Proactive Vehicular Traffic Rerouting for Lower Travel Time

Juan Pan, Iulian Sandu Popa, Karine Zeitouni, and Cristian Borcea, *Member, IEEE*

***Abstract*—Traffic congestion causes driver frustration and costs billions of dollars annually in lost time and fuel consumption. This paper presents five traffic rerouting strategies designed to be incorporated in a cost-effective and easily deployable vehicular traffic guidance system that reduces travel time. The proposed strategies proactively compute individually tailored rerouting guidance to be pushed to vehicles when signs of congestion are observed on their route. The five proposed strategies are the dynamic shortest path (DSP), the A^* shortest path with repulsion (AR^*), the random k shortest path (RkSP), the entropy-balanced kSP (EBkSP), and the flow-balanced kSP (FBkSP). Extensive simulation results show that the proposed strategies are capable of reducing the travel time as much as a state-of-the-art dynamic traffic assignment (DTA) algorithm while avoiding the issues that make DTA impractical, such as the lack of scalability and robustness, and high computation time. Furthermore, the variety of proposed strategies allows tuning the system to different levels of tradeoffs between rerouting effectiveness and computational efficiency. In addition, the proposed traffic guidance system can significantly improve the traffic even if many drivers ignore the guidance or if the system adoption rate is relatively low.**

***Index Terms*—Proactive driver guidance, traffic load balancing, vehicular congestion avoidance, vehicular networks.**

I. Introduction

DESPITE significant advances in in-car navigation systems (e.g., Garmin and TomTom), web services for route computation (e.g., Google and Microsoft), and dynamic traffic assignment (DTA) [9], [24], we are still spending a lot of time in traffic jams. It is predicted that, by 2015, the congestion cost will rise to $133 billion, and the amount of wasted fuel will jump to 2.5 billion gallons [34]. Hence, finding effective solutions for congestion mitigation at a reasonable cost is becoming a stringent problem.

With the deployment of traffic surveillance infrastructure on more roads (e.g., loop detectors and video cameras), we have started to witness web-based services/applications that present drivers with the current view of the traffic and let them decide which route to follow. However, the usefulness of these applications is limited: 1) They have mostly accurate information about the highways, and thus not very useful for city traffic; and 2) they cannot prevent congestion, and at the same time, it is known that no true equilibrium can be found under congestion [9].

Recently, companies such as Google [1] and TomTom [2] have started to use infrastructure-based traffic information to compute traffic-aware shortest routes. However, these solutions do not try to prevent congestion explicitly (i.e., they are reactive solutions) and provide the same guidance for all vehicles on the road at a certain moment as a function of their destination (i.e., a pull model in which drivers query for the shortest route to the destination). Therefore, similar to route oscillation in computer networks, they could lead to unstable global traffic behavior. When it happens, congestion is switched from one route to another if a significant number of drivers use the guidance.

This situation could be avoided by new solutions on dynamic user-optimal traffic assignment [9]. These solutions periodically compute the assignment of traffic flows to routes that lead to user equilibrium. Unfortunately, there is still a significant gap between the theoretical or simulation results and potentially deployable solutions. Some issues are as follows: tractability for large-scale road networks, capability of providing real-time guidance, behavior in the presence of congestion, ability to work when not all drivers are part of the system, and robustness to drivers who ignore the guidance.

The time is ripe for building a proactive intelligent real-time traffic guidance system based on the dynamic situations on the road network. In this system, vehicles can be viewed as both mobile sensors (i.e., collecting real-time traffic data) and actuators (i.e., changing their path in response to newly received guidance). The system is cost effective and easily deployable because it does not require roadside infrastructure; it can work using only smartphones carried by drivers.[1] When roadside sensors are available, the system can take advantage of them to supplement the data provided by vehicles to build an accurate representation of the global real-time traffic conditions. When the signs of congestion are observed on certain road segments, it computes proactive individually tailored rerouting guidance, which is pushed to vehicles that would pass through the congested segments.

This paper introduces a cost-effective and easily deployable vehicular traffic guidance system that reduces the effect of

Manuscript received October 1, 2012; revised February 12, 2013; accepted April 13, 2013. Date of publication April 26, 2013; date of current version October 12, 2013. This work was supported in part by KISS, a research project funded by French ANR Call INS 2011, and the National Science Foundation under Grant CNS-0831753. The review of this paper was coordinated by Dr. P. Lin.

J. Pan and C. Borcea are with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102-1982 USA (e-mail: jp238@njit.edu; borcea@njit.edu).

I. S. Popa is with the Department of Computer Science, University of Versailles Saint-Quentin-en-Yvelines, Versailles 78000, France, and also with Inria Paris-Rocquencourt, Le Chesnay 78145, France (e-mail: iulian.sandu-popa@prism.uvsq.fr).

K. Zeitouni is with the Department of Computer Science, University of Versailles Saint-Quentin-en-Yvelines, Versailles 78000, France (e-mail: karine.zeitouni@prism.uvsq.fr).

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/TVT.2013.2260422

[1] In the future, once vehicular embedded systems become widespread, they could be used instead of smartphones.

traffic congestion. Then, this paper evaluates five rerouting strategies designed to be incorporated in this system: 1) the dynamic shortest path (DSP), which assigns to each vehicle the current shortest time path to the destination; 2) the A^* shortest path with repulsion (AR^*), which modifies the A^* shortest path algorithm [17] by considering both the travel time and the paths of the other vehicles (as a repulsive force) in the computation of the shortest path; 3) the random k shortest path (RkSP), which computes kSPs[2] for each rerouted vehicle and randomly assigns the vehicle to one of them; 4) the entropy-balanced kSP (EBkSP), which computes kSPs for each vehicle and assigns the vehicle to the path with the lowest popularity as defined by the path entropy; and 5) the flow-balanced kSP (FBkSP), which computes kSPs for each vehicle and assigns the vehicle to the path that minimizes the impact of traffic flow in a network region. Specifically, our contributions are as follows.

- We propose five novel rerouting strategies and extensively evaluate them through various simulation settings over two medium-sized urban road networks. All our strategies result in significantly lower average travel time compared with a "no-rerouting" baseline. Among the proposed strategies, the AR^* has the lowest average travel time but has the highest computation cost. EBkSP and FBkSP can achieve comparable travel times to the AR^* while demanding lower CPU times for the rerouting computations.
- We employ a tool implementing a state-of-the-art DTA algorithm for traffic optimization to compare it against our strategies. Compared with DTA, our strategies obtain similar travel times at a (much) lower computation cost. In addition, our strategies are much more scalable with the number of vehicles than DTA.
- We measure the robustness of the system by investigating the compliance rate and the penetration rate. The results show that our strategies are still effective in alleviating congestion even if many drivers ignore the guidance or if the system adoption rate is relatively low, which is important in facilitating the adoption of the system at a large scale.

The remainder of this paper is organized as follows. Section II discusses related work. Section III describes the system model and our assumptions. Section IV introduces the five rerouting strategies and the DTA algorithm used as a baseline. Section V presents the experimental results and their analysis. We conclude and give directions for future work in Section VI.

II. Related Work

Projects such as Mobile Millennium [42], [20], CarTel [11], JamBayes [19], Nericell [30], and surface street estimation [16] use vehicle probe data collected from onboard GPS devices to reconstruct the state of traffic and to estimate the shortest travel time. This paper moves beyond this idea. Instead of investigating the feasibility and accuracy of using mobile phones as traffic sensors, we are focusing on using that information to recommend routes more intelligently, thus achieving better efficiency in terms of avoiding congestion and reducing travel time.

Services such as INRIX [3] provide real-time traffic information at certain temporal accuracy, which allows drivers to choose alternative routes if they are showing lower travel times. Systems such as Google Maps and Microsoft's Bing are able to forecast congestion and its duration by performing advanced statistical predictive analysis of traffic patterns. Additionally, short-term nonrecurrent congestion can be predicted as well [4]. Based on this information, according to Wardrop's first traffic equilibrium principle [40], drivers could be able to reach a user-optimal traffic equilibrium. It is known, however, that no true equilibrium can be found under congestion [21]. Even more importantly, the usefulness of such services is limited by their reactive nature, i.e., they cannot prevent congestion. Our solution moves one step forward by providing effective methods for proactive rerouting when congestion is predicted based on real-time traffic information.

A large body of existing route planning research focuses on fast generation of kSPs [35], [36] in highly dynamic scenarios with frequent traffic information updates. In particular, in [35], a transit-node routing and highway-node routing to reduce the average query time and memory requirements are presented. The work in [26] proposes two new classes of approximation techniques that use precomputation and avoidance of complete recalculations on every update to speed up the processing of continuous route planning queries. However, current instantaneous shortest paths are not necessarily equal to time-dependent shortest paths. These algorithms calculate the shortest paths based only on the snapshot of current traffic conditions without considering the dynamic future conditions.

One of the essential properties of the road network is the time dependence of the travel time. Computing the shortest paths in a time-varying spatial network is challenging since the edge (i.e., the road segment) travel time changes dynamically. In this case, the computation not only considers the instantaneous travel time in one single snapshot of the traffic graph but also the relationship among the consecutive snapshots across time. George *et al.* [15] demonstrated a fast greedy time-dependent shortest path algorithm (SP-TAG) by using a time-aggregated graph (TAG) data structure instead of a time-expanded graph. The SP-TAG saves storage and computation cost by allowing the properties of edges and nodes to be modeled as a time series, instead of replicating nodes and edges at each time unit. While algorithms such as the SP-TAG provide insights into the dynamics of traffic networks, two obstacles remain, in addition to an increased computational cost. First, it is impractical to assume that the system knows the exact travel time series of every single road segment given the traffic dynamics. Second, these algorithms do not help with switching congestion from one spot to another if all the drivers are provided the same time-dependent shortest path.

An alternative to this paper could be the research done on DTA, which leads to either system-optimal or user-optimal route assignments. DTA research can be classified into two categories: analytical methods and simulation-based models. Analytical models, such as those in [13], [29], and [28],

[2]We use kSPs for short for k shortest paths in the remainder of this paper.

formulate DTA as either nonlinear programming problems, optimal control problems, or variational inequalities. Although they provide theoretical insights, the computational intractability prevents their deployment in real systems [32].

Simulation-based approaches, in which the time-dependent user equilibrium is computed by iterative simulations [9], [14], [25], [39] have gained greater acceptability in recent years. The simulations are used to model the theoretical insights that cannot be derived from analytical approaches. This process computes the assignment of traffic flows until the travel times of all drivers are stationary. Unfortunately, there are still a number of issues associated with these approaches that make their deployment difficult: tractability for large-scale road networks given the computational burden associated with the simulator, capability of providing real-time guidance, effectiveness in the presence of congestion, and the behavior of drivers who ignore the guidance. For example, they assume that the set of origin–destination (OD) pairs and the traffic rate between every OD pair are known. This information is highly dynamic, particularly in city scenarios, leading to frequent iterations of computationally expensive algorithms even when not needed from a driver benefit's point of view. Additionally, the OD set is large, and the DTA algorithms may not be able to compute the equilibrium fast enough to inform the vehicles about their new routes in time to avoid congestion. Our system, on the other hand, is designed to be effective and fast, although not optimal, in deciding which vehicles should be rerouted when signs of congestion occur and in computing alternative routes for these vehicles.

The complexity of DTA systems has led scientists to look for inspiration in biology and Internet protocols. In [36], Wedde *et al.* developed a road traffic routing protocol, i.e., BeeJamA, which is based on the behavior of honey bees. Similarly, Tatomir *et al.* [38] proposed a route guidance system based on the trail-laying ability of ants. Inspired by the well-known Internet routing protocols, Prothmann *et al.* [33] proposed decentralized organic traffic control. However, since they employ ad hoc networking, these approaches have only a partial view of the traffic conditions, which may lead to less accurate rerouting. In addition, simply treating vehicles as packets, which always listen to the guidance, ignores the nature of human behavior. Furthermore, these systems react to real-time data without insight into future conditions, thus introducing greater vulnerability to switching congestion from one spot to another.

III. System Model

Our traffic guidance system is composed of: 1) a centralized traffic monitoring and rerouting service (which can physically be distributed across several servers); and 2) a vehicle software stack for periodic traffic data reporting (position, speed, direction) and for showing alternative routes to drivers. Vehicles run this software either on a smartphone or an embedded vehicular system. Vehicles are equipped with GPS receivers and can communicate with the service over the Internet when needed. When starting a trip, each vehicle informs the service of its current position and destination; the service sends back a route that is computed according to its strategy. It is assumed that the service knows the road network and the capacity and legal speed limits on all roads.

Logically, the traffic guidance system operates in four phases executed periodically: 1) data collection and representation; 2) traffic congestion prediction; 3) vehicle selection for rerouting; and 4) alternative route assignment for each vehicle and pushing the guidance to the vehicles. Since data collection has been extensively studied in the literature, we do not address this issue and assume that the centralized service receives traffic data from vehicles and roadside sensors when available. We discuss in detail each of the other phases here and in Section IV.

A. Traffic Data Representation and Estimation

The road network is represented as a directed weighted graph, where nodes correspond to intersections, edges to road segments, and weights to estimated travel times. The weights are periodically updated as new traffic data become available. Several methods can be employed to estimate the travel time over a road segment. For instance, using vehicle probe data collected from onboard GPS devices to reconstruct the state of traffic is a well-studied topic [24], [42]. We use Greenshield's model [6] to estimate the travel time since it is extensively used in DTA models by transportation researchers. The model considers that there is a linear relationship between the estimated road speed V_i and the traffic density K_i (vehicles per meter) on road segment i, i.e.,

$$V_i = V_f \left(1 - \frac{K_i}{K_{\text{jam}}}\right) \quad T_i = L_i/V_i \tag{1}$$

where K_{jam} and V_f are the traffic jam density and the free-flow speed for road segment i, respectively; and T_i and L_i are the estimated travel time and length for the same segment, respectively. The free-flow speed V_f is defined as the average speed at which a motorist would travel if there were no congestion or other adverse conditions. To simplify our implementation, we consider that the free-flow speed is the road speed limit. Basically, K_i/K_{jam} is the ratio between the current number of vehicles and the maximum number of vehicles. The current number of vehicles is obtained from the traffic data collected by the service, whereas the maximum number of vehicles is equal to length of road$/$(avg vehicle length + min gap).

B. Congestion Prediction

Periodically, the service checks the road network to detect signs of congestion. A road segment is considered to exhibit congestion signs when $K_i/K_{\text{jam}} > \delta$, where $\delta \in [0, 1]$ is a predefined threshold value. Choosing the right value for δ is particularly important for the service performance. If it is too low, the service could trigger unnecessary rerouting; this may lead to an increase in the drivers' travel times. If it is too high, the rerouting process could be triggered too late, and congestion will not be avoided. The evaluation in Section V-B confirms these hypotheses.

C. Selection of Vehicles to Be Rerouted

When a certain road segment presents signs of congestion, the service looks for nearby vehicles to reroute. Specifically, we select vehicles from incoming segments (i.e., segments that bring traffic into the congested one). To decide how far from congestion to look for candidates for rerouting, the service uses parameter L (level). This parameter denotes the farthest distance (in number of segments) that a candidate vehicle can be away from the congested segment. In practice, L could be computed as a function of the severity of congestion; for example, we can use the "level of service" defined in the Highway Capacity Manual [27]. The value of L has to be large enough to mitigate congestion. If L is too high, however, more vehicles than necessary will be selected for rerouting, which can have undesired consequences (e.g., creating congestion in another spot). Since our focus is on the rerouting algorithms and the analysis of their performance, we decided to consider L a tuning parameter that is varied during our experiments. We plan to investigate an efficient way for selecting its value in the presence of dynamic traffic conditions in future work.

The service performs a breadth-first search on the inverted network graph (i.e., the road network graph is directed), starting from the congested segments with maximum depth L, and considers all these cars as candidates for rerouting. The process is shown in Fig. 1. Assuming that $L = 2$ and the signs of congestion are detected on the segment Sc, the system recursively selects the vehicles situated on the incoming segments of the congested segment in two steps. First, the vehicles located on segments S1 are included in the candidate set, followed by the vehicles situated on segments S2.

We select the union of all the vehicles affected by all the congested segments in the whole network within the same level and then perform vehicle ranking (cf. Section III-D), path computations, and path assignments (cf. Section IV).

D. Ranking the Selected Vehicles

The selected vehicles need to be ranked and assigned to alternative paths according to their rank for all strategies, except for the DSP. In this way, the performance of the strategies improves. The impact that a congested road segment has on the travel time of a vehicle is different depending on the remaining distance to the destination of the vehicle. Intuitively, the drivers that are close to their arrival point may have a different perception of the congestion than the drivers that are far away from their destination. Our system uses an urgency function to rank the vehicles that are selected for rerouting. Hence, the vehicles with higher urgency are rerouted first and get relatively better routes.

Definition 1: Given a set of vehicles $V = (v_1, v_2, v_3, \ldots, v_m)$ to be rerouted, we define two urgency functions to compute the rerouting priority of a vehicle in V:

- absolute congestion impact: ACI = RemTT-RFFTT;
- relative congestion impact: RCI = (RemTT-RFFTT)/RFFTT

where RemTT is the remaining travel time, and RFFTT is the remaining free-flow travel time for the vehicle.

ACI measures the impact of all the (congested) segments of the remaining journey of a vehicle since (RemTT-RFFTT) is the absolute increase in the travel time with respect to the free-flow travel time. With ACI, the longer the remaining distance to the destination for a vehicle is, the higher is the probability for that vehicle to get a higher rank (as the difference RemTT-RFFTT normally increases with the RFFTT). On the other hand, RCI weighs the congestion impact on a vehicle relative to its remTT. Hence, RCI gives a higher priority to vehicles that are close to their destination. Since the farther the vehicle is from its the destination, the higher are the number of alternative paths and the potential benefit of rerouting, we expect ACI to perform better than RCI in our system.

IV. Rerouting Strategies

Recent research has proven that real-time traffic flow data and road travel time can be determined based on data reported by vehicles or roadside sensors [19], [30], [42]. The question is how to utilize this knowledge in an intelligent fashion to avoid congestion and to reduce the drivers' travel times. Here, we present five rerouting strategies that we classify in two categories. The first category, presented in Section IV-A, includes two rerouting strategies that compute a single alternative new path for each rerouted vehicle. The strategies are based on the well-known Dijkstra algorithm and on the A^* algorithm with a modified heuristic, respectively. Section IV-B presents the second category consisting of three rerouting strategies that compute multiple alternative new paths for each of the rerouted vehicles. Then, different heuristics are used to choose the best alternative path to be assigned to a vehicle. Based on the five proposed strategies, we describe the main rerouting process executed by the traffic guidance system in Section IV-C. Finally, Section IV-D presents a DTA strategy [14] that we use as a baseline to measure the effectiveness and efficiency of the proposed strategies.

A. Single Shortest Path Strategies

1) DSP: The DSP is a classical rerouting strategy that assigns the selected vehicles to the path with the lowest travel time. However, different from existing systems, our system takes a proactive approach. Specifically, each time that a road segment presents signs of congestion, the service obtains the set of cars whose paths intersect this road segment and computes for each car a new shortest path based on the current travel time in the road network. Therefore, the path of each car can be periodically updated on an event-driven basis. The advantage of this strategy lies in its simplicity and, consequently, reasonable computational cost, i.e., $O(E + V \log V)$ [12], where E is the number of road segments, and V is the number of intersections of the road network. We expect this strategy to provide good results when the number of rerouted vehicles is low since, in this case, the risk of switching congestion from one spot to another is low. Hence, locally redirecting the traffic when congestion happens should be sufficient in this case. On the other hand, when the traffic density is higher, there is an increased risk of switching the congestion from one road to

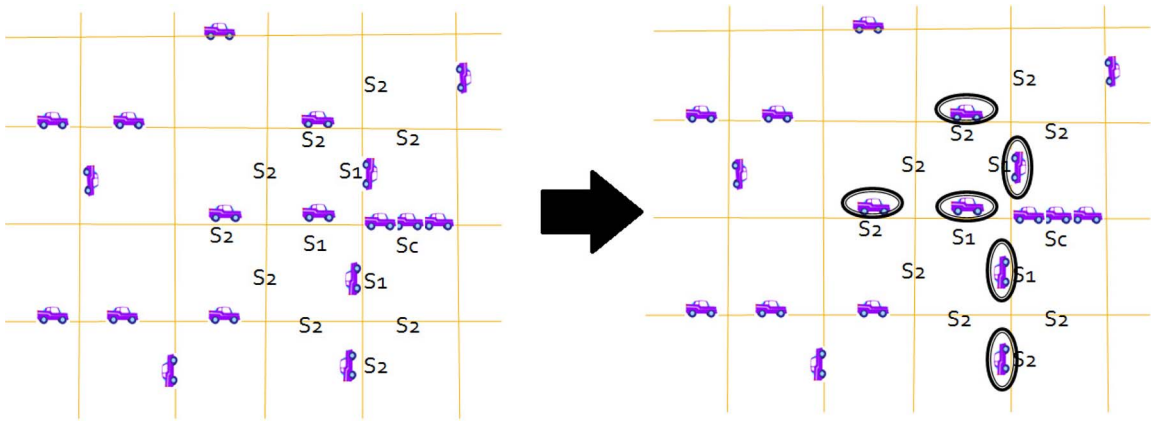


Fig. 1. Vehicle selection process.

another. Moreover, the rerouting frequency (RRF) for a driver is likely to increase in this case, which can be annoying to drivers.

2) AR:* The DSP strategy only takes into account the current view of the traffic when performing rerouting, without considering the impact that the rerouting will have on the future traffic. To address this limitation, we propose the AR*, which modifies the A^* search algorithm to include the prior rerouting decisions into the computation of the current shortest path. A^* [17] uses a best-first search and a heuristic function to determine in which order to visit the network nodes (road intersections in our case). Given node x, heuristic function $F(x)$ is computed as the sum $G(x) + H(x)$. $G(x)$ is the path cost from the start node to x, which corresponds to the travel time in our case, whereas $H(x)$ is a heuristic estimation of the remTT from x to the destination node. In addition, $H(x)$ has to be "admissible" (i.e., it must not overestimate the remTT to the destination) to produce the shortest path between the source and the destination. Therefore, $H(x)$ is computed as the Euclidean distance divided by the maximum speed in the road network.

Future congestion occurs if many drivers take the same road segment within the same future time window.[3] As we assume that the drivers share their route information with the service, it is possible to estimate the future footprint of each driver in the road network.

Definition 2: A weighted footprint counter fc_i of a road segment i is defined as $\text{fc}_i = n_i \times \omega_i$, where n_i is the total number of vehicles that are assigned to paths that include segment i, and ω_i is a weight associated with i. $\omega_i = (\text{len}_{\text{avg}}/\text{len}_i \times \text{lane}_i) \times (V_{f_{\text{avg}}}/V_{f_i})$, where len_{avg} is the average road segment length in the network, $V_{f_{\text{avg}}}$ is the average free-flow speed of the network, len_i is the length of i, V_{f_i} is the free-flow speed of i, and lane_i is the number of lanes of i.

In the formula, n_i represents the discretized future traffic flow on road i. We decided to use weights in the formula to count for different road characteristics. For example, suppose that there are two road segments r_i and r_j. Although $n_i = n_j$, the segments should not be equally treated since r_i has a higher capacity (more lanes or longer length); thus, the possibility of causing congestion is lower. In other words, the impact of the traffic flow n_i on road r_i is lower than n_j on r_j, although $n_i = n_j$.

In the AR*, we modify heuristic function $F(x)$ to include the other vehicles sharing the same path as a repulsive force. Specifically, we define the repulsive score $R(x)$ of node x as the sum of the weighted footprint counters (cf. Definition 2) from the starting node to node x. Thus, the path-cost function becomes $F(x) = (1 - \beta) \times (G(x) + H(x)) + \beta \times R(x)$, where $G(x)$ and $H(x)$ are computed as in the original algorithm, and β is a weighting parameter. $G(x) + H(x)$ measures the travel time factor, whereas $R(x)$ reflects the impact of other vehicle traces on the examined path. Since the travel time and the repulsive force use different metrics, we normalize their values and compute $F(x)$ as a linear combination of the two factors. The parameter β allows a variable weighting between the travel time factor and the repulsive force factor. If β is too large, the repulsive force factor becomes predominant, and the resulting path can be diverted too far away from the shortest path. Conversely, if β is too low, the AR* will behave similar to the naive DSP strategy. In our experiments, we empirically determine the value of β that leads to the best effectiveness for the AR* algorithm.

The complete algorithm is presented as a pseudocode in Algorithm 1. Starting from the initial node, the algorithm maintains a queue of nodes to be traversed, which is denoted as the open set (lines 3–5). At each iteration, the node with the lowest F-score value is removed from the queue (lines 16–19), the values of its neighbors are accordingly updated (lines 27–28), and these neighbors are added to the queue (line 30). The algorithm continues until the end node has been reached or until the queue is empty. The normalization of the travel time and the repulsive force factors is done at line 14. A path is returned at line 18 if found; otherwise, an empty path is returned in line 48.

Fig. 2 shows a simple example of how the AR* is used in rerouting. We suppose that vehicles v_1, v_2, v_3 having the same origin and destination, i.e., from ab to ij, need to be rerouted and that $\text{urgency}(v_1) > \text{urgency}(v_2) > \text{urgency}(v_3)$. At the beginning, since no vehicle has been assigned any path, the AR* performs normal A^* search and assigns the shortest path ab, bc, cd, di, ij to vehicle v_1. When computing the shortest path for vehicle v_2, the AR* will find ab, bg, gh, hi, ij. Although v_2 has the same destination as v_1, the path found by the

[3]The time window size is equal to the period used by the system to evaluate congestion.

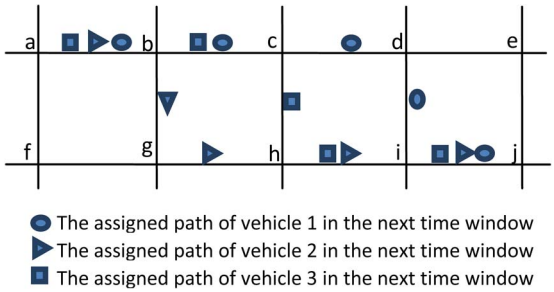


Fig. 2. AR* rerouting example. All road segments have the same weight and $\beta = 0.5$.

AR* is different since it considers the footprints produced by v_1 as a repulsion. Hence, the AR* avoids the already assigned paths as much as possible while still keeping the new path as short as possible. Finally, the procedure is repeated for vehicle v_3, and path ab, bc, cd, di, ij is obtained for the same reasons.

Notice that the AR* has to be employed by the rerouting system in an iterative manner. Namely, after the selected vehicles to be rerouted have been ranked based on their urgency, the system sequentially calculates each vehicle's route starting from the most urgent one. Therefore, in the case of the AR*, the computation time linearly increases with the number of rerouted vehicles. On the other hand, as explained in the following, the rest of the proposed rerouting methods optimize this phase by grouping the vehicles to be rerouted based on their OD pair, which leads to lower computational complexity.

Algorithm 1 A Star Shortest Path with Repulsion Rerouting

```
1: procedure AstarRepulsion(start, end)
2: P[start] = empty {the reverse pointer of the path, which is
   used to reconstruct the path}
3: closedset = set() {The set of nodes already evaluated}
4: openset = set()
5: openset.add(start) {The set of tentative nodes to be evalu-
   ated, initially containing the start node}
6: Gscore[start] = 0.0 {Travel time cost from start along best
   known path}
7: Hscore[start] = Euclidean(start, end)/maxspeed
8: Rscore[start] = 0.0
9: Fscore[start] = 1.0
10: while openset is not empty do
11:    sumF = SumFscore(openset)
12:    sumR = SumRscore(openset)
13:    for all node in openset do
14:       Fscore[node] = (1 − β)∗ Fscore[node]/SumF +β∗
          Rscore[node]/SumR
15:    end for
16:    current = getleastFscore(openset)
17:    if current == end then
18:       return (Fscore[current], P)
19:    end if
20:    openset.remove(current)
21:    closedset.add(current) {add current to closedset}
22:    for all edge in current.outEdges do
23:       node = edge.endnode
24:       if node in closedset then
25:          continue
26:       end if
27:       tentative_g_score = Gscore[current] +
          edge.actualtime
28:       tentative_r_score = Rscore[current] +
          edge.weight_footprints
29:       if node not in openset then
30:          openset.add(node)
31:          Hscore[node] = Euclidean(node, end)/maxspeed
32:          tentative_is_better = True
33:       else
34:          if tentative_g_score < Gscore[node] then
35:             tentative_is_better = True
36:          else
37:             tentative_is_better = False
38:          end if
39:       end if
40:       if tentative_is_better == True then
41:          P[node] = edge
42:          Gscore[node] = tentative_g_score
43:          Rscore[node] = tentative_r_score
44:          Fscore[node] = Gscore[node] + Hscore[node]
45:       end if
46:    end for
47: end while
48: return (0.0, {})
49: end procedure
```

B. Multiple Shortest Path Strategies

The two strategies proposed earlier compute a single path for each rerouted vehicle. However, the two methods have opposite behaviors. On the one hand, the DSP sacrifices effectiveness (since it does not consider the impact of rerouting on the future traffic) to optimize the computational cost (by grouping the rerouted vehicles on their OD). On the other hand, the AR* trades efficiency (since it computes an alternative path for each vehicle) for effectiveness (by taking into account the future traffic configuration). Here, we introduce a new class of rerouting strategies to obtain the best tradeoff between efficiency and effectiveness. For these strategies, the rerouting process is divided into two steps. First, loopless kSPs are computed for each selected vehicle based on the travel time in the road network, where k is a predefined parameter. Compared with the DSP, this approach involves a higher computation time, but it still permits to group vehicles on their OD. Therefore, we expect the computation time to be lower than the AR*. Second, the vehicles are assigned to one of their kSPs on the order of their ranking. Among the kSPs, the algorithm generally selects the path that is the least employed by other vehicle traces. Hence, we expect this class of strategies to have effectiveness similar to the AR*. We propose three heuristics for the selection of the best path among the kSPs.

1) RkSPs*:* The RkSP assigns each selected vehicle to one of the k paths randomly. The goal is to avoid switching congestion from one spot to another by balancing the rerouted traffic among several paths. Compared with the DSP, the price to pay

is higher computational complexity, i.e., $O(kV(E + V \log V))$ [23], which linearly increases with k. Although a larger k will allow better traffic balancing, it also increases the difference in the travel time among the k paths. Therefore, to prevent an excessive increase in the travel time for some drivers, RkSP limits the maximum allowed relative difference between the fastest path and the slowest path to 20%.

2) EBkSPs*:* While the RkSP addresses the main potential shortcoming of the DSP (i.e., moving congestion to another spot), it has its own deficiencies. First, it increases the computational time, which matters because the alternative paths must be computed and pushed to vehicles before they pass the rerouting intersection. Second, it randomly assigns paths to vehicles, which is far from optimal both from a driver point of view and from the global traffic point of view. To address this second shortcoming of RkSP, we propose the EBkSP strategy. The idea is to perform a more intelligent path selection by considering the impact that each selection has on the future density of the affected road segments. The more intelligent path selection comes at the cost of slightly increased complexity. However, we expect this optimization to improve the traffic from a global point of view. In addition, as in the AR*, the EBkSP ranks the cars to be rerouted based on an urgency function that quantifies the degree to which the congested road affects the driver travel time. Thus, the more affected vehicles will have priority and be rerouted first.

The entropy idea comes from the Shannon information theory [37]. Several works [10], [43] have successfully applied it to compute the popularity of an area. To avoid creating new congestion through rerouting, we associate a "popularity" measure to road segments in the EBkSP. We use entropy to define the popularity of a path as follows.

Definition 3: Let $(p_1, \ldots, p_k)$ be the set of paths computed for the vehicle which will be assigned next. Let $(r_1, \ldots, r_n)$ be the union of all segments of $(p_1, \ldots, p_k)$, and let $(\mathrm{fc}_1, \ldots, \mathrm{fc}_n)$ be the set of weighted footprint counters associated with these segments. The popularity of p_j is defined as $\mathrm{Pop}(p_j) = e^{E(p_j)}$. $E(p_j)$ is the weighted entropy of p_j and is computed as $E(p_j) = -\sum_{i=1}^{n} (\mathrm{fc}_i/N) \ln(\mathrm{fc}_i/N)$, where $N = \sum_{i=1}^{n} n_i$.

The value of $E(p_j)$ measures the probability that a number of vehicles will be on path p_j in a time window. According to the earlier definition, we have $0 \leq \mathrm{Pop}(p_j) \leq m$, where m is the number of vehicles. $\mathrm{Pop}(p_j)$ has the maximum value m when every previously assigned vehicle entirely traverses p_j (i.e., they take the same path). $\mathrm{Pop}(p_j)$ has the minimum value when no one takes path p_j. Intuitively, the higher the popularity of a path, the higher the probability that more drivers will take this path.

After vehicle selection and ranking, we assign each vehicle to the least popular path among its kSPs to avoid potential future congestion. Specifically, the first vehicle is assigned the current best path without considering others. Then, the road network footprints are updated based on the new path. When assigning the second vehicle, the popularity score of its kSPs is calculated, and the least popular path will be chosen. The process is then repeated for the rest of the rerouted vehicles.

Fig. 3 shows an example of EBkSP rerouting. We assume that vehicles (v_1, v_2, v_3) have been initially assigned to their shortest time paths, and each road has the same weight $\omega_i = 1$. Vehicles v4 (identified by the question mark) arrives, and then EBkSP rerouting takes place. The footprints of (v_1, v_2, v_3) in the next time window are (fg, gh, hi, ij), (ab, bg, gh, hi, ij), and (ch, hk), respectively. For v_4, which travels from ab to ij, there are three alternative paths with similar travel times: $p_1(ab, bg, gh, hi, ij)$, $p_2(ab, bc, ch, hi, ij)$, and $p_3(ab, bc, cd, di, ij)$. The union of their segments is set $(ab, bg, gh, hi, ij, bc, ch, cd, di)$, and their weighted footprint counters are $(1, 1, 2, 2, 2, 0, 1, 0, 0)$. Consequently, $N = 11, E_V(p_1) = 2.29, E_V(p_2) = 1.67$ and $E_V(p_3) = 0.53$. Hence, v_4 will be assigned to p_3 because it is the least popular.

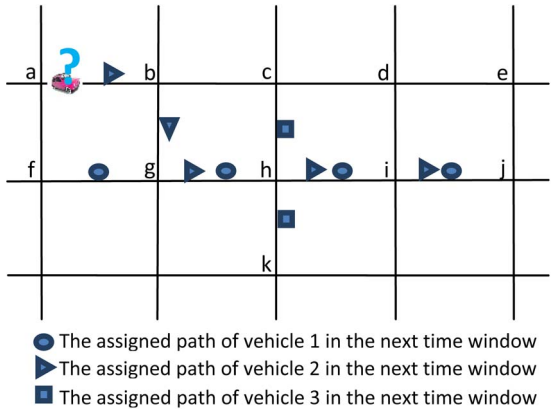


Fig. 3. EBkSP rerouting example. All segments have the same weight.

3) FBkSP*:* The RkSP and the EBkSP distribute the traffic load of the rerouted vehicles by randomly choosing between alternative paths or by balancing the system entropy among multiple paths. Since the key idea is load balancing, an alternative approach that we propose is to directly balance the traffic load, i.e., the weighted footprint counters, through local search optimization [18]. The goal of the local "search" is to find the path assignment in which the sum of the weighted footprint counters is minimal, i.e., to minimize $\sum_{s_i \epsilon S} \mathrm{fc}_{s_i}$ in a network region, where S is the set of all region segments. As we recall from Definition 2, weighted footprint counter fc_i indicates the impact of the traffic flow on road segment r_i (i.e., the possibility of generating future congestion on r_i). Therefore, the summation of the weighted footprint counters of all the road segments measures the risk of congestion of the whole network. In other words, as a weighted footprint counter indicates the future flow magnitude, minimizing the sum of the weighted footprint counters means having balanced flows on all paths, thus reducing the risk of producing congestion.

Fig. 4 shows how the path assignment affects the total number of weighted footprint counters. Assume that, initially, vehicles (v_1, v_2, v_3) are assigned to paths (ab, bc, cd, di, ij), (fg, gh, hi, ij), and (ab, bc, ch), respectively, and that the road segments have different weights [cf. Fig. 4]. Then, the sum of the weighted footprint counters in this network region is 18 [cf. Fig. 4(a)]. However, if v_1 switches to path (ab, bg, gh, hi, ij), the sum of the weighted footprint counters is reduced to 16, as shown in Fig. 4(b). Therefore, the system will select the latter assignment.

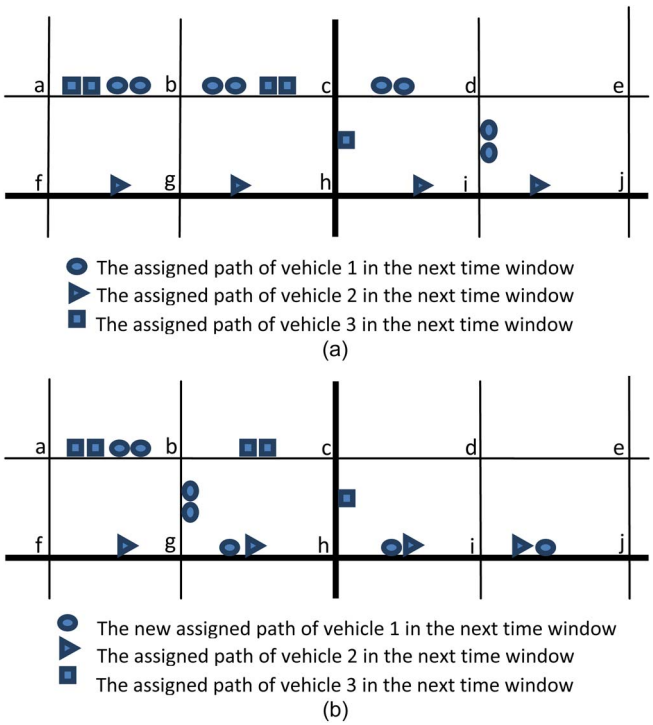


Fig. 4. FBkSP example. $\omega_{fg} = \omega_{gh} = \omega_{hi} = \omega_{ij} = \omega_{ch} = 1$. $\omega_{ab} = \omega_{bc} = \omega_{cd} = \omega_{de} = \omega_{af} = \omega_{bg} = \omega_{di} = \omega_{ej} = 2$. (a) Old assignment. (b) New assignment.

To implement the optimization of the total number of footprints in a road network region, we use a random search strategy (cf. Algorithm 2). The system generates first a good path assignment solution for all selected vehicles by assigning to each vehicle the path with the current least number of footprints (lines 2–8 in Algorithm 2). This initial assignment does not necessarily guarantee the minimum sum of footprint counters of the considered network region, i.e., the union of all segments of the k shortest paths of the rerouted vehicle. Therefore, the system randomly modifies the initial assignment to improve it (lines 14–16). If the new assignment reduces the total number of weighted footprint counters in the network region, the new assignment is accepted (lines 18–19). Otherwise, the assignment is rejected. This process iteratively runs until the limit number of iterations is attained (line 26).

Algorithm 2 Flow-Balanced k-Shortest Path Rerouting

```
1: procedure LocalOptAssign(allkPaths, sortedVehicles) {generate initial solution}
2: for all vehicle in sortedVehicles do
3:     {origin, dest} = getVehicleOD(vehicle)
4:     newpath = pickPath_leastfootprints(allkPaths, origin, dest)
5:     reduction = getReduction()
6:     vehicle.selectedpath = newpath
7:     updateFootprint(vehicle)
8: end for {locally optimize the initial solution}
9: iter = 0
10: repeat
11:     for all vehicle in sortedVehicles do
12:         {origin, dest} = getVehicleOD(vehicle)
13:         newpath = pickpath_random(allkpath, orgin, dest)
14:         newreduction = getReduction(newpath,vehicle.selectedpath)
15:         if newReduction < reduction then
16:             vehicle.selectedpath = newpath
17:             updateFootprint(vehicle)
18:             reduction = newReduction
19:         end if
20:     end for
21:     iter = iter + 1
22: until iter > MaxIteration {MaxIteration is a constant, set as 10 here.}
23: end procedure
```

Disjointness of the k paths: The kSP algorithm used in this paper computes a set of k shortest time paths that are loopless but potentially overlapping. Using k disjoint shortest paths (or paths with a low degree of similarity) does not necessarily improve the rerouting performance of our algorithms. To compute k disjoint shortest paths, a typical algorithm computes first the m shortest paths $(m > k)$, and then selects the k disjoint paths from the set of m paths. Once the computation cost for determining the m paths is paid, EBkSP and FBkSP will perform better over m paths than over a subset of k paths because the total number of road segments that can be used for load balancing is larger. The experimental results presented in Section V-B confirm that increasing the k value improves significantly the effectiveness of the rerouting.

Algorithm 3 The main process

```
1: procedure main
2: while true do
3:     updateEdgeWeights()
4:     congestedRoads = detectCongestion(edgeWeights)
5:     if #congestedRoads > 0 then
6:         for all road in congestionRoads do
7:             selectedVehicles = selectedVehicles ⋃ selectVehicles(road)
8:         end for
9:         sortedVehicles = sortByUrgency(selectedVehicles)
10:        allpaths = Emtpy
11:        if not AR* then
12:            odPairs = updateODPairs(selectedVehicles)
13:            if DSP then
14:                allPaths = Dijkstra(odPairs)
15:            else
16:                allPaths = compuate_all_kShortestPaths(odPairs)
17:            end if
18:            doReroute(allPaths, sortedVehicles)
19:        else
20:            for all vehicle in sortedVehicles do
21:                {origin, dest} = getVehicleOD(vehicle)
22:                newPath = AstarRepulsion(origin, dest)
23:                if newPath is not empty then
24:                    setRoute(vehicle, newPath)
```

```
25:          end if
26:        end for
27:      end if
28:    end if
29:    wait(period){The process executes periodically.}
30: end while
31: end procedure
32: procedure doReroute(allPaths, sortedVehicles)
33: if FBkSP then
34:    LocalOptAssign(allPaths, sortedVehicles)
35: else
36:    for all vehicle in sortedVehicles do
37:      {origin, dest} = getVehicleOD(vehicle)
38:      if DSP then
39:        newPath = allPaths[origin][dest][0]
40:      end if
41:      if RkSP then
42:        newPath = pickPath_random(allPaths[origin][dest])
43:      end if
44:      if EBkSP then
45:        newPath = pickPath_leastPopular(allPaths[origin]
          [dest])
46:        updateFootprint(vehicle, newPath)
47:      end if
48:      setRoute(vehicle, newPath)
49:    end for
50: end if
51: end procedure
```

C. Rerouting Process

Here, we present the global rerouting process on which our traffic guidance system is based on. The process is presented in Algorithm 3. The system periodically looks for signs of congestion in the road network (line 4). If signs of congestion are detected, then the system selects the vehicles situated near the congested road segments (cf. Section III-C) and ranks them based on the urgency function (cf. Section III-D). Finally, alternative routes are computed for the selected vehicles by using one of the five proposed rerouting strategies. It is worth noticing that, except for the AR*, all the other rerouting strategies optimize the alternative path search by grouping the vehicles on their OD (line 10). This can lead to a significant reduction of the computational cost, as shown in Section V-B.

D. DTA

The work on DTA algorithms is essential for the problem that we consider in this paper, i.e., improving the driving travel time through traffic rerouting and guidance. Nevertheless, as explained in Section II, DTA is not yet the most viable solution for real-time traffic guidance mainly because of the DTA's very high computational complexity coupled with the high dynamics of the traffic and the imperfections in traffic knowledge. In spite of this, DTA can offer valuable information, such as the level of improvement in the travel time that can be achieved in an ideal situation (i.e., where computational cost is not an issue, and the traffic information is perfect). Therefore, we employ DTA to obtain a lower bound on the optimization of the travel time for comparison with the results produced by the proposed strategies.

The DTA model that we use in this paper tries to achieve stochastic user equilibrium (SUE) through an iterative simulation process and mathematical modeling (cf. Section II). Given the traffic demand, it chooses some initial routes assuming zero traffic. Then, it calculates the network load and the travel times by simulation and updates the route choices of the drivers. This process is repeated until the travel times are stationary or a maximum number of iterations is reached. The simulation-based DTA tool that we employ was proposed in [14] and [5]. At least three parameters have to be given as input: 1) a road network; 2) a set of trips; and 3) the maximum number of iterations. The higher the number of iterations is, the higher the probability of achieving an SUE traffic state. In our experiments, we defined the maximum number of iterations as 50 since that was the value specified in [8]. The DTA algorithm, as defined in [14], is summarized as follows.

Step 1) Initialize the route of each driver by the optimal route in the empty network.

Step 2) Calculate the time-dependent costs of the road segments by simulation.

Step 3) Recalculate the optimal routes of a certain portion p of the drivers using the time-dependent costs from step 2.

Step 4) If routes have changed in step 3, go to step 2.

Note that the DTA algorithm involves not only the shortest path graph computations but also simulations. The purpose of the simulation is to help DTA acquire a relatively accurate estimation of the travel times, given the assignment of the previous iteration. Then, the estimated travel times are used to adjust the assignment in the succeeding iteration. However, this inevitably leads to an increased computational burden. In comparison, our approach proposes alternative routes to drivers during their entire journey based on the dynamic conditions in the road network, and most of the computation is spent on shortest path graph algorithms. Therefore, we expect our approach to be more efficient than DTA.

V. Evaluation

The main objective of our simulation-based evaluation is to study the performance of the five rerouting strategies under various scenarios. Specifically, we address the following questions.

- Which strategy leads to the most benefits for drivers in terms of travel time and the number of reroutings?
- What is the tradeoff between strategy effectiveness and their efficiency in terms of computation time? How do the proposed strategies compare to a DTA-based approach in terms of effectiveness and efficiency?
- Which strategies scale better with the number of cars?
- How do parameters (number of alternative paths, car selection level, etc.) influence the performance?
- How robust is the system under various compliance rates (i.e., percentage of drivers who follow the guidance) and penetration rates (i.e., percentage of vehicles which have our software)?

TABLE I
STATISTICS OF THE TWO ROAD NETWORKS

	Brooklyn	Newark
Network area	$75.85km^2$	$24.82km^2$
Total number of road segments	551	578
Total length of road segments	$155.55km$	$111.41km$
Total number of intersections	192	195

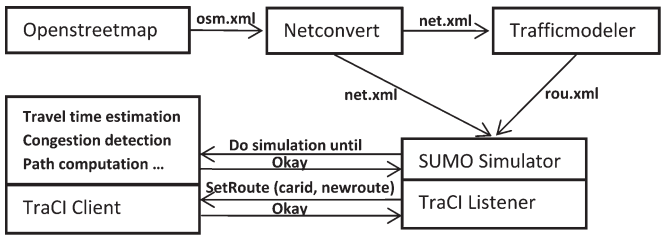


Fig. 5. Simulation process.

A. Simulation Setup

We employed SUMO 15.0 [7] and TraCI [41] for our simulations. SUMO is an open-source highly portable microscopic road traffic simulation package designed to handle large road networks. TraCI is a library providing extensive commands to control the behavior of the simulation, including the vehicle state, the road configuration, and traffic lights. We implemented the rerouting strategy algorithms using TraCI. Essentially, when SUMO is called with the option to use TraCI, SUMO starts up, loads the scenario, and then waits for a command. Thus, variables in the simulation can be changed (e.g., new paths assigned to certain vehicles). Then, a new command can be sent with how many seconds to run the simulation before stopping and waiting for another command.

We downloaded two urban road maps from OpenStreetMap [16] in OSM format. One is a section of Brooklyn, NY, USA, and the other is in Newark, NJ, USA. We use the Netconvert tool in SUMO to convert the maps into a SUMO usable format and the Trafficmodeler tool [31] to generate vehicle trips. Netconvert removes the pedestrian, railroad, and bus routes, and sets up a static traffic light at each intersection to make the simulations more realistic (as the maps do not have stop signs). All roads have the same speed limit (13.9 m/s); some roads have one lane in each direction, whereas others have just one lane based on the specification in the OpenStreetMap OSM file. The statistics of the two networks are shown in Table I. By default, the shortest travel time paths are automatically calculated and assigned to each vehicle at the beginning of simulation based on the speed limit. Fig. 5 shows the simulation process. Fig. 6(a) and (b) show the traffic flow in both networks. We used Trafficmodeler to generate a total of 1000 cars in the Brooklyn network from the left area to the right area in an interval of 1000 s. The origins and the destinations are randomly picked from the left area and the right area, respectively. In the Newark network, 906 cars were generated, having the origins randomly picked from the peripheral road segments and the destinations on the road segments inside the hot-spot circle.

In the simulations, we use the default settings in SUMO 15.0 for vehicle length = 5 m, the minimal gap = 2.5 m, the car following model (Krauss [22]), and the driver's imperfection = 0.5. For each scenario, we average the results over 20 runs. Initially, we assume an ideal scenario in which all drivers have the system and accept the route guidance. We relax these assumptions in the last part of the evaluation. Table II defines the parameters used in our evaluation. We performed extensive experiments to determine the best values for these parameters. Section V-B will show results for the urgency function, level L, and the number of paths k. For the sake of brevity, we do not show results for period, δ, and β. We choose 450 s as the rerouting period and 0.7 as the congestion threshold because they produce good results with moderate computation. We observe that β between [0.05, 0.1] produces good results on both networks for the AR*. Thus, we select 0.05 for all the following experiments.

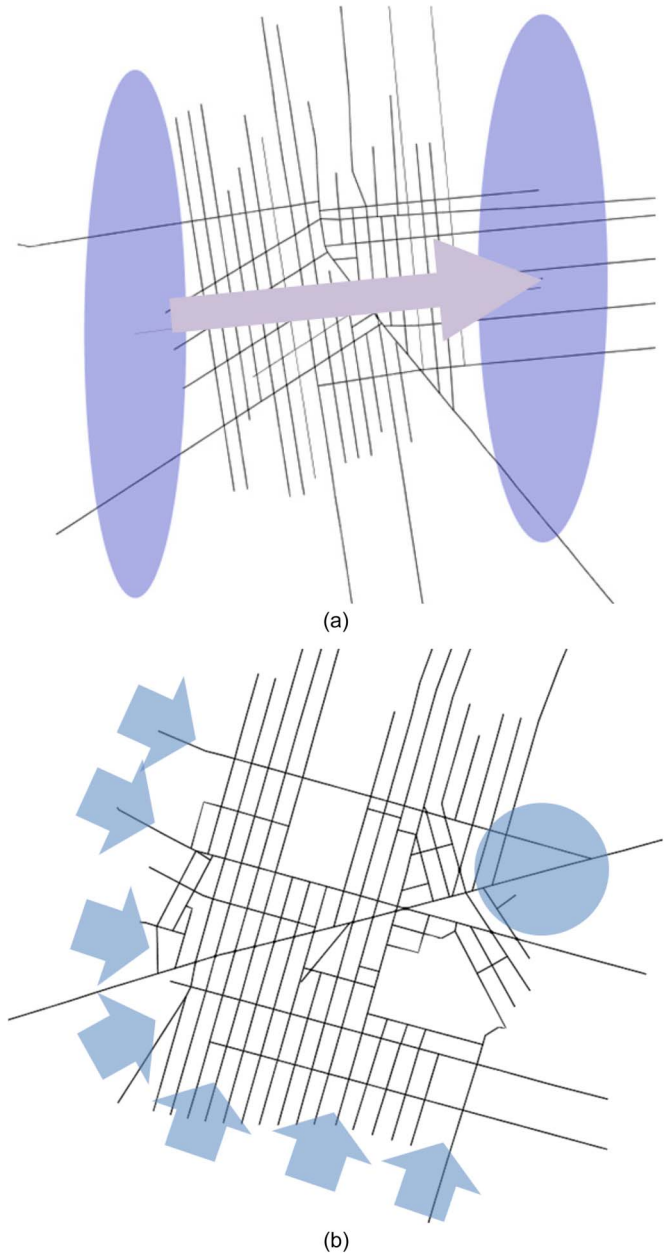


Fig. 6. Traffic flow in the road networks. (a) Brooklyn. (b) Newark.

TABLE II
PARAMETERS USED IN THE EVALUATION

period	The frequency of triggering the re-routing; by default period=450s
threshold δ	Congestion threshold; if $K_i/K_{jam} > \delta$, the road segment is considered congested; by default $\delta = 0.7$
urgency	Urgency policy: RCI or ACI
level L	Network depth to select vehicles for re-routing starting from the congested segment and using BFS on the inverted network graph
# paths k	The max number of alternative paths for each vehicle; by default $k = 4$
repulsion weight β	The weight of repulsion in AR^*; by default $\beta = 0.05$

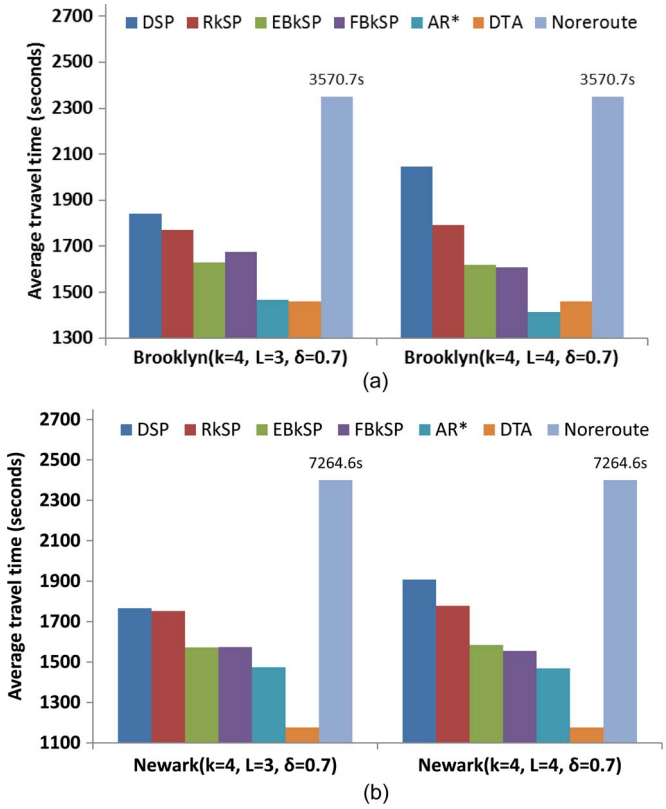


Fig. 7. Average travel time ($L = (3, 4)$, $k = 4$, urgency = ACI, period = 450 s, $\delta = 0.7$, and $\beta = 0.05$). (a) Brooklyn. (b) Newark.

We also implemented a DTA-based rerouting strategy (cf. Section IV-D) by using a DTA tool provided with the SUMO generator.

B. Results and Analysis

Average travel time: Fig. 7 shows the average travel time obtained with the five strategies and with DTA on both networks. The "no-reroute" bars indicate the travel time in the absence of any rerouting. The results show that all the proposed strategies improve the travel time significantly. In most cases, the proposed strategies obtain travel times at least two times lower than no-rerouting. For instance, with a selection level of 3, compared with "no-reroute," EBkSP reduces the travel time by 2.2 times and 4.5 times on Brooklyn and Newark,

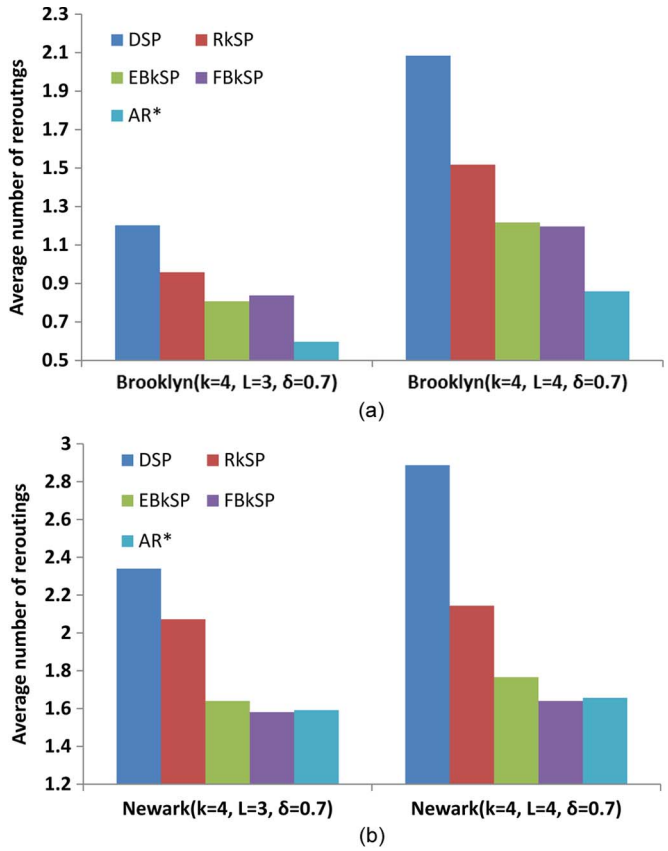


Fig. 8. Average number of reroutings ($L = (3, 4)$, $k = 4$, urgency = ACI, period = 450 s, $\delta = 0.7$, and $\beta = 0.05$). (a) Brooklyn. (b) Newark.

respectively. As expected, DTA has the best average travel time since it can achieve user equilibrium. Based solely on the obtained average travel time, we rank the five strategies as follows: DTA > AR* >(EBkSP, FBkSP)> RkSP > DSP >no-rerouting. The results confirm the hypotheses laid out in Section IV with the statistical significance of a 95% confidence interval. The DSP can improve the travel time because it dynamically reroutes the vehicles by considering the traffic conditions. However, in some cases, e.g., if many vehicles have similar current positions and destinations, respectively, new congestion can be created by the rerouting process. The RkSP avoids this shortcoming since it balances the traffic flow over several paths. Nevertheless, a randomly picked path is not necessarily the best one. The EBkSP and the FBkSP offer even better performance by carefully selecting the path for each rerouted vehicle. Finally, the AR* has the best performance among the proposed strategies as it considers all the other vehicles in the road network in the computation of a new route.

Our experiments also demonstrated that setting the depth level to 3 or 4 is best for selecting a relatively optimal number of vehicles for rerouting (the two values lead to similar performance for Brooklyn, whereas level 3 is better for Newark). Lower level values do not select enough cars, whereas higher values increase the number of reroutings (cf. Fig. 8). Therefore, we set the level parameter to 3 in the remaining experiments.

Average number of reroutings: It is important that the RRF for a given vehicle during a trip stays low. From the driver point of view, changing the path to the destination too

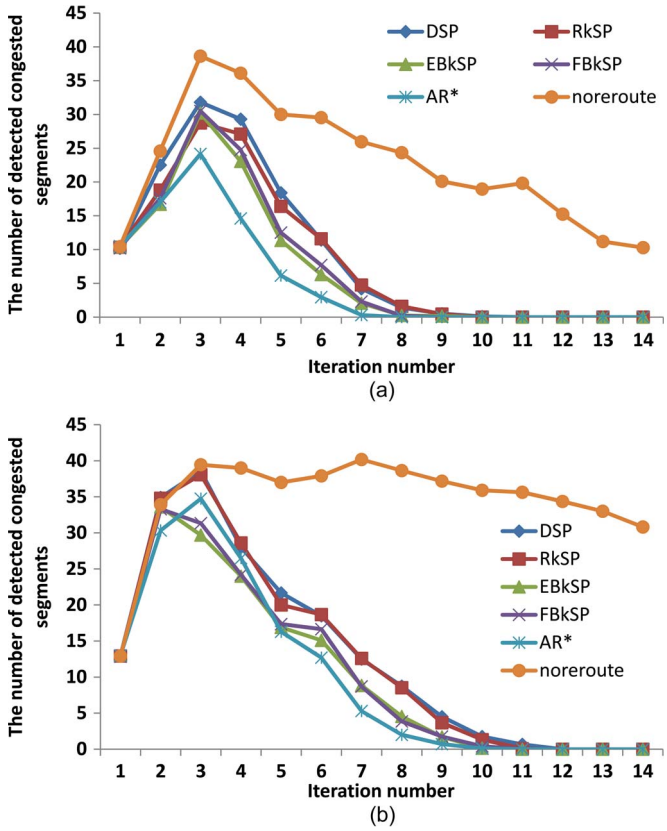


Fig. 9. Number of congested road segments as a function of the number of iterations ($L = 3$, $k = 4$, urgency = ACI, period = 450 s, $\delta = 0.7$, and $\beta = 0.05$). (a) Brooklyn. (b) Newark.

often can be distracting and annoying. From the system point of view, having a low number of reroutings means decreasing the computational burden because the rerouting process is costly. Fig. 8 compares the number of reroutings across the five proposed strategies. In terms of the average number of reroutings, $\text{AR}^* < (\text{EBkSP}, \text{FBkSP}) < \text{RkSP} < \text{DSP}$ with 95% confidence interval.[4] Compared with the DSP, the AR^* reduces the average number of reroutings by up to 2 and 1.5 times, whereas compared with the RkSP, the AR^* is better by 1.6 and 1.3 times on Brooklyn and Newark, respectively. The reason is that, by considering future path information in the rerouting decision, the EBkSP, the FBkSP, and the AR^* can not only mitigate the current congestion but also avoid creating new congestion; hence, there is lower necessity for recurrent rerouting.

To confirm this analysis, we also measured the number of congested segments in each iteration. Fig. 9 shows the results. As traffic is generated during the first 1000 s (i.e., iterations 1–3), the number of congested roads increases for all strategies. Then, the number of congested roads decreases for two reasons. First, no more traffic is generated, and this effect is observed in the "no-rerouting" curve. Second, and more importantly for our strategies, rerouting helps to dramatically reduce this number. As expected, the EBkSP and the FBkSP have comparable results and reduce the number of congested roads faster than the DSP and the RkSP. In addition, we noticed that, although the AR^* did not have the best performance at the beginning of the simulation, it was capable of alleviating congestion much faster than the other methods afterward.

[4]We performed a t test for each pair of strategies.

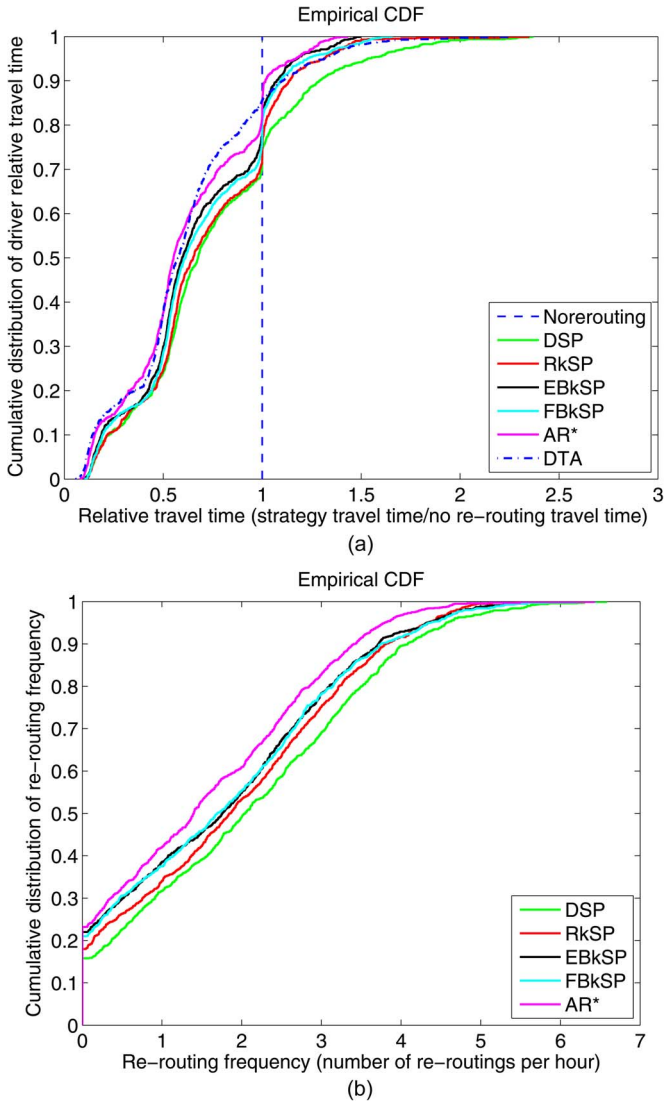


Fig. 10. Cumulative distribution function (cdf) of relative travel time and RRF per hour on the Brooklyn network ($L = 3$, $k = 4$, urgency = ACI, period = 450 s, $\delta = 0.7$, and $\beta = 0.05$). (a) Relative travel time cdf. (b) Rerouting frequency cdf.

Distribution of travel time and RRF: The average travel time and the average number of reroutings measure the performance of the system from a global point of view. Here, we investigate the performance from a driver point of view. How many drivers end up with a shorter travel time? We introduce two new metrics. The first metric is the relative travel time (RelT), which is defined as the ratio of the travel time with rerouting and the travel time with no rerouting; thus, RelT measures the travel time gains or losses for individual drivers. The second metric is RRF, which in this experiment is defined as the number of reroutings per hour experienced by a driver; thus, RRF measures the driver distraction due to rerouting.

Fig. 10(b) shows the cumulative distribution of RelT and RRF for each rerouting strategy for the Brooklyn network.

The values are averages (per driver) computed across 20 runs of simulations. We obtained similar results for the Newark network, which we omit. The AR* has the best results for both RelT and RRF, followed closely by the EBkSP and the FBkSP. The system manages to improve the travel time for a large majority of drivers. Similarly, a large majority of drivers experience no more than three reroutings per hour, which we believe is acceptable in city scenarios with heavy traffic.

However, there is a relatively small percentage of drivers (i.e., ranging from 10% for the AR* to 25% for the DSP) that end up with increased travel time after rerouting. The observed increase is limited to less than 50% for most of these drivers. Note that this phenomenon is equally present in DTA, where around 15% of the drivers have increased travel time. The main reason for these results is that the proposed rerouting strategies have not been designed to achieve user-optimal equilibrium and thus cannot guarantee the best travel time for each user. More surprisingly, even DTA which was designed to achieve user equilibrium cannot do it; our conjecture is that this is due to the difficulty in finding an equilibrium under congestion [9]. We understand that a few bad experiences with the system could impact its adoption rate. Therefore, as future work, we plan to investigate strategies to lower the number of drivers with increased travel time and to bound this increase to low values.

CPU time: Thus far, the results indicate that the AR* produces the best travel times (near to the DTA times), which is followed closely by the EBkSP, the FBkSP, and in some cases, the RkSP. An important question is what is the computational performance among all the proposed five strategies. To answer it, we need to first look at the algorithm complexity for the Dijkstra shortest path (used by the DSP), kSPs (used by the RkSP, the EbkSP, and the FBkSP) and A^* (used by the AR*). The Dijkstra shortest path and kSPs require $O(E + V \log V)$ and $O(kV(E + V \log V))$, respectively, whereas A^* was proven to be faster than Dijkstra [35]. However, this complexity analysis is pertinent only when we consider the selection of an alternative path for one single vehicle. From the system point of view, the global computational complexity also depends on the number of reroutings processed in a time window; this number is a function of the number of congested road segments and the congestion severity (i.e., how many vehicles are selected for rerouting). Additionally, the DSP, the RkSP, the FBkSP, and the EBkSP compute the shortest paths after grouping the vehicles on their OD, whereas the AR* calculates a new path for each vehicle. Therefore, the AR* could require a larger computation time than the other methods.

Fig. 11(a) shows the global CPU time consumed for rerouting by the five methods and by DTA. Note that the experiments were conducted on a 64-bit Ubuntu machine with Intel Core i5-2467M CPU (1.6GHz) and 4 GB of memory. We observe that the DSP requires the least CPU time for rerouting, mainly due to the low complexity of the shortest path algorithm and to grouping the rerouted vehicles. The AR* consumes significantly more CPU time. For example, it requires 2 and 2.3 times more CPU time than the RkSP and the EBkSP on Brooklyn. The main reason is that the AR* cannot group the rerouted vehicles similar to the other methods, as stated in Section IV-A2.

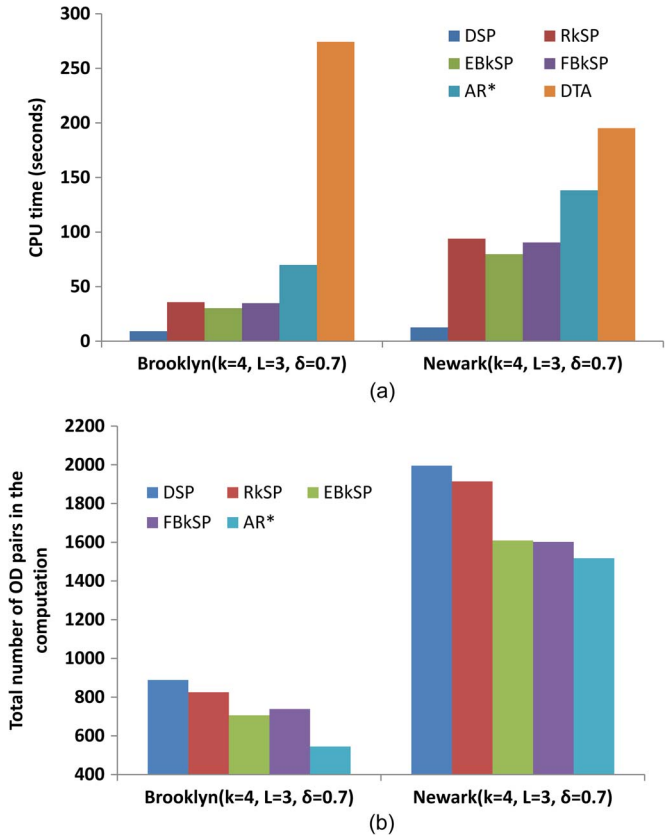


Fig. 11. CPU time and the number of OD pairs for both networks ($L = 3$, $k = 4$, urgency = ACI, period = 450 s, $\delta = 0.7$, and $\beta = 0.05$). (a) CPU time. (b) Number of OD pairs.

The EBkSP, the FBkSP, and the RkSP are situated in between the aforementioned methods from the CPU time point of view. Interestingly, the EBkSP and the FBkSP require less computation time than the RkSP, although they execute more complex path selection algorithms in addition to the kSP computation. The explanation is that the EBkSP and the FBkSP decrease the total number of reroutings processed in a period. This decrease becomes apparent when we look at the number of OD pairs involved in the computation, as indicated in Fig. 11(b). The total number of OD pairs is lower for the EBkSP and the FBkSP than for the RkSP. While the DSP has the largest number of OD pairs, it still has the lowest CPU time because of its much lower computational complexity for path calculation.

DTA has the largest CPU time and scales poorly with an increasing number of vehicles (in terms of CPU time) when compared with the AR* or the other proposed methods [cf. Fig. 12(b)]. In addition, it is worth noticing that DTA assumes all vehicles in the system known at the beginning (i.e., when it computes its routes). However, in real life, vehicles may appear at any time, and DTA would be required to perform its expensive computation over and over again. Therefore, due to its very high computational cost in real life, DTA may be impractical (i.e., it may not be able to compute alternative routes fast enough to mitigate congestion).

In conclusion, if we consider both the travel time and the CPU time, the EBkSP and the FBkSP appear to be the best strategies since they offer the best tradeoff between rerouting

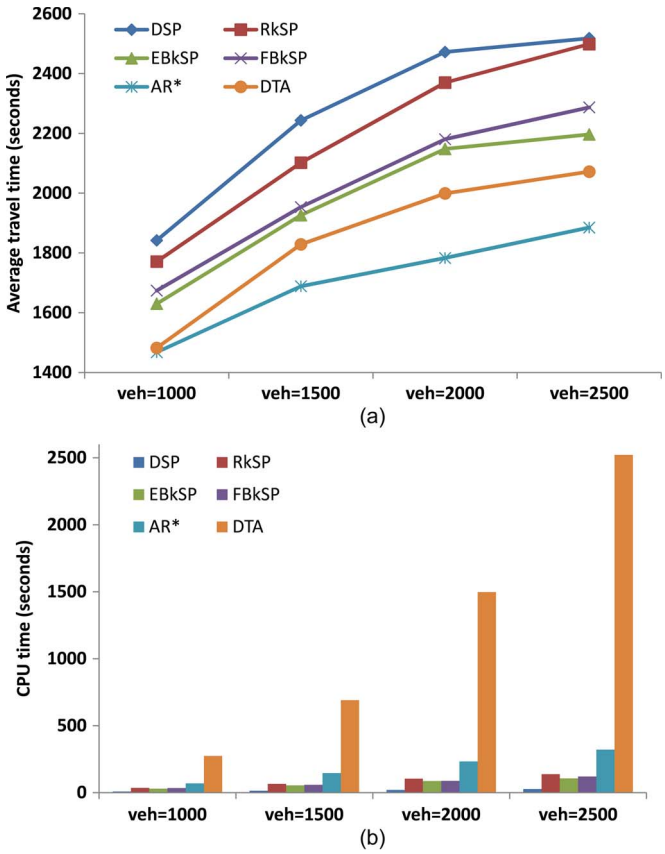


Fig. 12. Average travel time and CPU time for the Brooklyn network for different traffic densities ($L = 3$, $k = 4$, urgency = ACI, period = 450 s, $\delta = 0.7$, and $\beta = 0.05$). (a) Average travel time. (b) CPU time.

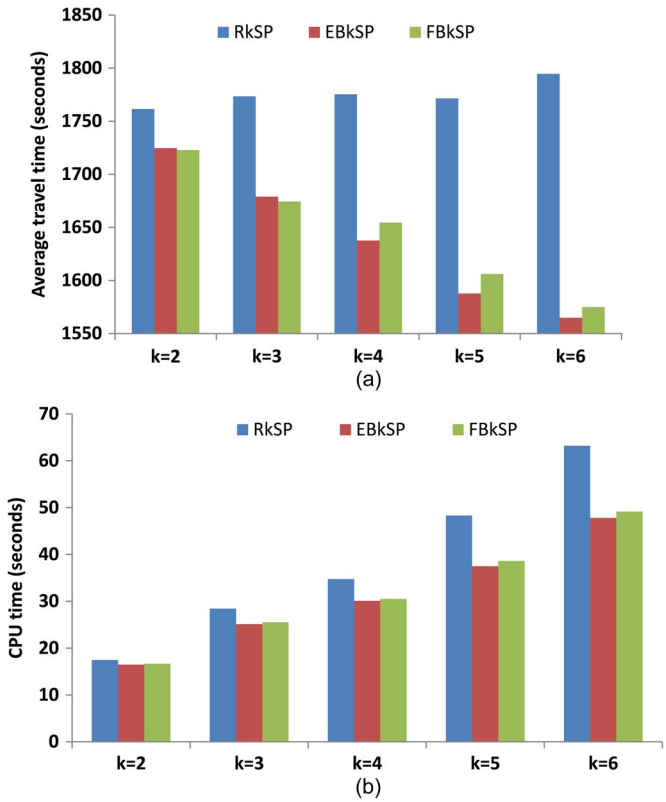


Fig. 13. Average travel time and CPU time for the RkSP, the EBkSP, and the FBkSP as a function of k for the Brooklyn network ($L = 3$, $k = (2, 3, 4, 5, 6)$, urgency = ACI, period = 450 s, and $\delta = 0.7$). (a) Average travel time. (b) CPU time.

effectiveness and computational efficiency. If computational cost is not an issue, one can use the AR* strategy, whereas in the opposite case, DSP is the most appropriate choice.

Traffic density: The results presented up to here already offer a good idea about the capabilities of the proposed rerouting strategies to alleviate traffic congestion. However, there is an important aspect that still needs to be explored, i.e., how the proposed methods scale with the increase in the traffic volume. To respond to this question, we conducted another set of experiments on the Brooklyn network, where we increased the number of vehicles from 1000 to 2500. Fig. 12 shows the obtained results both for the average travel time and the CPU time for different traffic densities. The AR* and DTA present the best scalability from the average travel time point of view. However, these methods are also the least scalable from the CPU time point of view. As we can see, DTA exhibits particularly poor scalability compared with the proposed strategies, confirming our hypothesis that DTA is not yet a suitable approach to real-time traffic management. In addition, somewhat interestingly, the AR* obtained better average travel times than DTA [cf. Fig. 12(a)] when the number of vehicles was above 1500. This is certainly due to the fact that the 50 iterations limit that we set in the DTA tool is not sufficient to achieve user equilibrium for higher traffic densities. Therefore, a higher number of iteration is needed in this case, which will evidently lead to even higher CPU times.

Number of alternative paths: k is a determinant parameter for the performance of the RkSP, the FBkSP, and the EBkSP, which require kSP computation. A larger k value allows for better traffic balancing but introduces higher computational complexity. Furthermore, the maximum allowed difference between the slowest path and the fastest path is 20% in our setting. Therefore, large k values may not be necessary because they would lead to computing many useless paths. Fig. 13 compares the performance of the RkSP, the EBkSP, and the FBkSP with different k values on the Brooklyn network. The k value is irrelevant for the DSP and the AR*.

We observe that the RkSP does not exhibit any performance improvement for $k > 2$, whereas both the EBkSP and the FBkSP consistently produce lower travel times with higher k values. Fig. 13(b) shows that the computational cost linearly increases with k for all the kSP methods. However, the EBkSP and the FBkSP are more scalable than the RkSP, particularly for larger k values (e.g., the EBkSP requires 32% less CPU time than the RkSP when k equals 6). The efficiency of the EBkSP and the FBkSP is due to the reduction of the number of OD pairs.

Urgency function: Among the five proposed algorithms, the EBkSP, the FBkSP, and the AR* use an urgency function to sort the list of vehicles selected for rerouting (cf. Section III-D). To measure the performance difference between the two proposed ranking policies, i.e., RCI and ACI (cf. Definition 1), we conducted the ANOVA statistics test over the average travel time from 30 simulations with the EBkSP and the FBkSP. The

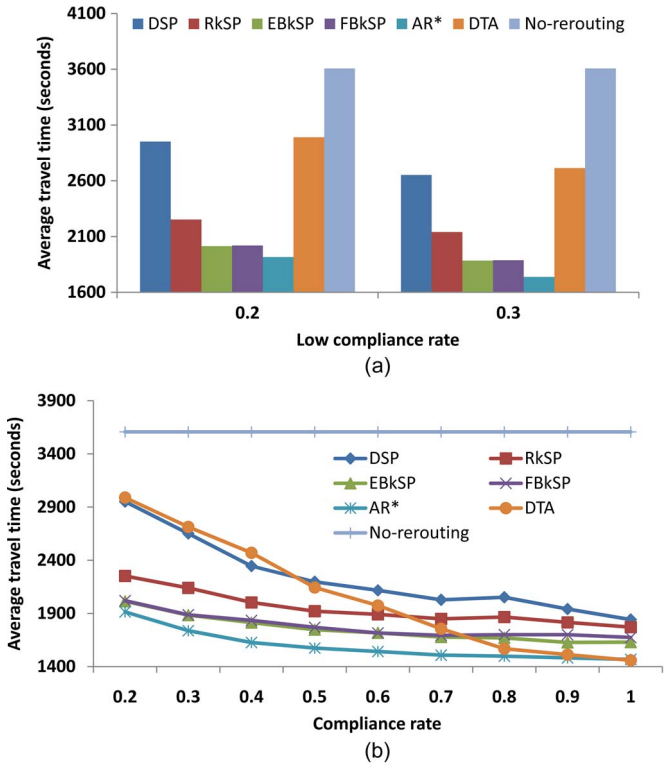


Fig. 14. Average travel time as a function of the compliance rate on the Brooklyn network ($L = 3$, $k = 4$, urgency = ACI, period = 450 s, $\delta = 0.7$, and $\beta = 0.05$). (a) Low compliance rate. (b) All compliance rates.

results show that ACI produces lower average travel times than RCI ($p < 0.01$) with a 95% confidence interval. The result confirmed our previous analysis in Section III-D. Thus, we used ACI as our default urgency function in all the experiments.

Compliance rate: It is unrealistic to assume that every driver follows the rerouting guidance. The drivers' compliance rate is an important factor for the rerouting strategy design. Therefore, we measured the average travel time while varying the compliance rate for the five proposed strategies and for DTA. For our strategies, a compliance rate of $x\%$ means that each of the selected vehicles switches to the new route with $x\%$ probability during each rerouting period. For DTA, $x\%$ of the vehicles are randomly selected to follow the DTA assigned route, whereas the remainder follow their initial shortest time route.

Fig. 14(a) indicates that the average travel time can be significantly improved by all five strategies even under low compliance rates. This is due to the fact that, even under low compliance rates, the drivers who comply with the guidance can still receive more rapid routes, which in turn can improve the travel time for the remainder of the drivers. Fig. 14(b) shows the average travel time for a wide range of compliance rates.

In particular, when the compliance rate is low, the RkSP, the EBkSP, the FBkSP, and the AR* show significantly better travel times than DTA. The reason for this is that, when compliance is low, the drivers who comply benefit much more from our guidance than from the DTA guidance. In the DTA approach, the route computation is done once before any vehicle enters the network. If the compliance rate is low, the DTA-computed routes are far from a user equilibrium, inclusively for the compliant drivers. Conversely, our strategies can periodically adjust the vehicles' routes based on the current traffic information. Therefore, although the noncompliant drivers create congestion in the network, the compliant drivers can still receive fairly good paths, which implicitly reduces the congestion level in the network.

Penetration rate: To understand how easy it is to deploy our solution in real life, we study the effect of the penetration rate on the average travel time. Specifically, each vehicle has $x\%$ probability of participating in the system, i.e., providing positioning information and receiving guidance. Hence, the system sees and guides only these $x\%$ of the vehicles. The accuracy of congestion detection and road travel time estimation directly depends on this parameter. Additionally, similar to the compliance rate, the effectiveness of the load balancing mechanism implemented by the rerouting strategies increases with the percentage of vehicles that use the system.

In this set of experiments, we consider two cases. Only the vehicles equipped with the system provide traffic information (which we call "penetration rate without sensors"), and both vehicles equipped with the system and roadside sensors provide traffic information (which we call "penetration rate with sensors"). For clarity, we list in Table III the main differences between these two cases and the experiments for compliance rate.

Fig. 15(a) shows the average travel time for various penetration rates, when traffic data are collected only from vehicles (i.e., no support from roadside sensors). When the penetration rate is low, the performance of the proposed methods is the same as "no-reroute." In this case, the service does not have enough data to accurately detect signs of congestion. Once the penetration rate is greater than 0.4, the system is able to improve the travel time. For penetration rates above 0.6, the EBkSP, the FBkSP, and the AR* start to perform better than DSP and RkSP since a larger number of vehicles are rerouted, which requires a more advanced load balancing mechanism. Compared with our methods, DTA performs better under low penetration rates since the DTA rerouting is not triggered by the congestion detection, as in our approach.

To boost the adoption of the system, we believe that data from roadside sensors (in conjunction with data from vehicles) can be leveraged to detect congestion more accurately in the case of low penetration rates. When road sensors are present, the road traffic density can be measured, the congestion can be detected, and the travel time can be estimated more accurately. Fig. 15(b) demonstrates that most of our methods can significantly improve the travel time even under low penetration rates if roadside sensor information is available. Furthermore, the EBkSP, the FBkSP, and the AR* perform better than DTA in this case. When the penetration rate is low ($x\%$), DTA distributes evenly the $x\%$ vehicles without considering the rest, which can still create congestion. Therefore, the alternative routes proposed by DTA are not as effective in alleviating congestion. Our strategies can take advantage of the sensor information to divert the $x\%$of the drivers and to reduce congestion.

Summary of the experimental results: The objectives of the experiments were threefold. First, we measured the effectiveness of the proposed methods to alleviate congestion

TABLE III
COMPARISON BETWEEN COMPLIANCE AND PENETRATION RATE

	Road traffic density	Congestion detection	Re-routing	Load balancing
Compliance rate	Accurate	Accurate	All vehicles provide positioning information. At each re-routing, only $x\%$ of vehicles follow the guidance	Considers all vehicles (except DSP)
Penetration rate without sensors	Inaccurate	Inaccurate	Predefined $x\%$ of vehicles that provide positioning information receive guidance and comply with the guidance	Considers only the $x\%$ of vehicles (except DSP)
Penetration rate with sensors	Accurate	Accurate	Predefined $x\%$ of vehicles that receive guidance and comply with the guidance. The system obtains accurate traffic data from sensors and vehicles	Considers only the $x\%$ of vehicles (except DSP)

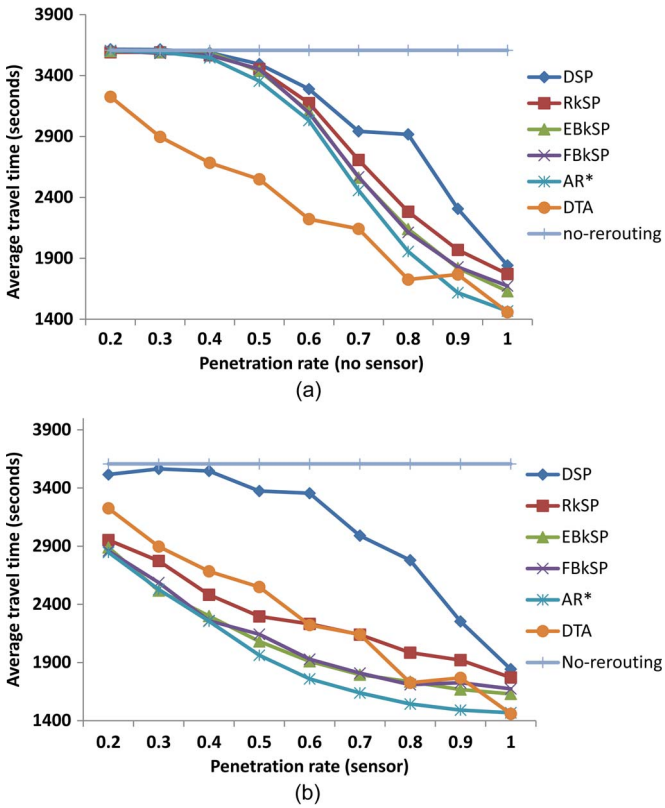


Fig. 15. Average travel time as a function of the penetration rate on the Brooklyn network ($L = 3$, $k = 4$, urgency = ACI, period = 450 s, $\delta = 0.7$, and $\beta = 0.05$). (a) Sensors: vehicles only. (b) Sensors: vehicles and roadside.

and to improve the drivers' travel time, their computational efficiency in achieving this goal, and their scalability with an increasing number of vehicles. To have a more precise idea of the performance level offered by the proposed methods, we used a state-of-the-art DTA tool as a baseline. The experimental results show that our methods are very effective in fighting congestion, being capable of improving the travel time as much as a DTA approach. In addition, our methods are (much) more computationally efficient and scalable than DTA, which makes them more appropriate for use in real-life applications.

TABLE IV
COMPARISON OF ALL THE FIVE STRATEGIES

Effectiveness	AR* >(EBkSP, FBkSP)>RkSP>DSP
Efficiency	DSP>(EBkSP, FBkSP)>RkSP>AR*
Re-routing frequency	AR* >(EBkSP, FBkSP)>RkSP>DSP
Robustness	(AR*, EBkSP, FBkSP)>RkSP>DSP

Furthermore, the palette of proposed methods offers a wide range of choices, having different tradeoffs between effectiveness and efficiency, which responds to a large variety of real-life scenarios.

Second, we conducted a more in-depth set of experiments to understand how the parameters used by our methods influence their performance. The experimental results give a good indication as to which are the most suitable values for some of these parameters (i.e., urgency function, rerouting period, and congestion threshold) in conjunction with the employed method. For other parameters (i.e., number of alternative paths k) and algorithms (i.e., EBkSP and FBkSP), it is still a matter of choice between effectiveness and efficiency, opening the possibility to even finer system tuning.

Third, we considered a more realistic scenario and assessed the robustness of our system under compliance and penetration rates below 100%. The results indicate that our system can still significantly improve the travel time even under low compliance rates and that our approach is more robust than DTA. In addition, if accurate traffic data (e.g., collected by roadside sensors) can be provided to the system, then our strategies exhibit good robustness with various penetration rates.

To summarize the performance of the proposed strategies, we rank them according to four criteria in Table IV, where $A > B$ means that strategy A is better than strategy B. While each method has its own advantages and shortcomings, the experimental results made us conclude that the entropy method (EBkSP) and the local optimization method (FBkSP), which led to very similar results, should be the preferred strategies as they offer the best tradeoff between performance and computation cost.

VI. Conclusion and Future Work

Heartened by the ubiquitousness of mobile devices, such as smartphones or onboard vehicle units, this paper has envisioned a novel approach to tackling the ever more stringent problem of traffic congestion. This approach is based on a traffic guidance system that monitors traffic and proactively pushes individually tailored rerouting guidance to vehicles when there are signs of congestion. The system is responsible for several functions, such as traffic data representation, congestion prediction, and selection of the vehicles to be rerouted. We have chosen to focus in this paper on a key element of our rerouting system, i.e., rerouting strategies. We proposed five rerouting strategies to compute alternative routes for vehicles. Then, we conducted an extensive set of simulation-based experiments to validate our approach. The results showed that the proposed rerouting algorithms are very effective in mitigating congestion and adapt well to the dynamic nature of the traffic, being also more efficient and scalable than existing approaches. In addition, our traffic guidance system remains useful even with a low compliance rate and a moderate penetration rate. The experiments also demonstrated how the performance can be tuned by varying parameters, such as rerouting method, number of alternative paths, and density threshold.

As future work, we plan to explore three directions. First, we will design an adaptive approach to vehicle selection that considers additional parameters, such as road segment length, measured compliance rate, and estimated penetration rate. The objective is to obtain a (fully) self-tunable system capable of automatically adjusting the system parameters depending on the traffic conditions. Second, we will study methods to reduce the number of drivers who do not benefit from rerouting and to limit the increase in their travel time. Third, we intend to investigate a hybrid architecture that off-loads parts of the computation and decision process in the network and uses vehicle-to-vehicle communication to better balance the need for privacy, scalability, and a low overhead with the main goal of low average travel time.

References

[1] [Online]. Available: http://www.google.com/mobile

[2] [Online]. Available: http://www.tomtom.com/en_gb/products/mobile_navigation

[3] [Online]. Available: http://www.inrix.com

[4] [Online]. Available: http://www.autobahn.nrw.de

[5] [Online]. Available: http://sumo.sourceforge.net/doc/current/docs/userdoc/Tools/Assign.html

[6] J. H. Banks, *Introduction to Transportation Engineering*. New York, NY, USA: McGraw-Hill, 2002.

[7] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo-Simulation of urban mobility: An overview," in *Proc. 3rd Int. Conf. Adv. Syst. SIMUL*, Barcelona, Spain, 2011, pp. 63–68.

[8] M. Behrisch, D. Krajzewicz, and Y. P. Wang, "Comparing performance and quality of traffic assignment techniques for microscopic road traffic simulations," in *Proc. DTA*, Leuven, Belgium, 2008.

[9] Y. C. Chiu, J. Bottom, M. Mahut, A. Paz, R. Balakrishna, T. Waller, and J. Hicks, "Dynamic traffic assignment: A primer," Transp. Res. E-Circular, Washington, DC, USA, (E-C153), 2011.

[10] J. Cranshaw, E. Toch, J. Hong, A. Kittur, and N. Sadeh, "Bridging the gap between physical location and online social networks," in *Proc. 12th ACM Int. Conf. Ubiquitous Comput.*, 2010, pp. 119–128.

[11] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: Using a mobile sensor network for road surface monitoring," in *Proc. 6th Int. Conf. MobiSys*, 2008, pp. 29–39.

[12] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM*, vol. 34, no. 3, pp. 596–615, Jul. 1987.

[13] T. L. Friesz, D. Bernstein, T. E. Smith, R. L. Tobin, and B. W. Wie, "A variational inequality formulation of the dynamic network user equilibrium problem," *Oper. Res.*, vol. 41, no. 1, pp. 179–191, Jan./Feb. 1993.

[14] C. Gawron, "Simulation-based traffic assignment–Computing user equilibria in large street networks," Ph.D. dissertation, Math.-Naturwissenschaftlichen Fakultät, Univ. Cologne, Cologne, Germany, 1999.

[15] B. George, S. Kim, and S. Shekhar, "Spatio-temporal network databases and routing algorithms: A summary of results," in *Proc. Adv. Spatial Temporal Databases*, 2007, pp. 460–477.

[16] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Comput.*, vol. 7, no. 4, pp. 12–18, Oct.–Dec. 2008.

[17] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[18] H. H. Hoos and T. Stützle, *Stochastic Local Search: Foundations and Applications*. San Mateo, CA, USA: Morgan Kaufmann, 2005.

[19] E. Horvitz, J. Apacible, R. Sarin, and L. Liao, "Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service," in *Proc. 21st Conf. UAI*, 2005, pp. 275–283.

[20] T. Hunter, R. Herring, P. Abbeel, and A. Bayen, "Path and travel time inference from GPS probe vehicle data," in *Proc. NIPS Anal. Netw. Learn. Graphs*, 2009, pp. 1–8.

[21] B. S. Kerner, "Optimum principle for a vehicular traffic network: Minimum probability of congestion," *J. Phys. A, Math. Theor.*, vol. 44, no. 9, p. 092001, Mar. 2011.

[22] S. Krauss, P. Wagner, and C. Gawron, "Metastable states in a microscopic model of traffic flow," *Phys. Rev. E*, vol. 55, no. 5, pp. 5597–5602, May 1997.

[23] E. L. Lawler, "A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem," *Manage. Sci.*, vol. 18, no. 7, pp. 401–405, Mar. 1972.

[24] S. Maerivoet, "Modelling traffic on motorways: State-of-the-art, numerical data analysis, and dynamic traffic assignment," Ph.D. dissertation, Dept. Elect. Eng., Katholieke Univ. Leuven, Leuven, Belgium, 2006.

[25] H. S. Mahmassani, T. -Y. Hu, and R. Jayakrishnan, "Dynamic traffic assignment and simulation for advanced network informatics (dynasmart)," in *Proc. 2nd Int. CAPRI Semin. Urban Traffic Netw.*, Capri, Italy, 1992.

[26] N. Malviya, S. Madden, and A. Bhattacharya, "A continuous query system for dynamic route planning," in *Proc. IEEE 27th ICDE*, 2011, pp. 792–803.

[27] *Highway Capacity Manual*, Transp. Res. Board, Washington, DC, USA, 2000.

[28] D. K. Merchant and G. L. Nemhauser, "A model and an algorithm for the dynamic traffic assignment problems," *Transp. Sci.*, vol. 12, no. 3, pp. 183–199, 1978.

[29] D. K. Merchant and G. L. Nemhauser, "Optimality conditions for a dynamic traffic assignment model," *Transp. Sci.*, vol. 12, no. 3, pp. 200–207, Aug. 1978.

[30] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in *Proc. 6th ACM Conf. Embedded Netw. Sensor Syst.*, 2008, pp. 323–336.

[31] L. G. Papaleondiou and M. D. Dikaiakos, "Trafficmodeler: A graphical tool for programming microscopic traffic simulators through high-level abstractions," in *Proc. 69th IEEE VTC Spring*, 2009, pp. 1–5.

[32] S. Peeta and A. K. Ziliaskopoulos, "Foundations of dynamic traffic assignment: The past, the present and the future," *Netw. Spatial Econ.*, vol. 1, no. 3/4, pp. 233–265, Sep. 2001.

[33] H. Prothmann, H. Schmeck, S. Tomforde, J. Lyda, J. Hahner, C. Muller-Schloer, and J. Branke, "Decentralized route guidance in organic traffic control," in *Proc. 5th IEEE Int. Conf. SASO*, 2011, pp. 219–220.

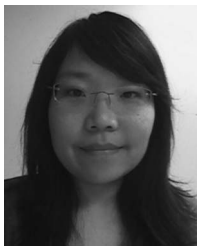
[34] D. Schrank, T. Lomax, and S. Turner, "Urban mobility report," Texas A&M Transp. Inst., College Station, TX, USA, 2011.

[35] D. Schultes, "Route planning in road networks," Ph.D. dissertation, Fakultät für Inf., Inst. für Theor. Inf., Algorithmik II, Univ. Karlsruhe (TH), Karlsruhe, Germany, 2008.

[36] S. Senge and H. Wedde, "Bee inspired online vehicle routing in large traffic systems," in *Proc. 2nd Int. Conf. ADAPTIVE*, 2010, pp. 78–83.

[37] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 5, no. 1, pp. 3–55, Jan. 2001.

[38] B. Tatomir, S. Fitrianie, M. Paltanea, and L. Rothkrantz, "Dynamic routing in traffic networks and MANETs using ant based algorithms," in *Proc. 7th Int. Conf. Artif. Evol.*, Lille, France, Oct. 2005.
[39] N. B Taylor, "CONTRAM 5, an enhanced traffic assignment model," Transp. Road Res. Lab., Wokingham, U.K., Res. Rep., 1990.
[40] J. G. Wardrop, "Some theoretical aspects of road traffic research," *Proc. Inst. Civil Eng.*, vol. 1, Part II, no. 36, pp. 252–378, Jan. 1952.
[41] A. Wegener, M. Piórkowski, M. Raya, H. J. Hellbrück, S. Fischer, and J.-P. Hubaux, "TraCI: An interface for coupling road traffic and network simulators," in *Proc. 11th Commun. Netw. Simul. Symp.*, 2008, pp. 155–163.
[42] D. B. Work, O. P. Tossavainen, S. Blandin, A. M. Bayen, T. Iwuchukwu, and K. Tracton, "An ensemble kalman filtering approach to highway traffic estimation using GPS enabled mobile devices," in *Proc. 47th IEEE Conf. Decis. Control*, 2008, pp. 5062–5068.
[43] T. Xu and Y. Cai, "Feeling-based location privacy protection for location-based services," in *Proc. 16th ACM Conf. Comput. Commun. Security*, 2009, pp. 348–357.



Juan (Susan) Pan received the M.S. degree in computer science from Tianjin University, Tianjin, China, in 2006. She is currently working toward the Ph.D. degree with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ, USA.

Her research interests include vehicular and ubiquitous computing, distributed systems, and social network analysis.

Iulian Sandu Popa received the B.S. degree from the University Politehnica of Bucharest, Bucharest, Romania, in 2005 and the Ph.D. degree from the University of Versailles Saint-Quentin-en-Yvelines (UVSQ), Versailles, France, in 2009, both in computer science.

Since 2012, he has been an Assistant Professor of computer science with the UVSQ and a member of the Secured and Mobile Information Systems Team, Inria Paris-Rocquencourt, Le Chesnay, France. His main research interests include embedded database management systems, spatiotemporal databases, and mobile data management and systems, with a particular focus on topics revolving around privacy and personal data management.

Karine Zeitouni received the Ph.D. degree in computer science from the Pierre and Marie Curie University, Paris, France, in 1991.

She is a Professor of computer science with the University of Versailles-Saint-Quentin-en-Yvelines, Versailles, France. She is the author of around 70 referred articles in journals and conference proceedings. Her main research interests include spatiotemporal databases and knowledge extraction, with a focus on applications in the fields of transport, environment, and health.

Dr. Zeitouni regularly serves at conferences of the (spatial) database community as a member of program committees (e.g., Association for Computing Machinery (ACM) Special Interest Group on Spatial Information, the International Symposium on Spatial and Temporal Databases, and the ACM Special Interest Group on Management of Data).

Cristian Borcea (M'04) received the Ph.D. degree from Rutgers University, New Brunswick, NJ, USA, in 2004.

He is currently an Associate Professor with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ, USA. He is also a Visiting Associate Professor with the National Institute of Informatics, Tokyo, Japan. His research interests include mobile computing and sensing, ad hoc and vehicular networks, and distributed systems.

Dr. Borcea is a member of the Association for Computing Machinery and USENIX.