Mitigating Congestion Using Environment Protective Dynamic Traffic Orchestration

Daniel Petrov, Rakan Alseghayer, Panos K. Chrysanthis
Department of Computer Science, University of Pittsburgh
{dpetrov,ralseghayer,panos}@cs.pitt.edu

Abstract—**Traffic congestion has a significant negative impact on the accelerating pace of daily human activities. Traffic jams increase the transportation costs for goods and humans. They are also amongst the leading factors for pollution in the atmosphere and consequently increase health risks for the population. One way to reduce the amount of emissions produced by vehicles in traffic jams is to mitigate traffic congestion and promote the usage of public transportation. In this paper, we propose a solution that establishes on-demand, virtual bus lanes to prioritize public transportation over other traffic and provide detour guidelines for other drivers, while causing insignificant detour penalties. Our solution leverages incremental window aggregations to identify the busiest road segments, priority scheduling, and Dijkstra shortest path algorithm to shape and detour traffic. Our experimental evaluation shows the effectiveness of our Environment Protective Traffic Orchestration (EPTrOn) algorithm in identifying and alleviating traffic congestions.**

Index Terms—**traffic, traffic congestion, Dijkstra, virtual bus lanes, internal combustion engine**

I. Introduction

Most internal combustion engine vehicles (ICEVs) have their engines idling when they are not in motion, i.e., when they stop at traffic lights and crosswalks, or when they are in traffic jams [3]. Furthermore, most public transit buses are equipped with diesel engines. These engines not only produce greenhouse gas emissions, but their exhaust contains a significant amount of fine particulate matter (FPM), the inhalation of which has a negative impact on human health. A plethora of diseases are attributed to FPM—asthma and lung cancer, to name a few.

Rapid proliferation of smart mobile devices that are equipped with positioning sensors (e.g., GPS and Galileo), and ubiquitous Internet connectivity, facilitated the growth of the near real-time traffic analysis necessary for effective solutions to traffic jams. Some cities already implement smart traffic lights that adaptively steer the traffic in an effort to mitigate congestion [1], [2]. However, studies show that the additional infrastructure built does not solve the problem with traffic jams. It only attracts new traffic and changes the scale of the problem—the phenomenon of "induced traffic" [4]–[6]. This suggests that *we need a balanced solution that promotes the use of public transportation while reducing idling of both cars and buses*.

In this paper, we propose such a balanced solution, called *Environment Protective Traffic Orchestration* (*EPTrOn*). *EPTrOn* mitigates congestion by establishing bus lanes on demand, which we coin *virtual* bus lanes, and shaping traffic by controlling traffic lights and directing traffic using lightboards at intersections. Our solution proactively ameliorates the traffic ahead of buses in congested areas, and adaptively detours cars away from the congested areas, while protecting the interests of both public transportation and car riders.

Our **Contributions** in brief are:

- A method that identifies congested road segments by analyzing trajectory data in real-time. It uses incremental sliding window aggregations to calculate the average speed of each vehicle, and an R^+ tree to record their positions in the road network, formalized using a semantics enriched graph G(V, E, M). (Sec. II)
- A solution that creates dynamic virtual bus lanes and provides guidelines for drivers about the least busy path towards their destinations. Our solution does not require drivers to disclose their destination, but is based on a bi-objective shortest path algorithm that computes all the detours to neighboring landmark points at each intersection that the drivers can follow. It further uses priority scheduling of traffic lights to increase the length of green lights for buses and cars in congested roads (Sec. III)
- An experimental evaluation of our *EPTrOn* solution using a real dataset of the city of Beijing shows that *EPTrOn* outperforms the baseline in terms of bus' completion time (by up to 590%), while causing small increases in cars' average detour distance (by up to 768 meters). The former metric captures the bus on time performance and bus users' satisfaction, whereas the latter captures the car detour penalty and car drivers' and passengers' potential dissatisfaction. (Sec. V)

II. Preliminaries

In this section, we introduce the notation used throughout the paper, and our system model, and formulate our problem.

A. Notation

In order to formally define the problem of creating dynamic bus lanes on demand, we adopted the following notation.

Definition 1: The road network of a city, including bus stops and facilities, is represented by a semantically enriched graph $G = (V, E, M)$, whereby the intersections are the vertices in V, the streets are the edges in E, and the semantic information for each vertex and each edge are vectors in M.

A vector $m_i = (w, \gamma, sem_1, sem_2, sem_3, ...), m_i \in M \wedge i \in V \uplus E$, is of varying length with type specific parameters:

w is the weight of the edge/vertex, which represents how busy the road/intersection is, and γ is the velocity threshold for an edge, which defines when the edge is congested.

An example of edge semantics of *the edge of 5th Avenue in New York City, right in front of the Public Library* is $m_{e_5} = \{0.342,\ 8,\ 40.753486, -73.980888, 40.752184, -73.981843, 1,\ 5,\ 25,\ 1,\ 1,\ o\}$; the first two numbers are the weight w and velocity threshold γ, followed by the latitude and longitude of the northern end of the segment, and the same coordinates of the southern end. The next value 1 means that the edge is one way, 5 is the number of lanes, and 25 is the speed limit in mph. The next two parameters denote the fact that there are sidewalks on each side of the edge. The last parameter o denotes the number of buses on that edge. An example of vector semantics at the intersection of the NYC Public Library is $m_{v_{1324}} = \{0.2412,\ o,\ 40.753486,\ -73.980888,\ 4\}$; the first element of the vector is the weight, followed by the cumulative number of buses o that approach the intersection on the edges connected to the vertex, and the coordinates of the vertex, as well as the number of edges it connects.

For our definition, we use an undirected graph. It is clear that a directed graph may be a more accurate model of the road networks of different cities. However, the extension from undirected to directed graph is trivial and the directed graph model does not impact our approach to solving the problem.

The terms "edge" and "road segment" for $e \in G$ will be used interchangeably. Given the above definitions for the city road network, we can formally define traffic congestion as:

Definition 2: A road segment is congested *iff* the average speed of the vehicles, passing through it over a given epoch of time e, is below a specified threshold of γ miles per hour.

Furthermore, we make a clear distinction between paths and trajectories of vehicles in our model of the city road network.

Definition 3: A path p, from a starting point $s \in G$ to an end point $t \in G$, is a sequence of edges (road segments) connecting the points (vertices) s and t in G. P is a subgraph of G that consists of all paths p from s to t.

Definition 4: A trajectory of a vehicle is defined as a path p in G, whereby each of its road segments is semantically enriched with one or more timestamps that show the moment(s) in time when the vehicle was on that particular road segment. Each trajectory has a *directionality* property (i.e., from s to t) and *diversion* property that is defined as follows.

Definition 5: The diversion of a trajectory D is a set of points $d_i \in G$ that do not extend the trajectory by more than a given threshold of τ miles when added to it.

The vehicle diversion controls a car's rerouting by preventing it from diverting too far away from its initial trajectory. The directionality of a trajectory is not changed if and only if the rerouting morphs the trajectory within the set D.

The bus routes are also paths in the road network G. The buses and their trajectories (i.e., locations of these buses) are known in real time, as many cities worldwide now provide this information in real time (e.g., Busgazer [7]).

Definition 6: A neighborhood is a subset $G' = (V', E', M')$ of G, whereby each two intersections v'_1 and v'_2 in V' are connected with an edge e' in E' and all edges from E' end in vertices in V'. Neighboring neighborhoods have edges in common, connecting them, but not vertices in common.

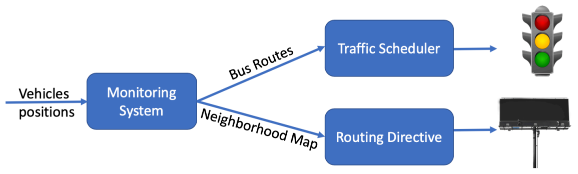

Fig. 1. EPTrOn Solutions

The idea of the neighborhoods reflects the concept of neighborhoods in cities and is inspired by the concept of *autonomous systems* in computer networks routing.

B. System Model

We assume that internal combustion engine cars and buses (ICEVs) are equipped with a mobile computing device that has a global navigation satellite system and Internet connectivity capabilities. These devices report the current location and the speed of the vehicle periodically, but not its destination.

A (monitoring) system receives these measurements from the n vehicles over an epoch of time e. Each data point is a tuple $tup(vehcID, ts, long, lat, a)$ consisting of a unique vehicle identifier $vehcID$, timestamp ts, longitude lon, latitude lat, and current speed a. The timestamp captures the moment in time when the tuple was produced and is denoted in global time. The type of vehicle can be derived from the unique vehicle ID, i.e., car or bus. The consecutive tuples for a given vehicle form its trajectory in the time epoch e.

In addition to the device that controls the traffic lights, we assume that each intersection is equipped with a lightboard that is used to display information to drivers. Some cities and highways already use such boards to provide traffic and weather updates, and details about detours and points of interest (POI).

C. Problem Formulation

Given a road network $G(V, E, M)$, its current state as captured by the semantics M, and the trajectories of the mass transit vehicles, calculate paths p' for the cars such that:

- they ameliorate the traffic in the way of public transportation vehicles at the next epoch of time e, and
- they do not change the directionality nor violate diversion τ of the trajectories of the ICEVs.

The objective of our solution is to find a subset of paths that reroute cars with minimal impact on their traveling time/distance. The optimization criteria are the alleviation of traffic in the way of buses and the mitigation of pollution.

III. EPTrOn Solution

Our solution is depicted in Fig. 1 and consists of three integral components: a *Monitoring System*, which identifies congested road segments, a *Traffic Scheduler*, which produces the scheduling for traffic lights, and a *Routing Directive*

Algorithm 1 Monitoring System

```
Input: $G$, $R$, $C$
Output: $Q$
    Update the weights of all edges, based on the position of each car or bus
1: for each $tup \in C$ do
2:     $R.CarNext++(vehcID_{tup}, lon_{tup}, lat_{tup}) = e_{curr}$
3:     $R.w_{e_{prev}} -= 1$
4:     $R.w_{e_{curr}} += 1$
5: end for
    Traverse the tree and place all vertices in Q
6: $R.traverse(Q)$
7: return $R$, $Q$
```

component, which calculates the optimal paths in each neighborhood and produces directives/guidelines to drivers that are shown on the light-boards. Each component is described next.

A. Monitoring System

The location data (tup) from cars and mass transit vehicles (buses) is produced at high velocity. The real-time analytical processing to identify congested road segments is performed in micro-batches. A micro-batch C is a group of tuple tup subsequences, $tup \in C$, over a set of data streams defined by a timestamp interval I of length l. The inter-arrival time of two consecutive micro-batches specifies the maximum computational time for processing a micro-batch. The inter-arrival time is the delay target, or deadline d, by which the last result can be produced while analyzing a micro-batch.

The monitoring system receives and ingests all tuples tup from both cars and mass transit vehicles within a micro-batch C_{curr}. It uses a two-dimensional hashing R, specifically R^+ tree, and associates the position of the cars and buses with the respective edges of G. It also uses incremental sliding window aggregation techniques [8] to calculate the average speed of each vehicle and the average speed of the vehicles on each edge. Subsequently, it updates the weights w of all edges and vertices in G and updates the semantic information o about the number of buses that are located on each edge and the cumulative number of buses that approach each vertex. It also traverses the R^+ tree and builds a priority queue Q that contains the vertices, sorted in decreasing order of number of buses and congested edges.

By the end of an interval, all $tup \in C_{curr}$ are processed and the updated graph G and the priority queue Q are passed to the next component, namely the Traffic Scheduler, which schedules the traffic lights at each intersection. While the traffic lights are scheduled, the next micro-batch C_{next} is generated and sent to the Monitoring System. The cumulative length of d to process a micro-batch and the time needed to schedule the traffic lights define the duration of the epoch e.

The monitoring system component is deployed on a per-neighborhood basis. The computations are independent, and they are trivially parallelizable. The pseudo code of the algorithm is shown on **Algorithm 1**.

B. Traffic Scheduler

Similarly to the Monitoring System, this component is deployed on a per-neighborhood basis and operates at two-levels. At the top level, the global scheduler controls the order

Algorithm 2 Traffic Scheduler

```
Input: $G$, $R$, $Q$
Output: $Traffic\ lights\ scheduling$
    Initialization:
1: $Q = \emptyset$, $Q' = \emptyset$, $''Q = \emptyset$, $'''Q = \emptyset$, $Q^{IV} = \emptyset$, $Q^V = \emptyset$
    Place vertices to queues
2: for each $v \in Q$ do
3:     if $Congested(w_v)\ AND\ q_m \in Q$ then
4:         $Q' = Q' \cup v$
5:     else
6:         if $Congested(w_v)\ AND\ not(q_m \in Q)\ ANDonBUsRoute(v)$ then
7:             $Q'' = Q'' \cup v$
8:         else
9:             if $Congested(w_v)\ AND\ not(q_m \in Q)\ ANDnot(onBUsRoute(v))$ then
10:                $Q''' = Q''' \cup v$
11:            else
12:                if $not(Congested(w_v))\ AND\ q_m \in Q$ then
13:                    $Q^{IV} = Q^{IV} \cup v$
14:                else
15:                    $Q^V = Q^V \cup v$
16:                end if
17:            end if
18:        end if
19:    end if
20: end for
21: for each $Q_{temop} \in \{Q', Q'', Q''', Q^{IV}, Q^V\}$ do
22:     for each $v \in Q_{temop}$ do
23:         $runTrafficScheduling(v)$
24:     end for
25: end for
26: return
```

of processing at intersections within the current interval using five priority queues. It initiates these queues by traversing the priority queue Q received from the monitoring system, and it distributes the vertices amongst five local priority queues, Q' to Q^V. Queue Q' maintains the vertices that contain at least one congested road segment with buses on them, Q'' maintains the vertices that are ends of edges that are on bus routes and are congested but do not currently have buses, Q''' is the priority queue that maintains the intersections that are ends of at least one edge that is congested but is not on bus routes, Q^{IV} has the vertices that connect segments that are on bus routes but are not congested, and Q^V has the information about other vertices (i.e., connecting edges that are not on bus routes, not congested). The global scheduler only sends a signal to the devices on the respective intersections to prepare and run their own green and red light scheduling.

At each intersection, a local scheduler controls the green and red light signal interval lengths. Our solution runs a priority scheduling, whereby the edge with the highest priority is the one with the highest number of buses on it. The interval length of the green signal is for as long as cars and buses from that edge can go through the intersection. Once the edge is empty, or no vehicles can move because the edge is saturated by a traffic jam, the next edge by number of buses at the same intersection gets a green light signal. The operation gets repeated until all edges get a green signal, or until the interval is over. If no edges at the intersection have buses, the more congested ones get a green light first. When two edges have the same number of buses, their congestion is used as a tie-breaker to schedule one before the other. The pseudo code for the algorithm is presented on **Algorithm 2**.

Algorithm 3 Routing Directive

```
Input: $G$, $R$
Output: $Detour\ Guidance$
  Obtain information for detours
1: for each $n \in N$ do
2:     for each $v \in n$ do
3:         $type = icev$
4:         $S_q^i = Dijkstra(G, M, type, v)$
5:         $type = ev$
6:         $S_q^e = Dijkstra(G, M, type, v)$
7:     end for
8: end for
9: return
```

C. Routing Directive

This component operates in neighborhoods and uses the well-studied Dijkstra shortest path algorithm [9] to calculate the optimal detour routes of the cars. *EPTrOn* has a two-fold gain by operating at the neighborhood level: (1) the optimal routing can be calculated for each neighborhood independently; and (2) the calculations for the optimal routing are dependent on the number of vertices, and the notion of neighborhood bounds this number to a small enough value that makes calculations possible in real time. Hence, *EPTrOn* can efficiently provide directives/guidance for the local drivers by informing them of the optimal way to the next neighborhood on the way to their ultimate destination. Furthermore, *EPTrOn* achieves privacy by relying on local drivers who know what neighborhoods of the city they have to go through in order to reach their ultimate destination from their source/current location without revealing their ultimate destination. The pseudo code of the algorithm is shown in **Algorithm 3**.

IV. Experimental Testbed

In this section, we present the experimental testbed that we developed to evaluate our *EPTrOn* solution.

Algorithms: In our evaluation, we compare $EPTrOn$ with two other algorithms, *Car biased* and *Naive*. *Car biased* is based on $EPTrOn$ and it differs from it in prioritizing vehicles routing. While our solution prioritizes buses over cars, *Car biased* does the opposite. *Naive* is the standard approach with the basic traffic lights and no dedicated bus lanes.

Dataset: The dataset we use is collected by Microsoft Asia and covers the routes of more than 12000 taxis in Beijing. The dataset contains more than 15 million data points and covers more than 9 million kilometers (5.6 million miles) [11], [12]. Each tuple in the dataset contains the unique ID of the car that generated the tuple, the timestamp when the tuple was generated, as well as the geographical coordinates.

Moreover, we downloaded the map of the city of Beijing from *https://www.openstreetmap.com*, and we converted it into a graph of vertexes, using the OSMnx tool (*https://github.com/gboeing/osmnx*). Each intersection is represented as a vertex, and the street that connects two intersections is represented as an edge. We used a subset of the city that contains 2100 vertexes and 2600 edges.

From the meta-data, available on OpenStreetMap, we extracted the bus routes of the public transportation. Our graph contained seventy four different routes.

Setup: We developed our testbed in C++ 11. It is based on a fixed length time interval for scheduling all edges that connect to the same intersection. A device at each intersection schedules the green light length for each edge. We consider all streets to be of the same width—one lane and that all vehicles move unidirectionally. Assuming that the majority of the intersections are on the crossing of two streets, the traffic lights run the red and green cycle for at most four different directions (i.e., when serial scheduling of each direction on each street is scheduled). At each epoch, all directions of the traffic on an intersection get scheduled. Typically, the total time is 30 sec. We skip the time of yellow lights for simplicity.

We calculate that on average 60 cars (15 cars per edge in a 4-edge vertex) can go through each intersection within an epoch of time by approximating the average car speed at 20 mph at the intersection, and the typical car size at 5 yards long.

Typically, cars average speed is 25 mph and buses' 15 mph. Some routes are served by more than one bus. Buses that serve the same bus route are spaced ten minutes apart, or 1.5 $miles$. The location of taxis is extracted from the dataset, and they are placed at their respective locations. The rest of the cars are distributed randomly based on the *road network load factor*.

As stated, cars are not required to disclose their destination. In order to simulate traffic flow or drive behavior at each intersection, the cars exhibit traffic distributions for the different directions: West (left), NW, North (Straight), NE, East (right), and no U-turns. For example, a uniform traffic distribution where an equal percentage of cars go in all directions will be {02, 02, 02, 02, 02}.

To mimic the situation whereby some bus stops are on the side of the street to avoid blocking the traffic behind the buses when they stop, we maintain a tunable parameter *bus delay* (*bd*), that specifies how many epochs will pass before buses can get back on the road after they stopped at a bus stop.

To specify bus priority in crossing an intersection during a green light, we maintain a parameter *the reserved space for buses* (*bp*) that marks how many bus spots will be reserved on edges on the bus' route.

Metrics: For our experimental evaluation, we collect two different metrics in order to assess the bus on time performance and the car detour penalty that quantify bus riders' satisfaction and car passengers dissatisfaction, respectively.

- *Time Performance*: the *number of epochs* it takes for all buses to conclude their trips.
- *Detour Penalty:* the average detour distance taken by the cars (in $meters$) to reach their ultimate destination.

V. Experimental Results

We conducted four different experiments to study the sensitivity of our algorithm to different tunable parameters, namely the road network load factor, the amount of traffic violators, the reserved space for buses, and the amount of time buses are delayed by car drivers. Due to space limitations, we report the results of the first two. We ran our experiments with 79 buses in total. We ran the experiments on a 2 Intel CPUs server,

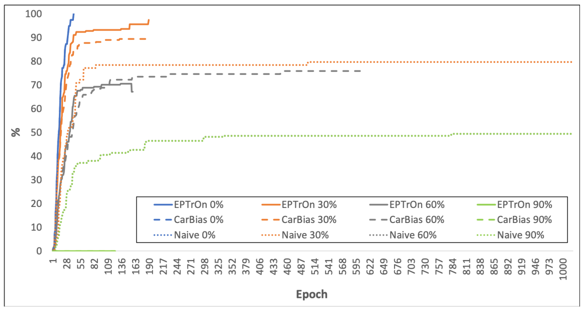

Fig. 2. The percentage of bus trips completed in epochs for [0%, 90%] traffic, $EPTrOn$, car biased, and naive approaches.

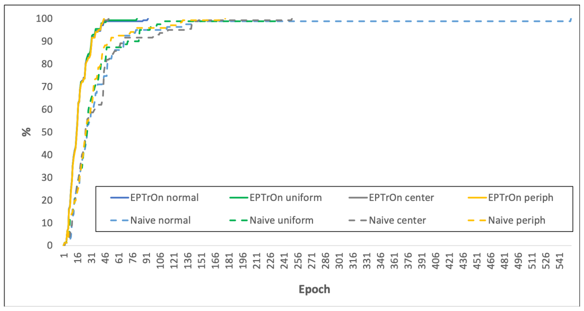

Fig. 3. The percentage of bus trips completed for Gaussian, uniform, center and peripheral destination distribution, for $EPTrOn$ and naive approaches.

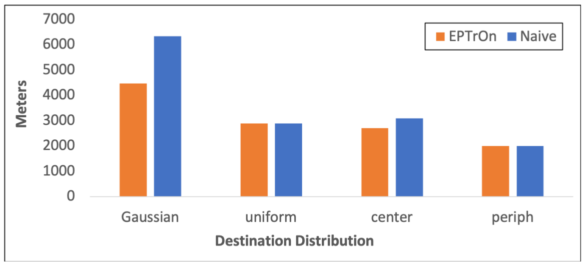

Fig. 4. The average detour distance for ICEV cars, Gaussian, uniform, center and peripheral destination distribution, $EPTrOn$ and naive approaches.

2.66GHz and 96GB, running CentOS 6.5., and using GCC version 6.3.0 compiler.

Experiment 1 (Fig. 2): In our first experiment we study the sensitivity of our algorithm to the amount of traffic (load factor) on the streets. This experiment shows how the different algorithms behave in rush hours as well as off-peak hours.

The amount of traffic is calculated as a percentage of the spots for cars on all streets in the network. When we say that the road network has 30% load, that means that the number of cars, distributed on the streets, is 30% of the total amount of car spots. The cars are distributed randomly and do not exceed the capacity of each road segment.

We experimented with 4 different values: 0%, 30%, 60%, and 90%. The traffic distribution based on directionality is Gaussian: $\{0.1, 0.2, 0.4, 0.2, 0.1\}$, the reserved space for buses is set to 1 (*bp*=1), and bus delay is set to 2 epochs (*bd*=2). The results are shown in Fig. 2.

The results show no difference between the three algorithms for 0% traffic, and that is expected. This is the case whereby there are no cars on the streets and only buses. Similarly, neither approach managed to complete even a single bus trip when there is 90% traffic load. This is an indication that very high street loads mean grid-lock for the public buses.

For both 30% and 60% traffic, the buses are "delayed" by cars. Our $EPTrOn$ solution outperforms both the car biased and the baseline approaches by up to 590% for the load of 30% and by 384% for 60%.

The naive and the car biased approaches have a crossing point for 60% traffic after 70% of the bus routes are completed. This shows that the car biased approach speeds up the buses, too, until a sufficient number of intersections get highly congested and the advantage to the baseline is lost.

Experiment 2 (Figs. 3 & 4): In this experiment we study the robustness of *EPTrOn* against the amount of traffic violators, who ignore the detour directives and go in a direction different than where they were directed to go by the light-boards. Naturally, vehicles start their trips on small neighborhood streets, go through several main streets, and end the trips on small neighborhood streets. We assume that the predominant distribution of destinations with respect to the source of each car is a Gaussian distribution, whereby 40% of the cars go west of their current location, 10% go south, 10% go north, 20% go southwest and the other 20% go northwest, i.e., $\{0.1, 0.2, 0.4, 0.2, 0.1\}$. In this experiment, there are no spots reserved ($bp = 0$) for buses.

We studied three different (violation) traffic distributions, namely *uniform*, whereby an equal percentage of cars goes in all five directions $\{0.2, 0.2, 0.2, 0.2, 0.2\}$, and the two extremes: *center focal point*, whereby all the traffic focuses in one direction $\{0.0, 0.0, 1.0, 0.0, 0.0\}$ and *peripheral focal points* $\{0.25, 0.25, 0.0, 0.25, 0.25\}$. Comparing these three distributions to the normal Gaussian distribution, the amount of violators is 40%, 60%, and 80%, respectively. For example, for the case of uniform distribution, 40% of the cars should have a destination in the west, but it's only 20%, who have it, thus we have 20% violators in that direction only. The other 20% come from the drivers, whose destinations are north or south. The amount of violators sums up to 40%.

The results of the experiment are shown in Fig. 3. The results for $EPTrOn$ are presented with solid lines, while the Naive approach is depicted on dashed lines. Clearly, our solution consistently outperforms the naive, despite the amount of violators. In conclusion, $EPTrOn$ is robust enough to accommodate up to 80% violators.

We also report the average length of car detour. We measured the length of detours for all cars. The Naive approach detour mimics the decisions drivers take when they get stuck in traffic. Specifically, they detour and expect to get onto a less busy road. When *EPTrOn* is used, the detour is defined by the routing directives. The results are depicted on Fig. 4. The difference between the detour penalty for Naive and *EPTrOn* for Gaussian, uniform, center, and peripheral distributions is

1859, 539, 394, and 283 meters, respectively. Our solution does not incur more than 768 meters detour penalty on average (between the 4 distributions).

VI. Related Work

The approach taken in [13] argues that graph weights, based on a single factor, are inefficient when a traffic congestion mitigation solution is provided. The work proposes edge weights that incorporate the three main factors that influence congestion mitigation, namely edge length, road conditions, and average velocity of the vehicles on that edge. Our work uses edge weights that are based on the number of vehicles on it. We use this information further to select the most optimal path, assessing the cumulative number of cars on each path. Moreover, our solution proposes dynamic bus lanes that ensure that mass transit vehicles are not slowed down by other traffic. At the same time, the road network's capacity is not decreased by permanent bus lanes.

Another concept that our solution is built upon is bi-objective optimal path selection. In [14] the proposed approach differs from previous work as it does not rely on the affine combination of the weights for distance and crime risk to amalgamate hybrid weights. Instead, the work uses the concept of skyline routes to identify a subset of paths that strike a balance between distance and crime risk. A similar bi-objective optimization problem is tackled in [15]. Our work differs from [14], [15] in getting data online and providing solutions based on the analysis of the data. Furthermore, our system provides personalized solutions for multiple actors in the traffic simultaneously rather than focusing on a single user.

Several algorithms have been proposed to update the optimal route for a given vehicle in real time. In [16], [17] the authors propose a scheme that balances the traffic on the road network and thus mitigates traffic congestions. The scheme relies on an ad-hoc network whereby vehicles share their velocity with other vehicles and are clustered on a per-edge basis. Furthermore, the weights of the different edges of the road network graph are calculated taking into consideration average vehicle velocity, fuel consumption, and vehicle density. Our approach also uses car density as a metric, but it does not assume that there is a navigation device in each car. The work in [16], [17] does not address the prioritization of mass transit vehicles either.

The focus on reducing emissions from mass transit vehicles has been partially addressed in [18]. The work proposes a system that annotates OSM road graphs with eco-weights that are based on the amount of fuel consumed by buses. The idea is extended into a framework named $EkoMark$ 2.0 that evaluates environmental impact models that are used for defining eco-weights. The framework uses 3D spatial network, GPS trajectories, and actual fuel consumption data from a case-study to evaluate the models. Our solution can be evaluated in $EkoMark$ 2.0, and we also dynamically assign eco-weights to road segments. Moreover, our solution has the bi-objective of creating on-demand virtual bus lanes and providing guidelines for optimal paths to local drivers [19], [20].

VII. Conclusions

In this paper we presented our $EPTrOn$ solution for the on-demand, dynamic creation of virtual bus lanes in congested urban environments. It follows a multi-fold approach, whereby priority scheduling and optimal routing techniques are interwoven to mitigate the pollution caused by internal combustion engine vehicles by clearing the way of the biggest polluters, namely the mass transit diesel-engined buses. Our experimental evaluation shows that our approach increases the throughput of the road network and outperforms the baseline by more than 590% saved travel time on average, while incurring no more than a 768 meters penalty.

Acknowledgments This work was supported, in part, by NSF CBET award #1609120 and NIH award U01HL137159. This paper does not represent the views of NSF or NIH.

References

[1] L. A. D. of Transportation. The automated traffic surveillance and controls (atsc) system. http://trafficinfo.lacity.org/about-atsac.php

[2] C. Signals. San jose connected vehicle pilot. [Online]. Available: https://connectedsignals.com/sjstudy/

[3] M. Chiang, E. Lim, W. Lee, and A. T. Kwee, "Btci: A new framework for identifying congestion cascades using bus trajectory data," in *IEEE Big Data 2017*, pp. 1133–1142.

[4] R. T. Milam, M. Birnbaum, C. Ganson, S. Handy, and J. Walters, "Closing the induced vehicle travel gap between research and practice," *Transportation Research Record*, vol. 2653, no. 1, pp. 10–16, 2017.

[5] G. Duranton and M. A. Turner, "The fundamental law of road congestion: Evidence from us cities," *American Economic Review*, vol. 101, no. 6, pp. 2616–52, 2011.

[6] H. Xu, J. Lin, and W. Yu, *Smart Transportation Systems: Architecture, Enabling Technologies, and Open Issues*, 2017, pp. 23–49.

[7] J. Ouyang. Busgazer. [Online]. Available: busgazer.com

[8] A. U. Shein, P. K. Chrysanthis, and A. Labrinidis, "Slickdeque: High throughput and low latency incremental sliding-window aggregation," in *EDBT 2018*, pp. 397–408.

[9] M. Thorup, "Undirected single-source shortest paths with positive integer weights in linear time," *J.ACM*, vol.46, no.3, pp.362–394, 1999.

[10] A. Guttman, "R-trees: A dynamic index structure for spatial searching," ser. ACM SIGMOD 1984, pp. 47–57.

[11] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," ser. ACM SIGKDD 2011, pp. 316–324.

[12] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: Driving directions based on taxi trajectories," in *ACM SIGSPATIAL 2010*, pp. 99–108.

[13] M. Wei and Y. Meng, "Research on the optimal route choice based on improved dijkstra," in *IEEE WARTIA 2014*, pp. 303–306.

[14] E. Galbrun, K. Pelechrinis, and E. Terzi, "Urban navigation beyond shortest route: The case of safe paths," *Information Systems*, vol. 57, no. Supplement C, pp. 160 – 171, 2016.

[15] G. D. Nunzio, L. Thibault, and A. Sciarretta, "Bi-objective eco-routing in large urban road networks," in *IEEE ITSC 2017*, pp. 1–7.

[16] J. Lin, W. Yu, X. Yang, Q. Yang, X. Fu, and W. Zhao, "A real-time en-route route guidance decision scheme for transportation-based cyberphysical systems," *IEEE VTC 2017*, vol. 66, no. 3, pp. 2551–2566.

[17] ——, "A novel dynamic en-route decision real-time route guidance scheme in intelligent transportation systems," in *IEEE ICDCS 2015*, pp. 61–72.

[18] O. Andersen, C. S. Jensen, K. Torp, and B. Yang, "Ecotour: Reducing the environmental footprint of vehicles using eco-routes," in *IEEE MDM 2013*, vol. 1, pp. 338–340.

[19] C. Guo, Y. Ma, B. Yang, C. S. Jensen, and M. Kaul, "Ecomark: Evaluating models of vehicular environmental impact," in *ACM SIGSPATIAL 2012*, pp. 269–278.

[20] C. Guo, B. Yang, O. Andersen, C. S. Jensen, and K. Torp, "Ecomark 2.0: empowering eco-routing with vehicular environmental models and actual vehicle fuel consumption data," *GeoInformatica*, vol. 19, no. 3, pp. 567–599, 2015.