Proactive vehicle re-routing strategies for congestion avoidance

Juan (Susan) Pan*, Mohammad A. Khan*, Iulian Sandu Popa†, Karine Zeitouni† and Cristian Borcea*
*Department of Computer Science
New Jersey Institute of Technology, University Heights, Newark, New Jersey 07102, USA
Email: jp238@njit.edu, mak43@njit.edu, borcea@cs.njit.edu
†Department of Computer Science
University of Versailles Saint-Quentin-en-Yvelines, France
Email: iulian.sandu-popa@prism.uvsq.fr, karine.zeitouni@prism.uvsq.fr

***Abstract*—Traffic congestion causes driver frustration and costs billions of dollars annually in lost time and fuel consumption. This paper presents three traffic re-routing strategies designed to be incorporated in a cost-effective and easily deployable vehicular traffic guidance system that reduces the effect of traffic congestions. This system collects real-time traffic data from vehicles and road-side sensors and computes proactive, individually-tailored re-routing guidance which is pushed to vehicles when signs of congestion are observed on their route.**

Extensive simulation results over two urban road networks show that all three strategies, namely multipath load balancing considering future vehicle positions (EBkSP), random multipath load balancing (RkSP), and dynamic shortest path (DSP), significantly decrease the average travel time. EBkSP is the best, with as much as 104% improvement compared to the "no re-routing" baseline. Additionally, it lowers with 34% the re-routing frequency compared to the other strategies. Finally, all strategies offer good improvements even when many drivers ignore the guidance or when the system adoption rate is relatively low.

I. Introduction

In 2010, traffic congestion caused urban Americans to travel 4.8 billion hours more and to purchase an extra 1.9 billion gallons of fuel for a congestion cost of $101 billion. It is predicted that by 2015, this cost will rise to $133 billion (i.e., more than $900 for every commuter) and the amount of wasted fuel will jump to 2.5 billion gallons (i.e., enough to fill more than 275,000 gasoline tanker trucks) [1]. While congestion is largely thought of as a big city problem, delays are becoming increasingly common in small cities and some rural areas as well. Hence, finding effective solutions for congestion mitigation at reasonable costs is becoming a stringent problem.

Static distributed road-side sensors (e.g., induction loops, video cameras) and vehicles acting as mobile sensors (i.e., using embedded vehicular systems or smart phones) can collect real-time data to monitor the traffic at fine granularity. For example, the Mobile Millennium project [2] demonstrated that only a low percentage of drivers need to provide data to achieve an accurate traffic view. Unlike a large amount of research which focuses on accurately collecting and calibrating the data (e.g., predicting travel times in specific areas at particular times of day or improving the traffic light cycles) [3], we focus on alleviating congestion by providing drivers with better alternative routes in real-time. Implicitly, fuel consumption and pollution will be reduced as well.

Recently, companies such as Google and Microsoft have started to use infrastructure-based traffic information to compute traffic-aware shortest routes. These solutions are better than just showing the current traffic conditions to drivers because they can guide them as function of other factors such as historic traffic and current weather. However, these solutions do not try to avoid congestions explicitly (i.e., they are reactive solutions) and provide the same guidance for all vehicles on the road at a certain moment as function of their destination (i.e., pull model in which drivers query for the shortest route to destination). Therefore, similar to route oscillations 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 paper presents three traffic re-routing strategies designed to be incorporated in a cost-effective and easily deployable vehicular traffic guidance system that reduces the effect of traffic congestions. In this system, vehicles can be viewed as both mobile sensors (i.e., collect real-time traffic data) and actuators (i.e., change their path in response to newly received guidance). The system is cost-effective and easily deployable because it does not require road-side infrastructure; it can work using only smart phones carried by drivers. Where road-side 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. Periodically, the system evaluates the congestion levels in the road network. When signs of congestions are observed on certain road segments, it computes proactive, individually-tailored re-routing guidance which is pushed to vehicles which would pass through the congested segments. It is important to notice that we do not "force" the drivers to follow alternative routes: the guidance may or may not be accepted by drivers.

The three re-routing strategies that we propose are: (1) dynamic shortest path (DSP), which assigns to each vehicle the current shortest path to destination; (2) random multipath load balancing (RkSP), which computes k shortest paths for

each vehicle and randomly assigns the vehicle to one of them; and (3) multipath load balancing considering future vehicle positions (EBkSP), which computes k shortest paths for each vehicle and assigns the vehicle to the path with the lowest popularity as defined by the path entropy. Intuitively, DSP may move congestion from one spot to another if many drivers share the same destinations and current positions. RkSP uses multiple paths to provide for better traffic flow distribution, but its random nature is far from optimal. EBkSP uses multiple paths as well, but it is expected to provide better load balancing because it estimates the future load on each path.

We extensively evaluated these strategies through simulations over two medium-size urban road networks and across several parameters including the number of alternative paths, the re-routing period, the congestion threshold, the vehicle selection level, the vehicle priority, and the driver compliance rate. The results show that all three strategies significantly decrease the average travel time. EBkSP is the best, with as much as 104% improvement compared to the "no re-routing" baseline. Additionally, it lowers with 34% the re-routing frequency compared to the other strategies; this is important because too frequent re-routings may annoy drivers who can subsequently ignore all guidance. Among the three strategies, DSP has the best computation time. Nevertheless, EBkSP will be the preferred strategy in our system due to its better average travel time and re-routing frequency. Finally, all strategies offer good improvements even for low compliance rate (i.e., drivers ignore the guidance) and relatively low penetration rate (i.e., ratio of vehicles who adopt our system).

The rest of this paper is organized as follows. Section II discusses related work. Section III describes the system model and our assumptions. Section IV presents the three re-routing strategies. Section V shows our simulation results and analysis. Conclusions and future work are presented in Section VI.

II. Related Work

Services such as INRIX [4] provide real-time traffic information at a certain temporal accuracy and allow drivers to receive travel time estimations for alternative routes. According to Wardrop's first traffic equilibrium principle [5], such services could lead to a user-optimum traffic equilibrium. It is known, however, that no true equilibrium can be found under congestion [6]. Even more important, the usefulness of such services is limited by their reactive nature: they cannot avoid congestions. System 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. Unfortunately, this is still not enough due to non-recurring congestions, which represent over 50% of all congestions [7]. Our solution moves one step forward by providing effective methods for proactive re-routing when congestion is predicted based on real-time traffic information.

An alternative to our work could be the research done on dynamic user-optimal traffic assignment (DTA) [8]–[10]. These solutions periodically compute the assignment of traffic flows to routes that lead to dynamic user equilibrium. Unfortunately, there is still a significant gap between the theoretical or simulation results and potentially deployable solutions. Some issues are: 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. For example, they assume the set of Origin-Destination (OD) pairs and the traffic rate between every OD pair are known. This information is highly dynamic especially in city scenarios, leading to frequent iterations of computationally expensive algorithms even when not needed from a driver benefit 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 congestions. Our system, on the other hand, is designed to be effective and fast, although not optimal, in deciding which vehicles should be re-routed when signs of congestion occur as well as computing alternative routes for these vehicles.

The complexity of DTA systems has led scientists to look for inspiration in Biology and Internet protocols. In [11], Wedde et al. developed a road traffic routing protocol, BeeJamA, based on honey bee behavior. Similarly, Tatomir et al. [12] proposed a route guidance system based on trail-laying ability of ants. Inspired by the well-known Internet routing protocols, Holger et al. [13] proposed decentralized Organic Traffic Control. However, since they are distributed, these approaches have only a partial view of the traffic conditions, which may lead to less accurate re-routing. Also, 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 re-routing service (which can physically be distributed across several servers), and (2) a vehicle software stack for periodic traffic data reporting (position, speed, direction) and showing alternative routes to drivers. Vehicles run this software either on an embedded vehicular system or a smart phone. 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 computed according to its strategy. It is assumed that the service knows the road network as well as 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 re-routing; and (4) choosing alternative routes for each such vehicle and pushing the guidance to the vehicles. Since data collection has been studied extensively in the literature, we assume that the centralized service receives traffic data from

vehicles and road-side sensors where available. We discuss in detail each of the other phases in this section and 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 updated periodically as new traffic data becomes available. Several methods can be employed to estimate the travel time over a road segment. For instance, using vehicle probe data collected from on-board GPS devices to reconstruct the state of traffic is a well-studied topic [2], [14]. We use the Greenshield's model [15] to estimate the travel time since it is used extensively in dynamic traffic assignment 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, as seen in Equation 1:

$$V_i = V_f(1 - \frac{K_i}{K_{jam}}) \qquad T_i = L_i/V_i \tag{1}$$

where K_{jam} and V_f are the traffic jam density and the free flow speed for road segment i, while T_i and L_i are the estimated travel time and length for the same segment. 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 speed limit. Basically, K_i/K_{jam} is the ratio between the $current\_number\_of\_vehicles$ and the $max\_number\_of\_vehicles$. The $current\_number\_of\_vehicles$ is obtained from the traffic data collected by the service, whereas the $max\_number\_of\_vehicles = length\_of\_road/(avg\_vehicle\_length + min\_gap)$.

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 $K_i/K_{jam} > \delta$, where $\delta \in [0, 1]$ is a threshold. Choosing the right value for δ is particularly important for the service performance. If it is too low, the service could trigger unnecessary re-routing; this may lead to an increase in the drivers' travel times. If it is too high, the re-routing 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 re-routed

When a certain road segment presents signs of congestion, the service looks for nearby vehicles to re-route. Specifically, we select vehicles from incoming segments (i.e., segments which bring traffic into the congested one). To decide how far from congestion to look for candidates for re-routing, the service uses a parameter L (level), which denotes the furthest distance (in number of segments) the vehicle can be away from the congested segment. Basically, the service performs a breadth first search (BFS) 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 re-routing.

IV. Re-routing Strategies

Recent research has proved that real-time traffic flow data and road travel time can be determined based on data reported by vehicles or road-side sensors [2], [16], [17]. The question is how to utilize this knowledge in an intelligent fashion to avoid congestion and reduce the drivers' travel times. This section presents our three re-routing strategies; all of them use the estimated travel time in the computation of the (k-)shortest path(s) for each of the vehicles selected as described in the previous section.

A. Dynamic Shortest Path

Dynamic Shortest Path (DSP) is a classical re-routing strategy that assigns the selected vehicles to the path with lowest travel time. The advantage of this strategy lays in its simplicity and consequently reasonable computational cost $O(E + Vlog(V))$, where E is the number of road segments and V is the number of intersections. We expect this strategy to provide good results when the number of re-routed vehicles is low. In this case, the risk of switching congestion from one spot to another is low.

B. Random k Shortest Paths

Random k Shortest Paths (RkSP) computes for each vehicle to be re-routed its k-shortest paths. Then, it 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 re-routed traffic among several paths. The price to pay is a higher computational complexity, $O(kV(E + Vlog(V)))$ [18], which increases linearly 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 of the travel time for some drivers, RkSP limits the maximum allowed relative difference between the fastest and the slowest path to 20%. In section V, we experimentally vary k to measure its impact.

C. Entropy Balanced k Shortest Paths

While RkSP addresses the main potential shortcoming of 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 re-routing intersection. Second, it assigns paths randomly to vehicles, which is far from optimal both from a driver point of view and the global traffic point of view. Therefore, we propose an Entropy Balanced k Shortest Paths (EBkSP) strategy to improve on RkSP at the cost of slightly increased complexity. 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. We expect this optimization to improve the traffic from a global point of view. In addition, EBkSP ranks the cars to be re-routed 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 re-routed first.

To avoid creating new congestions through re-routing, we associate a "popularity" measure to road segments in EBkSP. Future congestion occurs if many drivers take the same road segment within the same future time window. [1] As we assume that the drivers share their route information, it is possible to estimate the future footprint of each driver in the road network.

Definition 1. *A footprint counter, fc, of a road segment is the total number of vehicles that are assigned to paths that include this segment.*

Similar to entropy in information theory, we define the path popularity as follows.

Definition 2. *Let $(p_1, ..., p_k)$ be the set of paths computed for the vehicle which will be assigned next. Let $(r_1, ..., r_n)$ be the union of all segments of $(p_1, ..., p_k)$, and let $(fc_1, ..., fc_n)$ be the set of footprint counters associated with these segments. The popularity of p_j is defined as $Pop(p_j) = e^{E(p_j)}$. $E(p_j)$ is the weighted entropy of p_j and is computed as $E(p_j) = -\sum_{i=1}^{n} \omega_i \frac{fc_i}{N} \ln \frac{fc_i}{N}$, $N = \sum_{i=1}^{n} fc_i$, and $\omega_i = \frac{C_{avg}}{C_i}$ is a weight factor. C_{avg} is the average road capacity of the network and C_i is the road capacity of r_i..*

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

In addition to path popularity, EBkSP uses an urgency function to rank the vehicles to be re-routed.

Definition 3. *Given a set of vehicles $V = (v_1, v_2, v_3, ..., v_m)$ to be re-routed, we define two urgency functions to compute the re-routing priority of a vehicle in V:*

- *Relative Congestion Impact: RCI=($RemTT$-$RFFTT$)/$RFFTT$*
- *Absolute Congestion Impact: ACI=$RemTT$-$RFFTT$*

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

RCI measures the impact of congestion on a vehicle relative to its remaining travel time, whereas ACI emphasizes the absolute increase in the travel time. In section V-B, we evaluate EBkSP under these two urgency functions.

After vehicle selection and prioritization according to the urgency function, we assign each driver the path with least

[1]The time window size equals the period used by the system to evaluate congestion.

Algorithm 1 EBkSP re-routing pseudo-code

```
procedure main
updateEdgeWeights()
congestedRoads=detectCongestion(edgeWeights)
if #congestedRoads>0 then
    selectedVehicles=selectVehicles(congestedRoads)
    sortedVehicles=sortByUrgency(selectedVehicles)
    odPairs=updateODPairs(selectedVehicles)
    allPaths=compuate_all_kShortestPaths(odPairs)
    doReroute(allPaths, sortedVehicles)
end if
end procedure

procedure doReroute(allPaths, sortedVehicles)
for all vehicle in sortedVehicles do
    {origin, dest}=getVehicleOD(vehicle)
    kPaths=getkPaths(allPaths, origin, dest)
    newpath = getLeastPopularPath(kPaths)
    setRoute(vehicle, newpath)
    updateFootprint(vehicle, newpath)
end for
end procedure
```

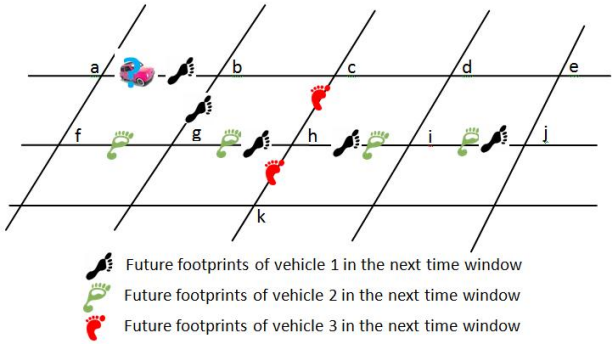


Fig. 1: A simple re-routing example

popularity among previously assigned vehicles. Specifically, the most urgent vehicle is assigned the current best path without considering others. Then, the future footprints is updated based on the new path. When assigning the second vehicle, the popularity score can be calculated and only the least popular path will be chosen. Algorithm 1 presents the EBkSP's pseudo code. Besides what was described so far, the pseudo-code shows an optimization related to shortest path computation. Instead of computing the k-shortest paths for each vehicle to be re-routed, we compute these paths for the origin-destination (OD) pairs that have selected vehicles traveling between them (i.e., the set of OD pairs is much smaller than the set of vehicles to be re-routed). The same procedure is used in RkSP and DSP.

Figure 1 shows an example. We assume vehicles (v_1, v_2, v_3) have been assigned to their paths before v_4, and each road has the same capacity (i.e., $\omega_i = 1$). The footprints of (v_1, v_2, v_3) in the next time window are $C_1(ab, bg, gh, hi, ij)$, $C_2(fg, gh, hi, ij)$, and $C_3(ch, hk)$. For v_4, which travels from ab to ij, there are three alternatives 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 $(ab, bg, gh, hi, ij, bc, ch, cd, di)$, and their footprint counters are (1,1,2,2,2,0,1,0,0). Consequently, N=9, $E_V(p_1)$=1.49, $E_V(p_2)$=1.16 and $E_V(p_3)$=0.58. Hence, v_4 will be assigned

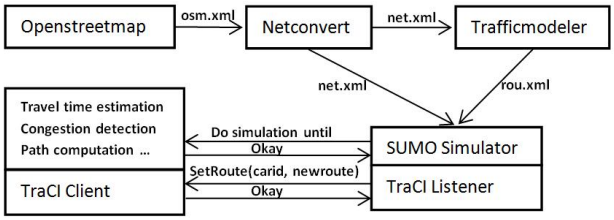


Fig. 2: The simulation process

to p_3 because it is the least popular.

V. Evaluation

The objective of our simulation-based evaluation is to study the performance of the three re-routing strategies under various scenarios. Specifically, we address the following questions:

- What is the average travel time, average number of re-routings and computation time among the proposed strategies?
- How does each parameter (e.g. number of alternative paths, re-routing period, congestion threshold, 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 that have our software)?

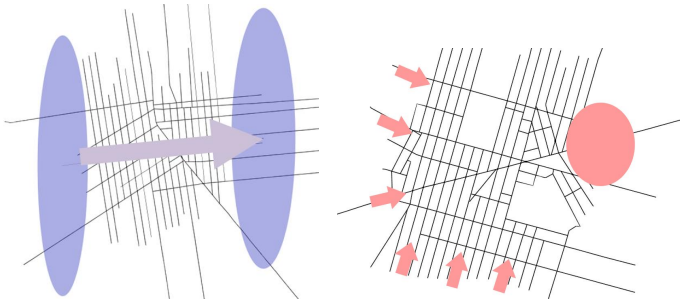
A. Simulation setup

We employed SUMO [19] and TraCI [20] 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 vehicle state, road configuration, and traffic lights. We implement the re-routing strategies 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 [21] in osm format. One is a section of Brooklyn, NY and the other is in Newark, NJ. We use the Netconvert tool in SUMO to convert the maps into SUMO usable format, and the Trafficmodeler tool [22] to generate vehicle trips. Netconvert removes the pedestrian, railroad, and bus routes, and sets up a static traffic light at each intersection. All roads have one lane in each direction, with the same speed limit. The statistics of the two networks are shown in Table I. By default, the shortest distance paths are automatically calculated and assigned to each vehicle at the beginning of simulation. Therefore, we modified the shortest path algorithm in SUMO to be based on the travel time instead of distance. Figure 2 illustrates the simulation process. Figures 3 (a) (b) show the traffic flow in both networks. We used Trafficmodeler to generate a total of 1000 cars in the Brooklyn network from the

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



(a) Brooklyn (b) Newark

Fig. 3: Traffic flow in the road networks

left area to the right area in an interval of 1000 seconds. The origins and the destinations are randomly picked from the left area and the right area, respectively. In the Newark network, 908 cars were generated having the origins picked randomly 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 13.0.1 for vehicle length=5m, the minimal gap=2.5m, the car following model (Krauss [23]), and the driver's imperfection=0.5. For each scenario, we average the results over 15 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.

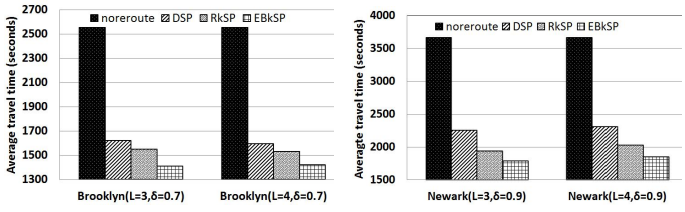
B. Results and Analysis

Average travel time. Figure 4 presents the average travel time obtained with the three strategies on both networks. The "noreroute" bars indicate the travel time in the absence of any re-routing strategy. The results show that all the proposed strategies improve the travel time significantly. EBkSP has the best results; compared to "noreroute", it reduces the travel time by up to 81% and 104% on Brooklyn and Newark, respectively. Compared to DSP, it is better by 15% and 25%.

The results confirm our hypotheses in Section IV. DSP can improve the travel time, since it re-routes dynamically the vehicles by considering the traffic conditions. However, in some cases, if many vehicles have similar current positions and

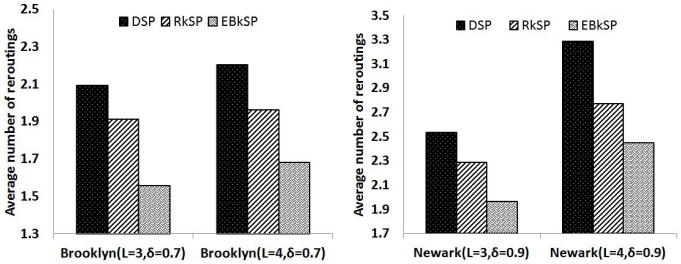
TABLE II: Parameters used in the evaluation

period	The frequency of triggering the re-routing
threshold δ	Congestion threshold; if $K_i/K_{jam} > \delta$, the road segment is considered congested
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 $k = 3$
compliance rate	Percentage of drivers who accept the guidance
penetration rate	Percentage of vehicles which have our software



(a) Brooklyn (b) Newark

Fig. 4: Average travel time (k=3, urgency=ACI, period=450s)



(a) Brooklyn (b) Newark

Fig. 5: Average number of re-routings (k=3, period=450s, urgency=ACI)

destinations, new congestion can be create by the re-routing process. RkSP avoids this shortcoming since it balances the flows among several paths. Nevertheless, a randomly picked path is not necessarily the best one. Finally, EBkSP offers the best performance by carefully selecting the path for each re-routed vehicle. The rationale is that the possibility of creating new congestion is lower if the driver takes the least popular path among k paths with similar travel time.

Our experiments also demonstrated that levels 3 and 4 work best for selecting a relatively optimal number of vehicles for re-routing (they have similar performance for Brooklyn, and level 3 is better for Newark). Lower levels do not select enough cars. As shown in Figure 5, level 4 leads to more re-routings. Thus, we believe level 3 is a better choice for the system.

Average number of re-routings. It is important that that the number of re-routings for a given vehicle during a trip stay low. From the driver point of view, changing the path to her destination too often can be distracting and annoying. This can eventually lead to a rejection of the guidance system by the drivers. From the system point of view, having a low number of re-routings means decreasing the computational burden on the system because the re-routing process is costly. Figure 5 compares the number of re-routings across the three proposed strategies. The results indicate that EBkSP produces the least number of re-routings. Specifically, compared to DSP, EBkSP reduces the average number of re-routings by up to 34% on both Brooklyn and Newark. Compared to RkSP, EBkSP is better by a margin of 22% and 17%. The reason is that by considering future path information in the re-routing decision, EBkSP can not only mitigate the current congestion, but also avoid creating new congestions; hence, the lower necessity for recurrent re-routing.

To confirm this analysis, we also measured the number of congested segments in each iteration for Brooklyn. Figure 6 shows the results. As traffic is generated during the first 1000s,

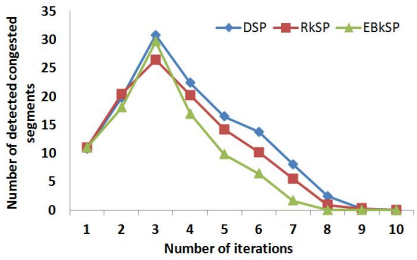


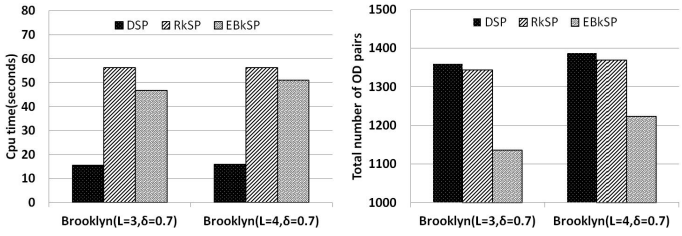
Fig. 6: Number of congested road segments on the Brooklyn network over time/iterations. (δ=0.7, period=450s, L=3, urgency=ACI, k=3)

the number of congested roads increases for all strategies in the first iterations. Then, all strategies manage to rapidly decrease the number of congested roads. As expected, EBkSP reduces the number of congested roads faster than the others.

CPU time. So far, the results have confirmed our hypothesis that EBkSP is the best strategy for fighting congestion, followed by RkSP and DSP. At the same time, the time complexity of the three algorithms inverts the ranking: DSP has the best computational efficiency, followed by RkSP and EBkSP. However, this complexity analysis is pertinent only when we consider the re-routing of a single vehicle. From the system point of view, the global computational complexity also depends on the number of re-routings 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 re-routing).

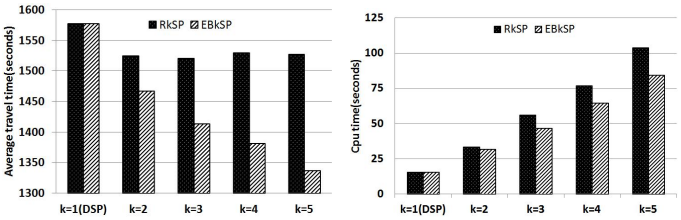
Figure 7 (a) shows the global CPU time consumed for re-routing by the three methods. DSP has the least CPU time, whereas RkSP needs higher computation time to achieve lower average travel time. Meanwhile, EBkSP requires less computation time than RkSP even though they have the same k-shortest path computation complexity. The reason is that EBkSP decreases the total number of re-routings processed in a period; this decrease becomes apparent when we look at the number of OD pairs in Figure 7 (b). DSP induces the most re-routings. However, since its complexity is the lowest, its computation time is the lowest as well. Nevertheless, since EBkSP performs significantly better in terms of average travel time and re-routing frequency, it will typically be the preferred strategy.

Number of alternative paths. k is a determinant parameter for the performance of both RkSP and EBkSP. 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



(a) CPU time (b) Number of OD pairs

Fig. 7: CPU time for the Brooklyn network(period=450s, urgency=ACI, k=3)



(a) Average travel time (b) CPU time

Fig. 8: Average travel time and CPU time for RkSP and EBkSP as function of k for the Brooklyn network (δ=0.7, period=450s, L=3, urgency=ACI)

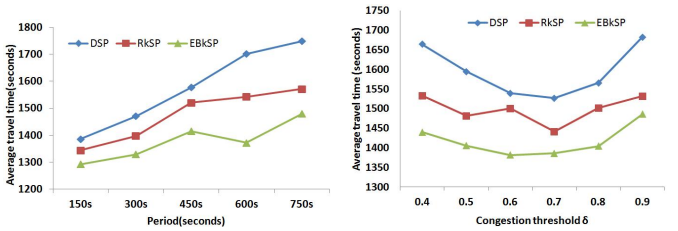
useless paths. Figure 8 compares the performance of RkSP and EBkSP with different k values on the Brooklyn network. When $k = 1$ the two methods degenerate into DSP, which produces the largest travel time at the lowest CPU cost. RkSP does not exhibit any performance improvement for $k > 2$, while EBkSP consistently produces lower travel times with higher k values. From Figure 8 (b), we can see that the computation cost increases linearly with k for both methods. However, EBkSP is faster than RkSP especially for larger k (e.g., EBkSP requires 23% less cpu time than RkSP when k equals 5). In conclusion, EBkSP has a much more robust and efficient performance than RkSP. Choosing the k value is a matter of trade-off between improving the average travel time and the computational cost.

Urgency function. EBkSP uses an urgency function to sort the list of vehicles selected for re-routing. To measure the performance difference between the two proposed ranking policies – RCI and ACI (cf. Definition 3), we conducted the ANOVA statistics test over the average travel time from 30 EBkSP runs. The results show that ACI produces lower average travel time than RCI ($p<0.01$) in a 95% confidence interval. Hence, we use ACI as our default urgency function in all the other experiments.

Re-routing period. Within the traffic guidance system, the re-routing process is triggered periodically at a pre-defined time interval. A shorter re-routing period leads to higher reactivity of the system, and thus to better travel times. However, the price to pay is increased computation cost, communication overhead, and potentially re-routings. At extreme cases, it might not even be possible to compute the alternative routes fast enough to push them to vehicles before they reach the re-routing intersections.

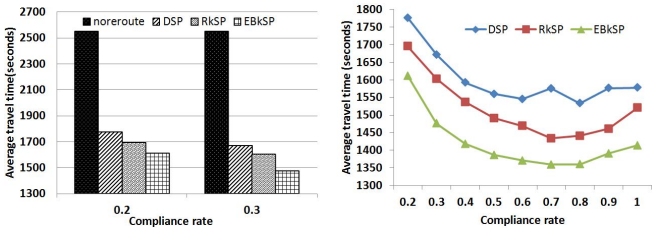
Figure 9 (a) shows the average travel time for different re-routing periods. Generally, the lower the period, the lower the average travel time is. Also, the strategies exhibit the closest results for the lowest period. As the period increases, the system's reactivity decreases, which translates into increased travel time. However, EBkSP consistently produces the lowest travel time even when the period is large. Furthermore, the performance loss is lower than with the other two strategies. Given the trade-offs between the average travel time on the one side and the computation cost, communication overhead, and number of re-routings on the other side, we chose a period of 450s for the other experiments.

Congestion threshold. Another key element determining the system reactivity is the congestion threshold (i.e., the



(a) Re-routing Period (δ=0.7) (b) Congestion Threshold (period=450s)

Fig. 9: Average travel time as function of the re-routing period (a) and congestion threshold (b) for the Brooklyn network (k=3, L=3, urgency=ACI)



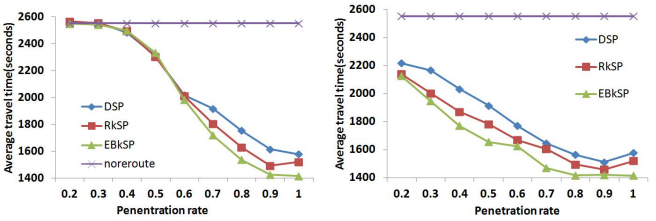
(a) Low compliance rate (b) All compliance rate

Fig. 10: Evaluation on compliance rate on Brooklyn network. (δ=0.7, period=450s, L=3, urgency=ACI)

density value on a road segment above which the system considers it as congested). Choosing a low density threshold may trigger unnecessary re-routing, which in turn leads to increased computational burden and re-routings. On the other hand, if the threshold is too large, congestion may be detected too late and the re-routing could be less effective. Figure 9 (b) confirms these hypotheses. When the threshold increases, the average travel time decreases due to better accuracy of congestion identification. However, above a certain threshold value, the congestion relief mechanism is triggered too late, leading to an increase of the average travel time. It is important to notice that the threshold depends on the network layout and type of the roads. For example, we observed that longer road segments work better with larger thresholds, while shorter road segments work better with slightly lower thresholds; in all the other experiments, we used δ=0.7 for Brooklyn and δ=0.9 for Newark.

Compliance rate. It is unrealistic to assume that every driver follows the re-routing guidance. The drivers' compliance rate (i.e., the proportion of drivers who accept the guidance) is an important factor for the re-routing strategy design. Therefore, we measured the average travel time while varying the compliance rate. Figure 10 (a) indicates that the average travel time can be significantly improved by all three strategies even under low compliance rates. Logically, Figure 10 (b) shows the performance improves for higher compliance rates. Somewhat surprisingly, we observe the performance degrades slightly for very high compliance rates. We believe this is due to our conservative approach to re-route more vehicles than necessary; thus, the strategies can be optimized even further.

Penetration rate. To understand how easy is to deploy our solution in real life, we study the effect of penetration rate (i.e., proportion of vehicles which have our software)



(a) Sensors: Vehicles Only (b) Sensors: Vehicles & Road-Side

Fig. 11: Average travel time as function of penetration rate (k=3, L=3, δ=0.7, period=450s, urgency=ACI, Brooklyn network)

on the average travel time. Figure 11 (a) shows the results when traffic data is collected only from vehicles (no support from road-side sensors). When the penetration rate is low, the performance is the same as "noreroute". This is because the service does not have enough data to accurately detect signs of congestion. Once the penetration rate is greater than 0.3, we observe increasing benefits. From 0.6 onward, EBkSP starts to perform better because a larger number of re-routed vehicles require good load balancing.

To simplify system adoption, we believe that data from road-side sensors (in conjunction with data from vehicles) can be leveraged to detect congestion more accurately. Figure 11 (b) demonstrates, indeed, that the performance can be significantly improved in such a case even for low penetration rates; thus, the initial adopters will have an incentive to use the system. The results also show EBkSP remains the best strategy.

VI. Conclusion and Future Work

This article presented three strategies for vehicular traffic re-routing that show very promising results compared to the "no re-routing" case. The EBkSP strategy balances best the trade-offs between low average travel time and low overhead along several parameters. The results also show that significant benefits can be achieved even with low compliance rate and moderate penetration rate. The experiments demonstrated how the performance can be tuned by varying parameters such as re-routing period, number of alternative paths, and density threshold. As future work, we plan to explore two directions. First, we will design an adaptive approach for vehicle selection that considers additional parameters such as road segment length, measured compliance rate, and estimated penetration rate. Second, we will investigate a hybrid architecture that off-loads parts of the computation and decision process in the network and uses V2V communication to better balance the need for privacy, scalability, and low overhead with the main goal of low average travel time.

Acknowledgment

This work was supported in part by the National Science Foundation under Grant Number CNS-0831753.

References

[1] D. Schrank, T. Lomax, and S. Turner, "Tti urban mobility report," 2011. [Online]. Available: http://tti.tamu.edu/documents/mobility-report-2011.pdf

[2] D. Work, O. Tossavainen, S. Blandin, A. Bayen, T. Iwuchukwu, and K. Tracton, "An ensemble kalman filtering approach to highway traffic estimation using gps enabled mobile devices," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE, 2008, pp. 5062–5068.

[3] S. Lämmer and D. Helbing, "Self-stabilizing decentralized signal control of realistic, saturated network traffic," 2010. [Online]. Available: http://www.santafe.edu/media/workingpapers/10-09-019.pdf

[4] http://www.inrix.com.

[5] J. Wardrop, "Some theoretical aspects of road traffic research," *Proceedings of the Institution of Civil Engineers, Part II*, vol. 1, no. 36, pp. 252–378, 1952.

[6] B. Kerner, "Optimum principle for a vehicular traffic network: minimum probability of congestion," *Journal of Physics A: Mathematical and Theoretical*, vol. 44, p. 092001, 2011.

[7] B. Coifman and R. Mallika, "Distributed surveillance on freeways emphasizing incident detection and verification," *Transportation Research Part A: Policy and Practice*, vol. 41, no. 8, pp. 750–767, 2007.

[8] Y. Chiu, J. Bottom, M. Mahut, A. Paz, R. Balakrishna, T. Waller, and J. Hicks, "Dynamic traffic assignment: A primer," *Transportation Research E-Circular*, no. E-C153, 2011.

[9] N. Taylor, Transport, and R. R. Laboratory, *CONTRAM 5, an enhanced traffic assignment model*, ser. TRRL research report. Transport and Road Research Laboratory, 1990.

[10] H. S. Mahmassani, T.-Y. Hu, and R. Jayakrishnan, "Dynamic traffic assignment and simulation for advanced network informatics (dynasmart)," in *2nd International CAPRI Seminar on Urban Traffic Networks*, Capri, Italy, 1992.

[11] S. Senge and H. Wedde, "Bee inspired online vehicle routing in large traffic systems," in *ADAPTIVE 2010, The Second International Conference on Adaptive and Self-Adaptive Systems and Applications*, 2010, pp. 78–83.

[12] B. Tatomir, S. Fitrianie, M. Paltanea, and L. Rothkrantz, "Dynamic routing in traffic networks and manets using ant based algorithms," in *Proceedings of the 7th International Conference on Artificial Evolution, Lille, France*, October 2005.

[13] H. Prothmann, H. Schmeck, S. Tomforde, J. Lyda, J. Hahner, C. Muller-Schloer, and J. Branke, "Decentralised route guidance in organic traffic control," in *Self-Adaptive and Self-Organizing Systems (SASO), 2011 Fifth IEEE International Conference on*. IEEE, 2011, pp. 219–220.

[14] S. Maerivoet, "Modelling traffic on motorways: State-of-the-art, numerical data analysis, and dynamic traffic assignment," Ph.D. dissertation, Katholieke Universiteit Leuven, 2006.

[15] J. Banks, *Introduction to transportation engineering*. McGraw-Hill, 2002.

[16] E. Horvitz, J. Apacible, R. Sarin, and L. Liao, "Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service," in *Twenty-First Conference on Uncertainty in Artificial Intelligence*, 2005, pp. 244–257.

[17] P. Mohan, V. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008, pp. 323–336.

[18] E. Lawler, "A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem," *Management Science*, pp. 401–405, 1972.

[19] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo - simulation of urban mobility: An overview," in *SIMUL 2011, The Third International Conference on Advances in System Simulation*, Barcelona, Spain, 2011.

[20] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J. Hubaux, "Traci: an interface for coupling road traffic and network simulators," in *Proceedings of the 11th communications and networking simulation symposium*. ACM, 2008, pp. 155–163.

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

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

[23] S. Krauss, P. Wagner, and C. Gawron, "Metastable states in a microscopic model of traffic flow," *Physical Review E*, vol. 55, no. 5, p. 5597, 1997.