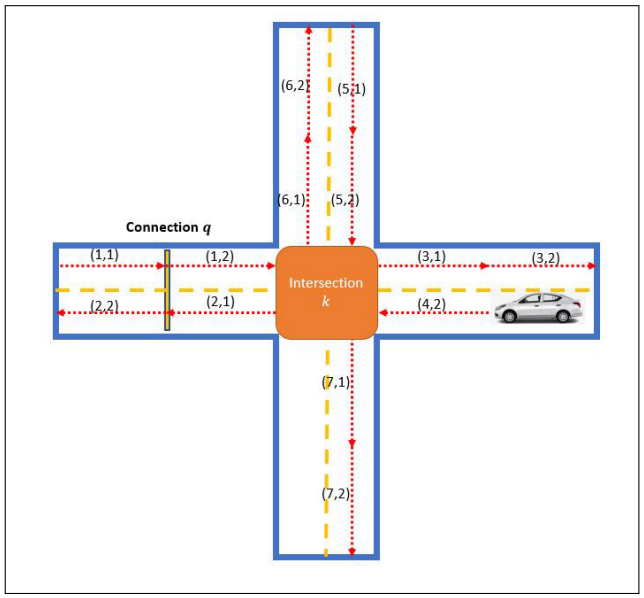



**FIGURE 1.** An example of intersection and segment connection. Seven roads are considered in this example. The connection $q$ has two inputs and two outputs where a U-turn is possible. The intersection $k$ has 3 inputs and 4 outputs.

Note that a U-turn is possible in this connection, which means that the vehicle can move in both directions. This is essentially useful when one of the segments corresponds to the starting point of the segment. It is also possible to force a vehicle to move into one direction. In this case, $\mathcal{C}_q^{\text{In}} = \{(1, 1)\}$ and $\mathcal{C}_q^{\text{Out}} = \{(1, 2)\}$.

Similarly, for each intersection $k$, we denote its input set by $\mathcal{I}_k^{\text{In}}$ and its output set by $\mathcal{I}_k^{\text{Out}}$. In the example shown in Fig. 1, $\mathcal{I}_k^{\text{In}}$ and $\mathcal{I}_k^{\text{Out}}$ are given as follows:

$$\mathcal{I}_k^{\text{In}} = \{(1, 2), (4, 2), (5, 2)\}, \tag{2a}$$

$$\mathcal{I}_k^{\text{Out}} = \{(2, 1), (3, 1), (6, 1), (7, 1)\}. \tag{2b}$$

We denote the probability that a vehicle will be stopped, e.g., by a traffic light or a stop sign, in a segment $(i, j)$ by $p_{i,j}^s$. If the vehicle moves from one segment to another while crossing a connection $q$, then $p_{i,j}^s = 0$. However, if the vehicle is crossing an intersection $k$, $p_{i,j}^s \geq 0$. For instance, if there is a stop sign in $k$, then $p_{i,j}^s = 1$. If there is a traffic light in $k$, like in Fig. 1, then $p_{i,j}^s = 0.75$. We also associate to each probability $p_{i,j}^s$ the corresponding expected waiting time as $T_{i,j}^s$. Finally, we define the starting position of a vehicle as $(S, Sg)$. For the one-destination problem, we define the destination of vehicle as $(D, Dg)$, where $(S, Sg) \neq (D, Dg)$. However, for the multi-destinations problem, we define the destinations of vehicle as $(D_n, Dg_n), n \in \{1, \cdots, \bar{\mathcal{D}}\}$ where $\bar{\mathcal{D}}$ denotes the cardinality of the set containing the destination coordinates $(D_n, Dg_n)$.

In our framework, we assume that the area of interest is managed by a central cloud server which is in charge of multiple functionalities in order to propose trajectories for vehicles based on collected real-time feedback, as shown in Fig. 2.

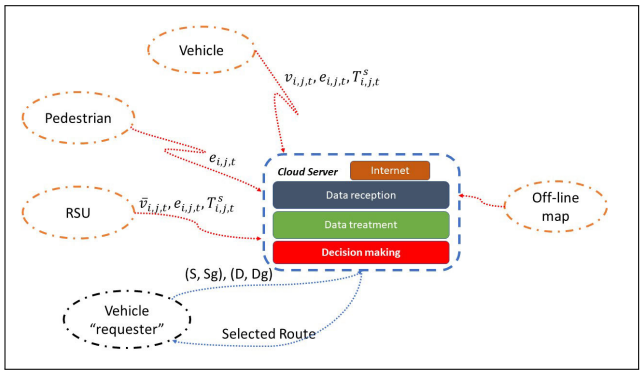



**FIGURE 2.** System architecture for the real-time navigation framework: an example of location-based query tasks in automated crowdsourcing. The cloud server will respond to the request of the vehicle ("requester") to provide it real-time routing based on the inputs of the different data sources.

The cloud server continuously collects data from different sources and accordingly, guides vehicles toward their destinations. We assume that the real-time data arrives periodically after $t$ seconds. The cloud server uses both static information, which corresponds to the complex traffic network map, and the dynamic real-time inputs provided by different traffic network sources to guide the requesters.

The real-time data include traffic speed $v_{i,j}$, accounted during the motion of the reporting vehicles, the expected waiting time at each segment $T_{i,j}^s$, accounted when the reporting vehicles are stopped, and the road segment statuses $e_{i,j}$ reported by other vehicles/devices crossing that segment $(i, j)$. The road status $e_{i,j}$ is a binary variable indicating whether a road segment is blocked or not. It is equal to 1 if it is the case. In the sequel, we add the index $t$ to the previous variables as they are updated with time according to the feedback of road users and other sensors.[1]

In a real-world scenario, since the data received by the server are from multiple vehicles, it might come at different time instants. Also, the server might receive multiple reports about the same segment from different sources. Hence, it is necessary for the server to have a data processing module, which treats and filters the data before making any decision. Due to the fact that traffic profiles in practice are usually static for short periods of time, we assume that the data is correlated over time. Therefore, we introduce the time period window size, $W_S$, where we assume that the recently received data is valid during the whole window size. The summary of the parameters is given in Table 1:

## III. PROBLEM FORMULATION

The objective of this framework is to minimize the time that the vehicle spends on roads to go from a starting position to a destination, or to go from a starting position to multiple destinations. In other words, the selected route should be

[1]More categories of data can be taken into account by the server. In this paper, for simplicity, we limit our analysis to these three categories: speed, road status, and waiting time.

**TABLE 1. List of parameters.**

| Parameter | Description |
|---|---|
| $i$ | Road id |
| $j$ | Segment id |
| $N$ | Total number of roads |
| $M_i$ | Number of segments of road $i$ |
| $\mathcal{I}$ | Intersection set |
| $k$ | Intersection id |
| $\mathcal{C}$ | Connection set |
| $q$ | id of connection |
| $\mathcal{I}_k^{\text{In}}, \mathcal{I}_k^{\text{Out}}$ | Input and output segment sets of intersection $k$ |
| $l_i$ | Length of the road $i$ |
| $d_{th}$ | Distance threshold determining road segment length |
| $l_{i,j}$ | Length of a segment $(i,j)$ |
| $\phi_{i,j}, \psi_{i,j}$ | Longitude and latitude of segment $(i,j)$ |
| $\mathcal{C}_q^{\text{In}}, \mathcal{C}_q^{\text{Out}}$ | Input and output segment sets of connection $q$ |
| $p_{i,j}^s$ | Stop probability at segment $(i,j)$ |
| $T_{i,j}^s$ | Expected stopping time at segment $(i,j)$ |
| $(S, Sg)$ | Starting position (segment $Sg$ of road $S$) |
| $(D, Dg)$ | Destination position (segment $Dg$ of road $D$) |
| $v_{i,j}$ | Average speed at segment $(i,j)$ |
| $e_{i,j}$ | Status of segment $(i,j)$ (blocked or not) |
| $f_{i,j,t}$ | Expected traveling time to cross segment $(i,j)$ during time period $t$ |
| $W_S$ | Window size to update the navigation |

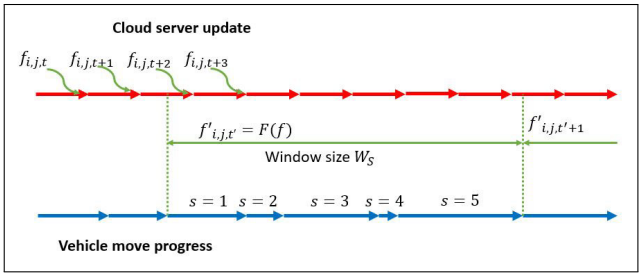



**FIGURE 3. Traffic data reception at the cloud server level. The figure clearly distinguishes between the time slots at which the data is received from crowdsources (red line) and the vehicle steps computed based on the motion of the vehicle of interest (blue line).**

preferably the least congested one with minimum stopping times and no blocked streets.

Before introducing the decision variable and formulating the problem, we need to distinguish between the following two measures: first, the fixed data collection time slot of cloud server, $t$, introduced earlier, that corresponds to the instant at which the cloud server receives updates from reporting devices and vehicles, and second, the steps taken by the vehicle of interest, indexed by the parameter $s$ and that correspond to the time needed by the vehicle to cross a segment of road. The duration of each step varies according to the vehicle moves across the segment. Fig. 3 shows the difference between the time slots and vehicle steps. The vehicle is guided based on the last updates received by the central server.

We introduce the decision variable $x_{i,j,s}$ to represent the process of selecting a route. We define it as follows:

$$x_{i,j,s} = \begin{cases} 1, & \text{if the vehicle crosses segment } (i,j) \text{ at step } s, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Let us assume that the vehicle takes at most $T$ steps to reach destination, then the set $\{x_{i,j,s}\}$ where $x_{i,j,s} = 1$, for $s \in \{1, \cdots, T\}$ indicates the entire route. In the following, we introduce the problem formulations of different routings.

### A. SINGLE-DESTINATION ROUTING

The first problem we tackled is the navigation from one starting position to one destination, in order to ensure that the selected route is reasonable, the following constraints must be considered:

**Step by step constraints:** First, we need to ensure that the vehicle is moving such that it is located in one segment during each step as long as it did not reach the destination. That is, the following conditions are added:

$$\begin{aligned} &\text{if } \sum_{\tau=1}^{s-1} x_{D,Dg,\tau} = 0, \quad \text{then, } \sum_{i,j} x_{i,j,s} = 1, \\ &\text{else } \sum_{i,j} x_{i,j,s} = 0, \quad \forall s. \end{aligned} \tag{4}$$

In order to convert those conditions to linear constraints, we use the big-M method as follows:

$$\begin{aligned} \sum_{i,j} x_{i,j,s} &\le 1 + M \sum_{\tau=1}^{s-1} x_{D,Dg,\tau}, \quad \forall s, \\ \sum_{i,j} x_{i,j,s} &\ge 1 - M \sum_{\tau=1}^{s-1} x_{D,Dg,\tau}, \quad \forall s, \end{aligned} \tag{5}$$

where $M$ is a sufficiently large positive number.

**Intersection constraints:** Second, we need to ensure that the vehicle moves in a sequential manner when crossing an intersection. For example, if the vehicle chooses to enter an intersection in the region, then in the next step, it must leave from that intersection. This is assured using the following if-else statements, which can be linearized using the big-M method too:

$$\begin{aligned} &\text{if } \sum_{\tau=1}^{s-1} x_{D,Dg,\tau} = 0, \quad \text{then,} \\ &\sum_{(i,j)\in\mathcal{I}_k^{\text{In}}} x_{i,j,s} = \sum_{(i,j)\in\mathcal{I}_k^{\text{Out}}} x_{i,j,s+1}, \quad \forall s, \forall k. \end{aligned} \tag{6}$$

For instance, the RHS and LHS of (6) can be written as follows for the intersection shown in Fig. 1:

$$\begin{aligned} \sum_{(i,j)\in\mathcal{I}_k^{\text{In}}} x_{i,j,s} &= x_{1,2,s} + x_{4,2,s} + x_{5,2,s}, \quad \forall s, \\ \sum_{(i,j)\in\mathcal{I}_k^{\text{Out}}} x_{i,j,s'} &= x_{2,1,s'} + x_{3,1,s'} + x_{6,1,s'} + x_{7,1,s'}, \end{aligned} \tag{7}$$

where $s' = s + 1$.

**Connection constraints:** Third, for each segment connection, the vehicle moves in a sequential manner in a way

similar to one of the intersections. We then have:

$$\text{if } \sum_{\tau=1}^{s-1} x_{D,Dg,\tau} = 0, \quad \text{then,}$$
$$\sum_{(i,j)\in\mathcal{C}_q^{\text{In}}} x_{i,j,s} = \sum_{(i,j)\in\mathcal{C}_q^{\text{Out}}} x_{i,j,s+1}, \quad \forall s, \forall q. \tag{8}$$

According to Fig. 1, we obtain the following constraint for the given connection:

$$\sum_{(i,j)\in\mathcal{C}_q^{\text{In}}} x_{i,j,s} = x_{1,1,s} + x_{2,2,s}, \quad \forall s,$$
$$\sum_{(i,j)\in\mathcal{C}_q^{\text{Out}}} x_{i,j,s'} = x_{2,1,s'} + x_{1,2,s'}, \quad s' = s+1. \tag{9}$$

**Initialization constraint:** Last, we initialize the position of the vehicle as follows:

$$x_{S,Sg,1} = 1. \tag{10}$$

**Utility Function and Optimization Problem:** We define a metric corresponding to the expected time spent on each segment at time slot $t$, as shown in the follows:

$$f_{i,j,t} = \frac{l_{i,j}}{v_{i,j,t}} + T^s_{i,j,t} p^s_{i,j} + E_0 e_{i,j,t}, \tag{11}$$

where the first term in (11) is the time spent by a vehicle in motion while crossing the segment (i,j). The second term in (11) represents the delay due to a stop sign or a traffic light, and the third term is the time due to an unexpected event blocking the street during the time period $t$. The parameter $E_0$ is a penalty factor that is imposed when segment $(i, j)$ is blocked ($E_0 \to +\infty$). Based on the metric (11), we define the fitness function at each step $s$ as follows:

$$f_s(x_{i,j,s}) = \sum_{i,j} f'_{i,j,t'} x_{i,j,s}, \tag{12}$$

where $t'$ represents the last measure of the metric before the execution of step $s$ as illustrated in Fig. 3 and $f'$ is the output of the data process made by the server, that we denote by the function $F$ where $f'_{i,j,t'} = F(f_{i,j,t})$. The function $F$ incorporates the filtering and the treatment of the reported data. We assume that the fitness metric $f'_{i,j,t'}$ remains valid for a certain number of time slots that is denoted by $W_S$. The choice of $W_S$ is decided by the cloud server according to the data correlation level. The impact of $W_S$ is investigated in Section V.

On the other hand, we define the geographical distance as $g_{i,j}$, to represent the distance from the segment $(i, j)$ to the destination $(D, Dg)$, calculated by the longitude and latitude values $\phi_{i,j}$ and $\psi_{i,j}$ defined earlier. Then, we define the distance function at each step $s$ as follows:

$$g_s(x_{i,j,s}) = \sum_{i,j} g_{i,j} x_{i,j,s}. \tag{13}$$

By minimizing the fitness function $f_s$, the cloud server will find the fastest segment at each time slot, but the direction selected for the vehicle could be erroneous. By minimizing the distance function $g_s$, the cloud server will find the right direction, but the selected route will be slow. In order to select the fastest route for the vehicle with the correct direction, we define the utility function at each step $s$ as a weighted sum of both functions as follows:

$$\mathcal{U}_s(x_{i,j,s}) = \omega f_s + (1-\omega) g_s. \tag{14}$$

These functions are weighted by a Pareto parameter $\omega$ ($0 < \omega < 1$). When $\omega \to 1$, we are dealing with the fitness function given in (12). This corresponds to an on-line navigation strategy that selects the fastest route given the traffic status. When $\omega \to 0$, we deal with the fitness function given in (13), which corresponds to an off-line navigation strategy finding the shortest path regardless of the traffic status. Other values of $\omega$ constitute a trade-off between these two extremes. Finally, the optimization problem is formulated as follows:

$$\text{(P0): } \underset{x_{i,j,s}\in\{0,1\}}{\text{minimize}} \sum_{s=1}^{T} \mathcal{U}_s(x_{i,j,s})$$
$$\text{subject to: (4), (6), (8), and (10).} \tag{15}$$

In practice, the Pareto parameter $\omega$ could be tricky to choose for best results, some $\omega$ returns high performance for one scenario and low performance for another scenario. In this case, we formulate a new problem that does not need to consider the distance function $g_s$. To this end, we add the following extra constraint:

**Arrival condition constraint:** This constraint forces the vehicle to reach the destination as follows:

$$\sum_{\tau=1}^{T} x_{D,Dg,\tau} = 1. \tag{16}$$

Then we formulate the problem as:

$$\text{(P1): } \underset{x_{i,j,s}\in\{0,1\}}{\text{minimize}} \sum_{s=1}^{T} f_s(x_{i,j,s})$$
$$\text{subject to: (4), (6), (8), (10), and (16).} \tag{17}$$

### B. MULTI-DESTINATIONS ROUTING

We also investigate the problem with multiple destinations where the vehicle leaves from one starting position and sequentially visits several destinations. The constraints are slightly different from what is given earlier.

In the multiple-destinations problem, minimizing the distance function $g_s$ is no longer convenient because there are many distances between the vehicle and destinations to minimize which is infeasible in the multi-destinations case, especially that we are not providing a specific ranking for the destinations. Based on the traffic status, the cloud server will determine which destination to be visited first.

**Arrival condition constraint:** Hence, we introduce the following constraints of arrival conditions which guarantee

that the vehicle reaches all the destinations:

$$\sum_{n=1}^{\bar{\mathcal{D}}}\sum_{\tau=1}^{T} x_{D_n,Dg_n,\tau} = \bar{\mathcal{D}}. \tag{18}$$

**Step by step constraints:** Then, we need to ensure that the vehicle is moving such that it is located in one segment during each step as long as it did not reach all the destinations. The following constraints are added:

$$\begin{aligned} &\text{if } \sum_{n=1}^{\bar{\mathcal{D}}}\sum_{\tau=1}^{s-1} x_{D_n,Dg_n,\tau} < \bar{\mathcal{D}}, \quad \text{then, } \sum_{i,j} x_{i,j,s} = 1, \\ &\text{else } \sum_{i,j} x_{i,j,s} = 0, \quad \forall s. \end{aligned} \tag{19}$$

The rest constraints, namely **Intersection constraints**, **Connection constraints** and **Initialization constraints** remain the same as (6), (8), and (10).

**Optimization Problem:** Hence, the multi-destination routing problem is formulated as follows:

$$\begin{aligned} &\text{(P2): } \underset{x_{i,j,s}\in\{0,1\}}{\text{minimize}} \sum_{s=1}^{T} f_s(x_{i,j,s}) \\ &\text{subject to: (6), (8), (10), (18), and (19).} \end{aligned} \tag{20}$$

### C. DELIVERY VEHICLE PROBLEM

Moreover, we formulate another problem to solve the puzzle of the delivery vehicle problem where the objective is to find the optimum route for vehicle visiting several locations and then going back given the crowdsourced data. For example, this can correspond to the case of a truck delivering goods to many stores around the city. The constraints are slightly different from the ones of the previous multi-destinations problem as follows:

**Return constraints:** The vehicle must return to its start position after reaching all the destinations, hence, the following constraints are added:

$$\begin{aligned} &\text{if } \sum_{n=1}^{\bar{\mathcal{D}}}\sum_{\tau=1}^{s} x_{D_n,Dg_n,\tau} = \bar{\mathcal{D}}, \\ &\text{then, } \sum_{\tau=s+1}^{T} x_{S,S_g,\tau} = 1, \quad \forall s. \end{aligned} \tag{21}$$

**Step by step constraints:** Then, we need to ensure that the vehicle is moving before it reaches back to the starting position. That is, the following constraints are added:

$$\begin{aligned} &\text{if } \sum_{\tau=2}^{s-1} x_{S,Sg,\tau} = 0, \quad \text{then, } \sum_{i,j} x_{i,j,s} = 1, \\ &\text{else } \sum_{i,j} x_{i,j,s} = 0, \quad \forall s. \end{aligned} \tag{22}$$

rest of the constrains remain the same as (6), (8), and (10)
**Optimization Problem:** We formulate the delivery vehicle routing problem as follows:

$$\begin{aligned} &\text{(P3): } \underset{x_{i,j,s}\in\{0,1\}}{\text{minimize}} \sum_{s=1}^{T} f_s(x_{i,j,s}) \\ &\text{subject to: (6), (8), (10), (21), and (22).} \end{aligned} \tag{23}$$

The problems (P0), (P1), (P2), and (P3) are ILPs that can be solved optimally using off-the-shelf software which implements algorithms such as the branch and bound technique to determine optimal routes for the vehicles. This assumes the perfect knowledge of the road network status from the departure of the vehicle until its arrival to destination, which is not true in practice. Therefore, in the Section IV, we proceed with iterative ILP-based approaches where routes are modified (if needed) whenever the server receives new traffic status updates. Moreover, we propose in the following to optimize the navigation of vehicles while considering uncertain crowdsourced feedback.

### D. ROUTING UNDER UNCERTAINTY CONSIDERATION

In practice, accurate knowledge about the traffic status cannot be perfectly measured especially when it is crowdsourced in real-time from different sources. Indeed, data received by cloud server might be erroneous, missed, or contaminated. For instance, there is difference between the actual speed and the one reported to the server. In order to take this uncertainty effect into consideration, we model the input data that will be treated by the cloud server through introducing an uncertainty parameter $\sigma_{i,j,t}$ modeled as a zero-mean distribution and standard-deviation denoted by $\theta$. We obtain:

$$\underbrace{\tilde{f}'_{i,j,t'}}_{\text{Estimated value}} = \underbrace{f'_{i,j,t'}}_{\text{Submitted value}} + \underbrace{\sigma_{i,j,t}}_{\text{Uncertainty}}. \tag{24a}$$

In order to linearize the uncertainty property [28], we add the following constraints:

$$\begin{aligned} \tilde{f}'_{i,j,t'} &= f'_{i,j,t'} + f_1 - f_2, \quad i,j \in ST \\ f_1 + f_2 &= \sigma_{i,j,t} \\ 0 \le f_1, f_2 &\le \sigma_{i,j,t} \end{aligned} \tag{24b}$$

where $f_1$ and $f_2$ are lag inputs of the model, representing the uncertainty, and $ST$ represents the group of streets that have variation between their estimated values and the submitted values. In this case, the utility function becomes:

$$\tilde{f}_s(x_{i,j,s}) = \sum_{i,j} \tilde{f}'_{i,j,t'} x_{i,j,s}, \tag{25}$$

$$\mathcal{U}_s(x_{i,j,s}) = \tilde{f}_s. \tag{26}$$

**Optimization Problem:** Then, the optimization problem for single destination under uncertainty is given as follows:

$$\begin{aligned} &\text{(P4): } \underset{x_{i,j,s}\in\{0,1\}}{\text{minimize}} \sum_{s=1}^{T} \mathcal{U}_s(x_{i,j,s}) \\ &\text{subject to: (4), (6), (8), (10), (16), and (24}b\text{).} \end{aligned} \tag{27}$$

We apply a robust optimization approach to sought against uncertainty. Hence, the obtained route is expected to be the fastest and the least risky one with minimum uncertainty level. Similar approach can be applied to deal with the uncertainty in the multi-destinations and the delivery-vehicle problems.

## IV. PROPOSED REAL-TIME NAVIGATION ALGORITHMS

In this section, we first introduce two algorithms solving the ILP problems formulated earlier while taking the real-time aspect of the framework. The first algorithm, defined as the Window Size Algorithm (WSA), finds, for each window size, a full route for the vehicle from the starting point to the destination. The route is updated after the expiration of $W_S$ according to the recent data reported by the different sources. To reduce complexity, we propose a second algorithm that we call the Low Complexity Window Size Algorithm (LCWSA) where the ILP is solved in a much smaller region defined by the window size. Then, we propose two heuristic approaches for the Window Size Algorithm (WSA) which run much faster while providing reasonable results compared to the ILP-based solutions as it will be shown in Section V.

### A. WINDOW-SIZE ALGORITHM (WSA)

The proposed approach aims to find the fastest route towards the destination based on the latest data received by the server. The latter is assumed to remain valid for a certain duration of time that we defined earlier as the window size $W_S$. Hence, the algorithm will find a route based on the metric $f'_{i,j,t'}$. In other words, for every $W_S$, the cloud server selects the entire route for the vehicle. After $W_S$, a new route is selected based on the updated metrics. The starting point is then updated to the latest location of the vehicle.

In the WSA algorithm, we do not need to consider the distance function $g_s$ since every $W_S$, a full route will be selected for the vehicle. Hence, in the proposed WSA for single destination, the vehicle sends its initial location and destination to the cloud server at first, then at every $W_S$, the cloud server obtains the last updated metric $f'_{i,j,t'}$, and initializes the decision variable $\mathbf{X} = \{x_{i,j,s}\}$. Afterwards, the cloud server solves the optimization problem (P0) for the $\mathbf{X}$ and selects the best route for the vehicle at this iteration. After $W_S$ unit of time, the new iteration begins and a new route for the vehicle is determined. The algorithm converges when the vehicle arrives at destination, or it stops when the total steps $T$ is reached. The proposed WSA navigation algorithm for the one-destination problem is given in Algorithm 1.

For multiple-destinations and delivery vehicle problems, the algorithms are similar, except the fact that the cloud server solves the optimization problem (P1) and (P2) instead. For the routing with uncertainty consideration the $f_{i,j,t}$ are replaced by $\tilde{f}_{i,j,t}$ ones.

**Algorithm 1** Window Size Algorithm

1: Inputs $= \{(S, Sg), (D, Dg)\}$.
2: $\mathbf{t} = 0$; $s = 0$; $x_{loc} = (S, Sg)$.
3: **while** vehicle does not reach the destination **do**
4: Obtain the last update $f'_{i,j,t'}$.
5: Initialize the decision variable $\mathbf{X} = \{x_{i,j,s}\}$ where $(i, j) \in \{1, \cdots, N\} \times \{1, \cdots, M_i\}$, $s \in \{1, \cdots, T\}$.
6: Solve the optimization problem (P0) for $\mathbf{X}$.
7: Determine the new location of the vehicle $x_{loc}$ within $W_S$.
8: **end while**


**FIGURE 4.** LCWSA strategy: The cloud server will provide a part of the route within a circle every $W_S$ period.

### B. LOW COMPLEXITY WINDOW SIZE ALGORITHM (LCWSA)

Route selection in a large city, that contains thousands of roads with a large number of segments, could be very computationally hungry. In the previous proposed approach, WSA runs the optimization problem over the entire region, which leads to high computational complexity. Hence, we propose LCWSA to reduce the problem complexity while maintaining the real-time navigation property. This is done by shrinking the entire region into multiple circular areas where, at each iteration, the problem (P) is solved. The circles $C(x_{loc}, R_E)$ are centered at $x_{loc}$, i.e., the location of the vehicle at the end of each iteration, and have a radius $R_E$, which corresponds to the maximum distance that can be traveled by a vehicle during $W_S$. Recall that, for every $W_S$, the vehicle moves several steps according to the cloud server instructions. We define the maximum number of expected steps taken by the vehicle during $W_S$ as the batch step of the algorithm, $B_S$. Thus, $T$ in (P) is replaced by a lower value, i.e. $B_S$. A simple example is shown in Fig. 4.

In the proposed LCWSA, the vehicle sends its location and destination to the cloud server. At each iteration, it moves forward by $B_S$ steps based on the latest feedback $f'_{i,j,t'}$. Note it may take more than $W_S$ unit of time for a vehicle to cross the selected route; in this case, the cloud server starts a new iteration after $W_S$ to update the route. In order to select the next $B_S$ segments for the vehicle, the cloud server determines the new circular region defined $x_{loc}$ and $R_E$. Then, it solves

**Algorithm 2** Low Complexity Window Size Algorithm

Inputs $= \{(S, Sg), (D, Dg)\}$.
$\mathbf{t} = 0$; $s = 0$; $x_{loc} = (S, Sg)$.
**while** vehicle does not reach the destination **do**
  Obtain the last update $f'_{i,j,t'}$.
  Set $B_S$ based on $W_S$.
  Obtain the shrinking region $(R_E, x_{loc})$.
  Initialize the decision variable $\mathbf{X} = \{x_{i,j,s}\}$ where $(i, j) \in \mathrm{C}(x_{loc}, R_E)$, $s \in \{1, \cdots, B_S\}$.
  Solve the optimization problem (P) for $\mathbf{X}$.
  Determine the new location of the vehicle $x_{loc}$ within $W_S$.
**end while**

the optimization problem (P) for that region. The proposed algorithm is given in Algorithm 2. Note that LCWSA is not suitable for multiple destinations problem.

### C. GRAPH-BASED HEURISTIC APPROACHES

In the previous section, the problems are solved using ILP, which returns the optimum solution but still requires computational complexity. When the map is large and there are multiple destinations, it takes a long time for the ILP-based approaches to converge. Therefore, in this section, we introduce a heuristic approach that solving solves the navigation problems for the different scenarios with closely good results and fast convergence time using the Recurrent Dijkstra's algorithm [29]. Other graph-based shortest path algorithms such as Bellman-Ford algorithm and Johnson's algorithm can be adapted too [30].

**One-destination Problem:**

In the off-line map, we convert the intersections $\mathcal{I}$ and connections $\mathcal{C}$ to the vertices of a graph while the segments are modeled as the edges connecting the vertices. The expected time that the vehicle spends on one segment represents the weight of the associated edge. The weights of all edges are continuously updated, i.e., every $W_S$. Hence, we employ the Dijkstra's algorithm to determine the fastest route from a starting node to the destination one given the current values of the edges' weights. After $W_S$, the Dijkstra's algorithm is re-executed given the new location of the vehicle and a new route is obtained. The algorithm is given in Algorithm 3.

**Multi-destination problem:**

For the multi-destinations problem, an order must be chosen. For example, the vehicle could go to the closest destination first, then go to the next closest one, and so on. Based on this rule, and hence, the one-destination algorithm can be applied to each destination following the chosen order. The algorithm providing real-time navigation while following a pre-chosen order of destinations is called as Ordered Destinations Navigation Algorithm (ODNA) and is summarized in Algorithm 4.

On the other hand, setting the order of destinations based on the shortest distance does not necessary lead to the minimum expected time spent due to the traffic status variation. Therefore, we design the another algorithm where the order of the destinations is updated in real-time according to the expected navigation time measured using the crowdsourced data. Hence, the order of destinations

**Algorithm 3** Modified Dijkstra's Algorithm based on Crowdsourced Data

Inputs $= \{(S, Sg), (D, Dg)\}$.
$\mathbf{t} = 0$; $s = 0$; $x_{loc} = (S, Sg)$.
**while** vehicle does not reach the destination **do**
  Obtain the last update $f'_{i,j,t'}$, updated as the weights of edges in the graph.
  Run Dijkstra's algorithm to find the fastest route from $x_{loc}$ to $(D, Dg)$.
  Determine the new location of the vehicle $x_{loc}$ within $W_S$.
  Update actual time spent $t$.
**end while**

**Algorithm 4** Ordered Destinations Navigation Algorithm

Inputs $= \{(S, Sg), \{D_n, Dg_n\}\}$.
Order $\{D_n, Dg_n\}$ based on distance.
$\mathbf{t} = 0$; $s = 0$; $x_{loc} = (S, Sg)$.
**while** vehicle does not reach all destinations **do**
  Update the destinations (remove destinations that been reached already).
  Obtain the last update $f'_{i,j,t'}$, updated edges of the graph.
  Compute the cloest route from $x_{loc}$ to the first order destination with Dijkstra's algorithm.
  Determine the new location of the vehicle $x_{loc}$ within $W_S$.
  Update $t$.
**end while**

**Algorithm 5** Updated Destinations Navigation Algorithm

Inputs $= \{(S, Sg), \{D_n, Dg_n\}\}$.
Obtain the last update $f'_{i,j,t'}$.
$\mathbf{t} = 0$; $s = 0$; $x_{loc} = (S, Sg)$.
**while** vehicle does not reach all destinations **do**
  Update the destinations (remove destinations that have been reached already).
  Obtain the last update $f'_{i,j,t'}$, updated edges of the graph.
  Compute the expected time spent on remaining destinations based on $f'_{i,j,t'}$; find the fastest destination $(D_j, Dg_j)$.
  Get the fastest route from $x_{loc}$ to that destination $(D_j, Dg_j)$ with modified Dijkstra's algorithm.
  Determine the new location of the vehicle $x_{loc}$ within $W_S$.
  Update $t$.
**end while**

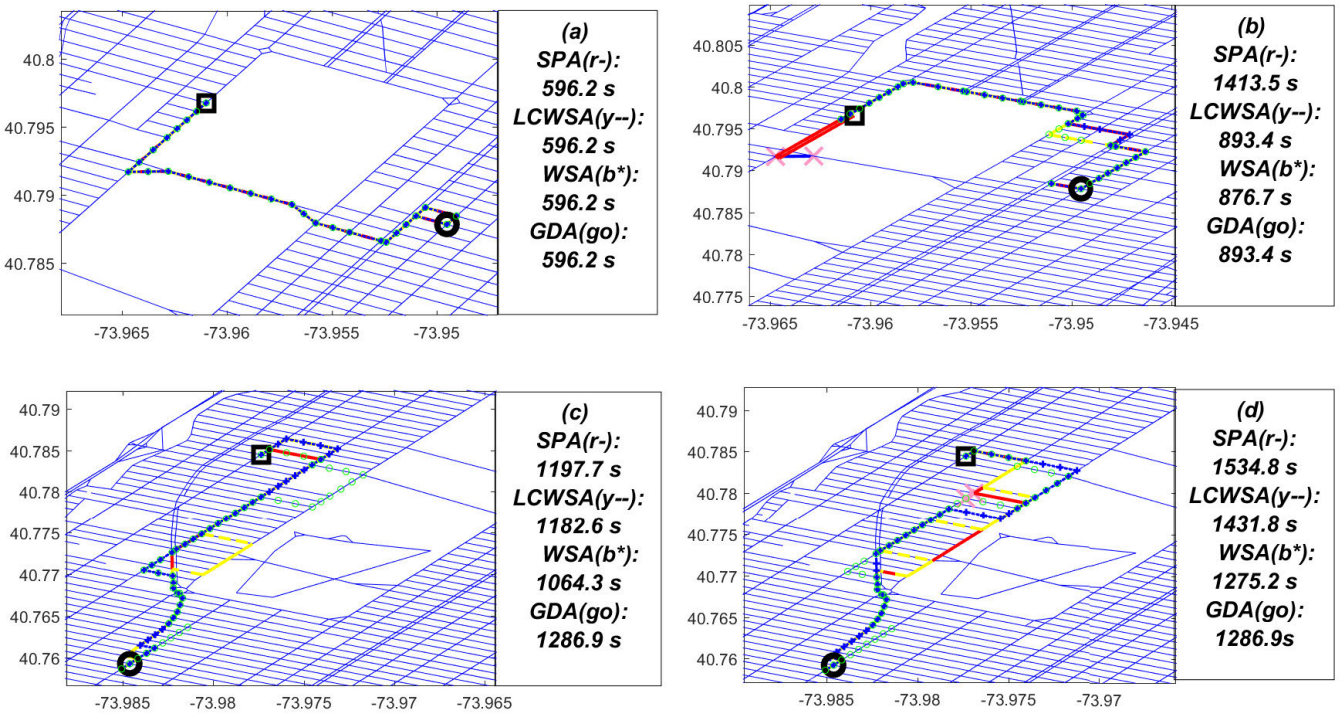



**FIGURE 5.** Route Selection with the proposed algorithms and SPA for the single-destination problem. Square: ($S$, $Sg$) and circle: ($D$, $Dg$). In the legend, 'r', 'y', 'b', and 'g' represent red, yellow, blue, and green, respectively. The gray crosses represent blocked roads. The proposed algorithms effectively avoid the blocked roads and select the less congestion routes towards the destination.

can be updated every $W_S$ according to the traffic status and the vehicle progress. This algorithm is summarized in Algorithm 5 and named as Updated Destinations Navigation Algorithm (UDNA).

**Delivery-vehicle problem:** For the delivery-vehicle problem, the heuristic approaches are similar. After reaching all the destinations, the vehicle heads to the start position ($S$, $Sg$).

**Uncertainty:** In order to take the uncertainty into consideration for our heuristic algorithms, we assign an estimated weight for each road. This weight is calculated based on the received data from the cloud server while considering its uncertainty level. The worst-case scenario is taken into account when running the Dijsktra's algorithm to find the fastest route. Hence, the roads that are highly expected to be congested will be avoided. The graph-based algorithms developed earlier for the single-destination, multi-destinations, and delivery-vehicle problems can be applied.

## V. SIMULATION AND RESULTS

In this section, we start by presenting the simulation environment and thenn evaluate the performance of the proposed WSA, LCWSA, and graph-based heuristic algorithms for the different problems. The case with uncertainty is also investigated.

### A. SIMULATION PARAMETERS

We assign the Pareto parameter $\omega$ as 0.5 for LCWSA. We compare their performances with the ones of the standard Shortest Path Algorithm (SPA). In our simulations, we consider the area of Manhattan, New York. The parameters of the off-line map are obtained from Open Street Map [31]. In total, there are 9683 roads and 4469 intersections in the area of interest. We choose the threshold of segments' length $d_{th} = 100$ meters.

Apart from off-line map information, the proposed navigation algorithms require the real-time traffic information data, which include $v_{i,j,t}$, $e_{i,j,t}$, and $T^s_{i,j,t}$ provided by road users and infrastructure-based sensors. We assume that the cloud server is updated every 10 seconds ($t = 10\,s$) and the average expected stop time per intersection is 10 seconds $T^s_{i,j,t} = 10\,s$ for simplicity. Due to the non-availability of realistic data and in order to model a comprehensive traffic situation happening in urban cities, we consider that the traffic speed data $v_{i,j,t}$ reported over the segments follow a normal distribution with a mean $\frac{2}{3}Sl_{i,j}$ and a standard deviation $= 2$, where $Sl_{i,j}$ is the maximum regulated speed of segment $(i, j)$.

### B. SINGLE-DESTINATION PROBLEM

Under this scenario, we select two different possibilities of $(S, Sg)$ and $(D, Dg)$ and compare the routes obtained by WSA, LCWSA, SPA, and the graph-based Dijkstra's algorithm (GDA). The selected routes are shown in Fig. 5.

In Fig. 5(a), the four algorithms gives exactly the same route. This is due to the fact that the real-time route exactly correspond to the shortest route. Indeed, since the distance to the destination is considered in the real-time navigation

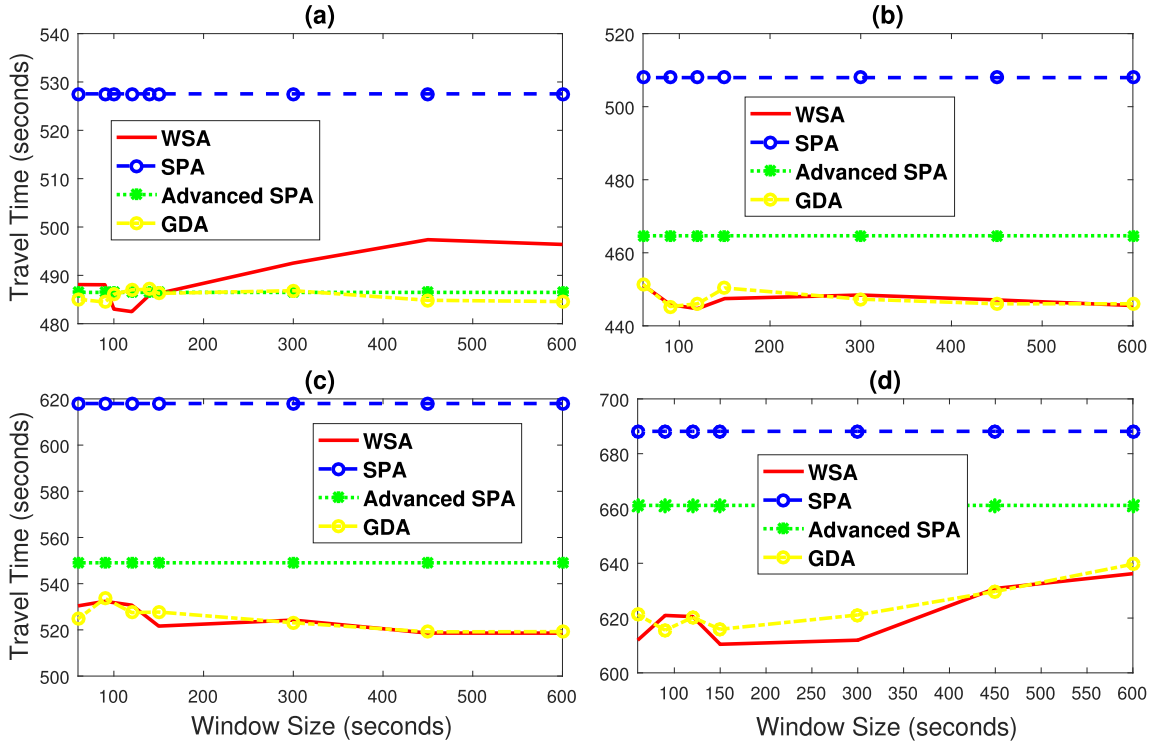
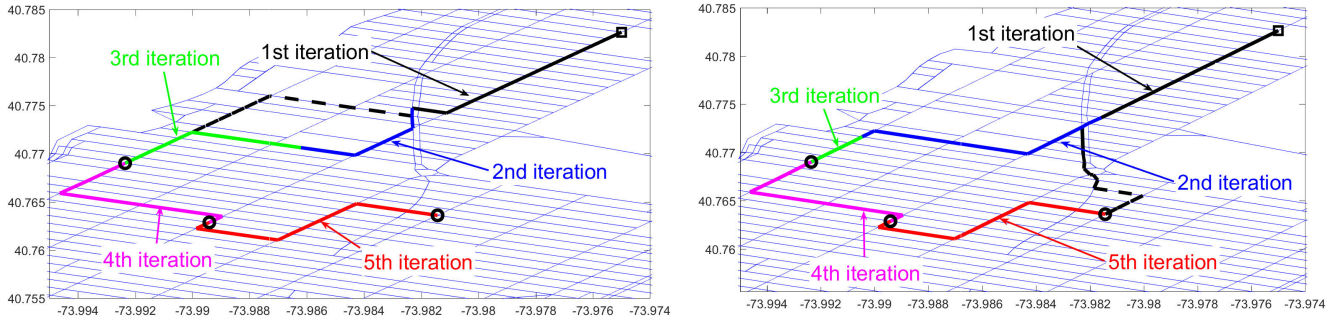



**FIGURE 6.** Average travel time versus the window size $W_S$ for randomly chosen starting points. (a) Random traffic pattern, (b) stable traffic pattern, (c) decreasing traffic trend, and (d) cyclic traffic pattern.

algorithms, similar routes can be obtained. However, in Fig. 5(b), we intentionally blocked road 8359. In other words, we set $e_{8359,1,t} = 1 \quad \forall t$. This road corresponds to an important intersection between the starting point and the destination (see the blue cross in Fig. 5(b)). In that case, WSA, LCWSA, and GDA avoid the blocked road in advance thanks to the real-time updates and hence, save more than 30% of the time compared to SPA. Indeed, the SPA, as in practice for not reported blocked roads, will choose the fastest route assuming normal road operation till it reaches the blocked road. A re-computation is then made to adjust the route. This usually leads to a high delay. In Fig. 5(c) and Fig. 5(d), we show the selected routes with another choice of $(S, Sg)$ and $(D, Dg)$. A new blocked road, $(i, j) = (3110, 1)$ is also made in Fig. 5(d). The performance of WSA is always the best among the four algorithms, it saves more than 20% travel time with the blocked road and approximately 14% travel time without the blocked road compared to SPA. Indeed, WSA efficiently exploits the real-time feedback to find the fastest route unlike the LCWSA, which performs a higher number of route direction changes (oscillation) as shown in Fig. 5(d) since it looks for the solution with the region defined by the window size. Nevertheless, the simulations show that the performance of WSA and LCWSA are always upper-bounding those of the SPA. The GDA has fairly good results in Fig. 5(a), (b), (d). But lower performance in Fig. 5(c) since the Dijkstra's algorithm is stuck at local optimum results. Another reason is that $W_S$ value is not appropriately selected for this scenario. Nevertheless, in most case, it achieves close performance to WSA. The following results will show this fact.

To prove the advantages of the proposed algorithms in general cases, and better evaluating the performance of the GDA, we depict, in Fig. 6, the average travel time versus the window size ($W_S$), for 100 routes obtained by four different algorithms: i) WSA, GDA, SPA, and advanced-SPA. The latter is similar to the SPA except that the blocked roads are assumed to be known (i.e., reported by human inputs like in WAZE). In this Monte Carlo analysis, we fix the destination and randomly change the starting position (100 times) then, we compute the average travel time for these 100 trips using four different scenarios.

The first scenario plotted in Fig. 6(a), corresponds to the case described earlier and used in Fig. 5 (random speed pattern). The second scenario plotted in Fig. 6(b), considers stable traffic pattern where the traffic speed $v_{i,j,t}$ remains stable around $\frac{2}{3}Sl_{i,j}$ through the whole period and in all roads with minor white noise variation. The third scenario, depicted in Fig. 6(c), represents the decreasing traffic trend, where the speed remains stable in the first period of time then, decreases slowly in the second period of time, and finally, remains stable again at the end. The fourth scenario represents a cyclic traffic pattern where the speed per road is reattained after 20 minutes. Except for scenario 1, there is a time correlation of the speed. The results, shown in Fig. 6, indicate that the proposed WSA and GDA outperforms traditional




(a) ODNA. (b) UDNA.

**FIGURE 7.** An example of real-time navigation usingGreg ODNA and UDNA algorithms with three destinations. (a) ODNA. (b) UDNA.

state-of-the-art algorithms except for the first scenario (Fig. 6(a)) where the proposed algorithm performance is worse than Advanced-SPA for some selected window sizes. The reason is that, in the first scenario, we assume there is no time correlation of the speed, which contradicts the assumption on which WSA and GDA are based. In WSA and GDA, we assume that the latest updated values provided by road users remain valid for $W_S$ time. However, the data is varying unexpectedly. Hence, the choice of $W_S$ is primordial in designing real-time navigation algorithms to balance between complexity and efficiency. On one hand, for high $W_S$ values, fewer iterations are needed to find a full route however, the obtained results are not adapted to rapid traffic fluctuations. On the other hand, for low $W_S$ values, the frequency of updating route is high and may cause unsatisfied user experience and may lead to poor performance. Therefore, a good choice of $W_S$ is important to find a tradeoff between complexity, efficiency, and user experience. From our empirical results, setting $W_S$ around 120 seconds returns the best results. Similar remarks can be noticed with the cyclic pattern (Fig. 6(d)). When strong time correlation exists, the proposed real-time algorithms always outperform the offline algorithms. We also notice that the performances of GDA are near the ones of WSA, sometimes it even outperforms WSA. The reason is that both algorithms are based on the $W_S$ predicting assumptions and this assumption does not reflect the traffic speed change precisely, which make the performance of WSA inappropriate for some $W_S$. Note that, the optimum solution from WSA is better than the GDA.

### C. MULTIPLE-DESTINATIONS PROBLEM

In this section, we depict the route updating process when the vehicle is heading to multiple destinations with both ODNA (presented in Algorithm 4) and UDNA (presented in Algorithm 5). We then compare the performance of ODNA and UDNA with the optimum WSA solution.

The example of route updating of ODNA at every iteration is shown in Fig. 7(a). The 'square' represents starting position and 'circles' represent destinations. Different colored solid lines represent the route selection per iteration of the vehicle at each $W_S$, and the dashed lines represent the previous selected route at certain iteration and that being updated afterwards. For example, we notice that the selected route at the first iteration (black line) changes at the second iteration (blue line), due to the changes of traffic situation. The remaining route is kept the same. Note that, the destinations that are visited by the vehicle maintain the same order initially defined based on their distances from the starting position, while, in the UDNA, the order of the destinations may change according to current location of the vehicle and the traffic situation, which is shown in Fig. 7(b).

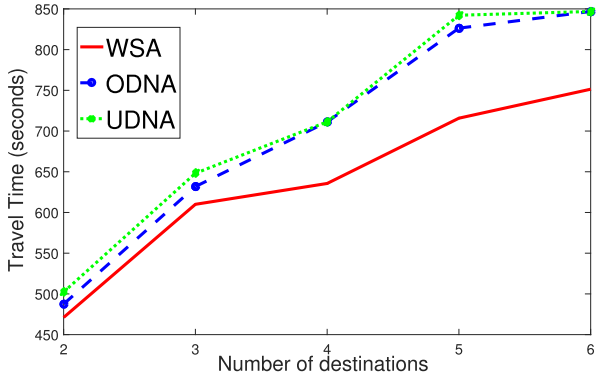



**FIGURE 8.** Average travel time for the WSA, ODNA, and UDNA with different number of destinations.

In Fig. 8, we compare the performance of both ODNA and UDNA to the one of the WSA.

Note that, we execute a Monte Carlo simulation with 50 different groups of destinations and record their average travel time. We notice that when the number of destinations increases, the average travel time between WSA and UDNA, ODNA increases. The reason is that, for a high number of destinations, the destinations order affects more the performance of the navigation algorithms ODNA and UDNA. Indeed, unlike WSA for multi-destinations, ODNA and UDNA determine the destinations order using greedy solutions(based on the shortest path for ODNA and iteratively while navigating for UDNA). On the other hand, we notice that the performances of ODNA and UDNA are very close to each other while ODNA is slightly better than UDNA in our simulations.

### D. DELIVERY-VEHICLE PROBLEM

In Fig. 9, we show an example of the delivery vehicle route selection with WSA, ODNA, and UDNA algorithms. In the WSA (red path), the vehicle reaches left destination first, then

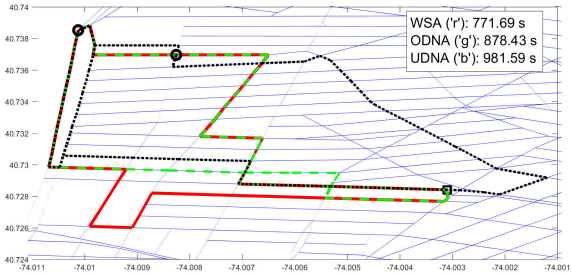



**FIGURE 9.** Different routes selected for delivery-vehicle problem with WSA, ODNA, and UDNA for two stops. 'Red', 'green' and 'black' lines represent the routes selected by WSA, ODNA, and UDNA, respectively.

**TABLE 2.** Total running time in seconds of the optimal and heuristic approaches for multi-destinations problem.

| | $\bar{D}=2$ | $\bar{D}=3$ | $\bar{D}=4$ | $\bar{D}=5$ | $\bar{D}=6$ |
|---|---|---|---|---|---|
| WSA | 471.100 | 610.027 | 635.630 | 715.792 | 751.385 |
| ODNA | 0.317 | 0.776 | 0.929 | 0.961 | 1.164 |
| UDNA | 1.375 | 1.158 | 1.927 | 2.388 | 3.530 |

it heads to the right destination. With ODNA (green path), the vehicle reaches the right destination first so that result in the delay of total traveling time. For the 'UDNA' algorithm (black path), although it heads to the left destination first, the selected route is not the optimum one compared to the WSA. In this example, WSA saves 12% of the time compared to ODNA and 20% of the time compared to UDNA.

### E. COMPUTATION COMPLEXITY

Although the travel time of WSA is better than ODNA and UDNA, the running time of heuristic approaches is much faster than the WSA, the running times of WSA, ODNA, and UDNA to converge to the results illustrated in Fig. 8 are shown in Table 2.

The average solving time of WSA is between 300 and 4000 seconds and it increases dramatically when the number of destinations increases, while the solving time of heuristic approaches is between 1 to 4 seconds. In short, our heuristic approaches are applicable in practice and we recommend the ODNA in this case.

In details, for the WSA, we are iteratively solving a large-scale ILPs with more than 700 constraints and thousands of decision variables. For the GDA, the complexity is $O(E + VlogV)$ where $E$ is the number of edages ($E = 11381$) and $V$ is the number of vertices ($V = 8758$). Note that all tests were performed on a laptop machine featuring an Intel(R) Core(TM) i7 CPU and running Windows 8:1. The clock of the machine is set to 2:66 GHz with an 8 GB memory.

### F. NAVIGATION UNDER UNCERTAINTY FEEDBACK

We have investigated the performance of the real-time navigation using the estimated traffic status measured through the collected data. In this section, we evaluate the performances of the WSA algorithm with the same simulation parameters but while considering different levels of uncertainty. The navigation problem is then solved using robust optimization at every iteration of the WSA. In Fig. 10, we depict the selected routes given zero, low, and high uncertainty levels. The red line represents the perfect knowledge where no uncertainty

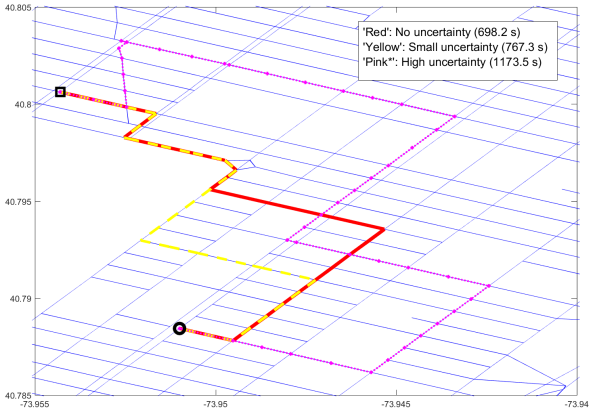



**FIGURE 10.** Time spent by a vehicle for different uncertainty levels using WSA and robust optimization.

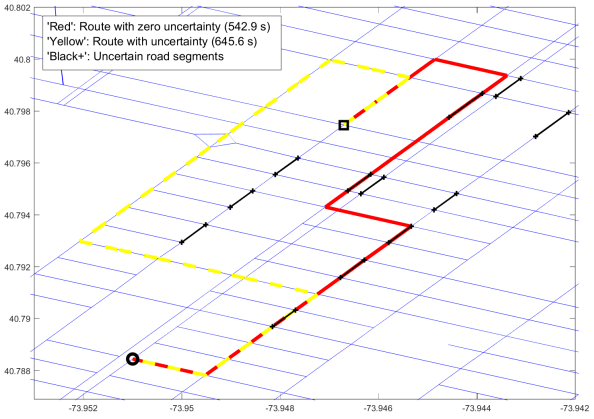



**FIGURE 11.** Avoidance of uncertain roads using WSA and robust optimization.

exists. The yellow and pink lines represent the selected routes with low and high uncertainty levels, respectively. We notice that with high uncertainty level in some roads, the selected route usually becomes longer and time consuming, due to the fact that the vehicle tends to avoid those uncertain road segments.

In Fig. 11, we visualize how vehicles avoid uncertain roads (colored in black). The red line represents the original route selection with no uncertainty, while the yellow line represents the new selected route with some uncertainties. We find out that the vehicle avoids most of the uncertain roads in the map and reaches the destination with longer traveled distance and time.

## VI. DISCUSSION AND PERSPECTIVE

In this paper, we mainly investigated one among many potential applications of mobile and real-time crowdsourcing, that is, real-time navigation problem. In such an application, performances are significantly depending on the input data provided by ubiquitous sources having different characteristics and controlled by various owners. Therefore, data collection, availability, and treatment are all important steps to optimize before making real-time decisions.

Enabling real-time navigation requires the simultaneous collection of an important amount of data. Hence, dedicated infrastructure need to be deployed and novel technologies must be adopted to enable large-scale data exchange. Indeed, vehicular adhoc network (VANET) roadside units alone will not be sufficient to meet the requirements of effective real-time mobile crowdsourcing services due to the expensive installation cost and the authority limitation [32]. Fortunately, 5G will represent a great technological breakthrough in this context, by providing communication services with low latency, higher throughput, and increased reliability [33]. Moreover, hybrid and heterogeneous communication solutions must bel employed in a transparent manner to the cloud server, requesters, and road data sources (machine or human workers) such that reliable data exchange is guaranteed.

Effective real-time navigation requires the joint analysis of off-line, historical, and online data. Analytical and artificial intelligent techniques to estimate/predict the future traffic status in every segment of the road network and deal with uncertainty with contaminated data need to be designed to deliver accurate and precise information and allow the server determine least congested and fastest routes and avoid false-positive and false-negative prediction of blocked roads.

The data treatment is a complex problem requiring significant computational resources especially when dealing with hundreds or thousands of crowdsourcing units. Centralized solutions residing on a single cloud crowdsourcing server might not be the best solution for use-cases. Distributed solutions based on edge computing resources could be an alternative solution for such applications [34]. For instance, when navigating, during each window size, the route is provided by the nearest edge server instead of being limited to a single entity throughout the route. This will help in reducing congestion on the communication network and the cloud server itself. Effective coordination and data sharing among the entities of the decentralized approach is needed for reliable real-time navigation recommendation.

To sum-up. from a crowdsourcing perspective, there are three major challenges [35]:

- Quality control: address noisy and/or incorrect submissions,
- Latency control: deal with delays due to human workers's submissions which are usually slower compared to automated computing time scales,
- Cost control: finally, find solutions to reduce monetary cost since the crowd is not always free.

On the other hand, the trend of collecting crowdsourcing data is also paving the way towards efficient navigation of self-driving vehicles. Real-time navigation would very useful to future self-driving vehicles, which should not be solely depending on their own on-board sensors. Via VSN and crowdsourcing, autonomous vehicles can play at the same time the role of requester and worker by asking its peers some information about the road network to navigate and, at the same time, provide useful information to the server to help other vehicles navigate. It is challenging to develop a complete framework for machine-to-machine crowdsourcing allowing safe navigation for these vehicles.

Finally, privacy remains a continuous concern, for such applications, considering the sensitivity of data such as geo-spatial data that maps real-time movement of end-users, ITS crowdsourcing systems must be engineered securely first, incorporating powerful end-to-end encryption to protect transmissions from interception by malicious actors. Additionally, the systems should be designed to allow for end-users to be in control of their data, and should be explicitly informed in plain language how their information is used and shared, in order to protect their privacy, build trustful services, and improve the sustainability of ITS crowdsourcing platforms.

## VII. CONCLUSION

In this paper, we have designed real-time navigation algorithms by exploiting automatic sensed mobile crowdsourcing data to predict the status of the traffic network in urban areas and enable online route updates. We formulate an ILP to model the real-time navigation and designed ILP-based and graph-based approaches with different computational complexity levels to determine routes. The algorithms updates the routes every window size, if needed, according to the reported updates. Results have shown that our approaches outperform state-of-the-art algorithms by exploiting the knowledge about the road network status thanks to information sharing. Moreover, the proposed algorithms determine less congested routes while avoiding reported blocked streets in advance. Uncertainty in identifying road statuses is also addressed. Our algorithms choose routes with low risk of uncertainty to guarantee safe navigation with minimum error risk.

## ACKNOWLEDGMENT

A part of this work has been accepted for publication in IEEE Intl. Syst. Conf. (SYSCON'19), Orlando, Florida, USA, April 2019 [1].

## REFERENCES

[1] X. Wan, H. Ghazzai, and Y. Massoud, "Real-time navigation in urban areas using mobile crowd-sourced data," in *Proc. IEEE Intl. Syst. Conf. (SYSCON)*, Orlando, FL, USA, Apr. 2019.
[2] T. I. Bv, "Tomtom traffic index/New York congestion statistics," TomTom N.V., Amsterdam, The Netherlands, Tech. Rep., Jan. 2016. [Online]. Available: https://corporate.tomtom.com/node/25601/pdf
[3] K. Lord, "North avenue smart corridor: Intelligent mobility innovations in Atlanta improving safety and efficiency," Atkins, Epsom, U.K., Tech. Rep., Jul. 2018. [Online]. Available: https://www.atkinsglobal.com/en-gb/angles/all-angles/north-ave-smart-corridor
[4] N. Caceres, L. M. Romero, F. G. Benitez, and J. M. Del Castillo, "Traffic flow estimation models using cellular phone data," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 1430–1441, Sep. 2012.
[5] X. Chen and L. Wang, "A cloud-based trust management framework for vehicular social networks," *IEEE Access*, vol. 5, pp. 2967–2980, 2017.
[6] A. M. Vegni and V. Loscrí, "A survey on vehicular social networks," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2397–2419, 4th Quart., 2015.
[7] Y. R. B. Al-Mayouf, N. F. Abdullah, O. A. Mahdi, S. Khan, M. Ismail, M. Guizani, and S. H. Ahmed, "Real-time intersection-based segment aware routing algorithm for urban vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2125–2141, Jul. 2018.
[8] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[9] S. Orgera, "What is waze and how does it work?" Lifewire, New York, NY, USA, Tech. Rep., Nov. 2018. [Online]. Available: https://www.lifewire.com/what-is-waze-4153570
[10] E. Wirtschafter, "Driving apps like waze are creating new traffic problems," KALW, San Francisco, CA, USA, Tech. Rep., Mar. 2017. [Online]. Available: https://www.kalw.org/post/driving-apps-waze-are-creating-new-traffic-problems#stream/0
[11] J. Moghaddasi and K. Wu, "Multifunctional transceiver for future radar sensing and radio communicating data-fusion platform," *IEEE Access*, vol. 4, pp. 818–838, 2016.
[12] C. Roman, R. Liao, P. Ball, S. Ou, and M. de Heaver, "Detecting on-street parking spaces in smart cities: Performance evaluation of fixed and mobile sensing systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2234–2245, Jul. 2018.
[13] L. Tang, X. Yang, Z. Dong, and Q. Li, "CLRIC: Collecting lane-based road information via crowdsourcing," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 9, pp. 2552–2562, Mar. 2016.
[14] Y. Y. Ye, X. Li Hao, and H. J. Chen, "Lane detection method based on lane structural analysis and CNNs," *IET Intell. Transport Syst.*, vol. 12, no. 6, pp. 513–520, 2018.
[15] X. Fan, J. Liu, Z. Wang, Y. Jiang, and X. S. Liu, "Crowdnavi: Demystifying last mile navigation with crowdsourced driving information," *IEEE Trans. Ind. Informat.*, vol. 13, no. 2, pp. 771–781, Oct. 2016.
[16] H. Yan and D.-J. Yu, "Short-term traffic condition prediction of urban road network based on improved SVM," in *Proc. IEEE Int. Smart Cities Conf. (ISC)*, Wuxi, China, Sep. 2017, pp. 1–2.
[17] R. Hussain, F. Abbas, J. Son, D. Kim, S. Kim, and H. Oh, "Vehicle witnesses as a service: Leveraging vehicles as witnesses on the road in vanet clouds," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Bristol, UK, Dec. 2013, pp. 439–444.
[18] W. Alasmary, H. Sadeghi, and S. Valaee, "Crowdsensing in vehicular sensor networks with limited channel capacity," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Budapest, Hungary, Jun. 2013, pp. 1833–1838.
[19] X. Zhang, Z. Yang, and Y. Liu, "Vehicle-based bi-objective crowdsourcing," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 10, pp. 3420–3428, Oct. 2018.
[20] D. Cerotti, S. Distefano, G. Merlino, and A. Puliafito, "A crowd-cooperative approach for intelligent transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1529–1539, Jun. 2017.
[21] Z. Li, I. V. Kolmanovsky, E. M. Atkins, J. Lu, D. P. Filev, and Y. Bai, "Road disturbance estimation and cloud-aided comfort-based route planning," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3879–3891, Jul. 2016.
[22] J.-D. Zhang, Y.-J. Feng, F.-F. Shi, G. Wang, B. Ma, R.-S. Li, and X.-Y. Jia, "Vehicle routing in urban areas based on the oil consumption weight-Dijkstra algorithm," *IET Intell. Transp. Syst.*, vol. 10, no. 7, pp. 495–502, 2016.
[23] C. Ruan, J. Luo, and Y. Wu, "Map navigation system based on optimal dijkstra algorithm," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. Intell. Syst.*, Shenzhen, China, Nov. 2014, pp. 559–564.
[24] K. Kaneko and S. Honda, "A map database system for route navigation with multiple transit points and destination points," in *Proc. 5th IIAI Int. Congr. Adv. Appl. Informat. (IIAI-AAI)*, Kumamoto, Japan, Jul. 2016, pp. 219–223.
[25] J. Zhang, J. Fan, and Z. Luo, "Generating multi-destination maps," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 8, pp. 1964–1976, Aug. 2017.
[26] W. Rahiman and Z. Zainal, "An overview of development GPS navigation for autonomous car," in *Proc. IEEE 8th Conf. Ind. Electron. Appl. (ICIEA)*, Melbourne, VIC, Australia, Jun. 2013, pp. 1112–1118.
[27] Y. Zhao and Q. Han, "Spatial crowdsourcing: Current state and future directions," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 102–107, Jul. 2016.
[28] J. Löfberg, "Automatic robust convex programming," *Optim. Methods Softw.*, vol. 27, no. 1, pp. 115–129, Sep. 2012.
[29] A. Bahabry, X. Wan, H. Ghazzai, H. Menouar, G. Vesonder, and Y. Massoud, "Low-altitude navigation for multi-rotor drones in urban areas," *IEEE Access*, vol. 7, pp. 87716–87731, 2019.
[30] J. C. Dela Cruz, G. V. Magwili, J. P. E. Mundo, G. P. B. Gregorio, M. L. L. Lamoca, and J. A. Villaseñor, "Items-mapping and route optimization in a grocery store using Dijkstra's, Bellman-Ford and Floyd-Warshall Algorithms," in *Proc. IEEE Region 10 Conf. (TENCON)*, Singapore, Nov. 2016, pp. 243–246.
[31] G. Boeing, "OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Comput. Env. Urban Syst.*, vol. 65, pp. 126–139, Sep. 2017.
[32] L. Xue, Y. Yang, and D. Dong, "Roadside infrastructure planning scheme for the urban vehicular networks," *Transp. Res. Procedia*, vol. 25, pp. 1380–1396, May 2017.
[33] C. M. Silva, B. M. Masini, G. Ferrari, and I. Thibault, "A survey on infrastructure-based vehicular networks," *Mobile Inf. Syst.*, vol. 2017, Apr. 2017, Art. no. 6123868.
[34] M. Marjanovic, A. Antonic, and I. P. Žarko, "Edge computing architecture for mobile crowdsensing," *IEEE Access*, vol. 6, pp. 10662–10674, 2018.
[35] G. Li, J. Wang, Y. Zheng, and M. J. Franklin, "Crowdsourced data management: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 9, pp. 2296–2319, Sep. 2016.

**XIANGPENG WAN** received the bachelor's degree in electrical engineering from the Harbin Institute of Technology, Harbin, China, in 2015, and the master's degree in applied mathematics from the University of Minnesota, Duluth, MN, USA, in 2017. He is currently pursuing the Ph.D. degree in engineering management with the Stevens Institute of Technology, Hoboken, NJ, USA. His research interests include smart city design, big data analysis, and applied machine learning.

**HAKIM GHAZZAI** (S'12–M'15) received the Ph.D. degree in electrical engineering from KAUST, Saudi Arabia, in 2015, and the Diplome d'Ingenieur (Hons.) in telecommunication engineering and the master's degree in high-rate transmission systems from the Ecole Superieure des Communications de Tunis (SUP'COM), Tunis, Tunisia, in 2010 and 2011, respectively. He was a Visiting Researcher with Karlstad University, Sweden, and a Research Scientist with the Qatar Mobility Innovations Center (QMIC), Doha, Qatar, from 2015 to 2018. He is currently a Research Scientist with the Stevens Institute of Technology, Hoboken, NJ, USA. His general research interests include the intersection of wireless networks, UAVs, the Internet of Things, intelligent transportation systems, and optimization.

**YEHIA MASSOUD** (F'15) received the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA. He has held several industrial and academic positions, including a member of the technical staff with the Advanced Technology Group, Synopsys, Inc., Mountain View, CA, USA, a tenured Associate Professor with the Departments of Electrical and Computer Engineering and Computer Science, Rice University, Houston, TX, USA, the W. R. Bunn Head of the Department of Electrical and Computer Engineering, The University of Alabama, Birmingham, Birmingham, AL, USA, and the Head of the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, USA. He is currently the Dean of the School of Systems and Enterprises, Stevens University of Science and Technology, Hoboken, NJ, USA. He leads research efforts in various areas of electrical and computer engineering, computing, and systems and software engineering. He has authored over 225 articles in peer-reviewed journals and conferences. He was an elected member of the IEEE Nanotechnology Council, from 2009 to 2011. He was selected as one of the ten MIT Alumni Featured in the MIT EECS Newsletter, in 2012. He was a recipient of the Rising Star of Texas Medal, in 2007, the National Science Foundation CAREER Award, in 2005, the DAC Fellowship, in 2005, the Synopsys Special Recognition Engineering Award, in 2000, several best paper award nominations, and two best paper awards at the IEEE International Symposium on Quality Electronic Design, in 2007, and the IEEE International Conference on Nanotechnology, in 2011. He served as the Theme Leader of Novel Interconnects and Architectures in the SRC Southwest Academy of Nanoelectronics, from 2006 to 2011. He was named a Distinguished Lecturer by the IEEE Circuits and Systems Society, from 2014 to 2015.