Multi-Destination Vehicular Route Planning with Parking and Traffic Constraints

Abeer Hakeem
amh38@njit.edu

Narain Gehani
gehani@njit.edu

Xiaoning Ding
xiaoning.ding@njit.edu

Reza Curtmola
reza.curtmola@njit.edu

Cristian Borcea
borcea@njit.edu

Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA

ABSTRACT

This paper aims to provide an efficient solution for people in a city who drive their cars to visit several destinations, where they need to park for a while, but do not care about the visiting order. This instance of the multi-destination route planning problem is novel in terms of its constraints: the real-time traffic conditions and the real-time free parking conditions in the city. The paper proposes a novel Multi-Destination Vehicle Route Planning (MDVRP) system to optimize the travel time for all drivers. MDVRP's design has two components: a mobile app running on the drivers' smart phones that submits real-time route requests and guides the drivers toward destinations, and a server in the cloud that optimizes the routes by finding the most efficient order to visit the destinations. MDVRP uses TDTSP-FPA, an algorithm that finds the fastest route to the next destination and also assigns free curbside parking spaces that minimize the total travel time for drivers. We evaluate MDVRP using a driver trip dataset that contains real vehicular mobility traces of over two million drivers from the city of Cologne, Germany. By learning the spatio-temporal distribution of real driver destinations from this dataset, we build a novel experimental platform that simulates real, multi-destination driver trips. Extensive simulations executed over this platform demonstrate that TDTSP-FPA delivers the best performance when compared to three baseline algorithms.

CCS Concepts

•Information systems → Location based services; **•Human-centered computing → Mobile computing;**

Keywords

Route planning, Multiple Destinations, Parking Assignment, Cooperative System, Mobile App

1 Introduction

The aim of multi-destination route planning is to find the most efficient order of visiting a number of destinations in order to reduce the trip cost, such as the travel time. While this problem has been studied extensively in the context of the Traveling Salesman Problem (TSP) [16, 27], this paper defines a new instance of the problem that is important in real-life. We aim to provide an efficient solution for people in a city who drive their cars to visit several destinations, where they need to park for a while, but do not care about the visiting order. Specifically, the problem's novelty comes from its constraints: the real-time traffic conditions and the real-time free parking conditions in the city. Furthermore, the two constraints influence each other. For example, traffic congestion increases when the drivers cruise looking for parking in a region where all the free parking spaces are already taken.

Managing the interplay between traffic conditions and parking conditions to reduce the travel time for drivers can help both delivery companies and individuals in a city. For example, many times, delivery drivers must park around their destinations (e.g., big buildings) where they need to deliver several packages. An individual, on the other hand, may have a number of tasks to do in a weekend day: grocery shopping, take clothes to/from dry cleaning, stop by the work office to get some papers, and see a small art exhibition downtown. The tasks can be done in any order, and we want to do it as efficiently as possible.

The problem that we need to solve is two-fold: a route planning problem and a free parking assignment problem. To solve it, the paper proposes a Multi-Destination Vehicle Route Planning (MDVRP) system to efficiently plan routes for all drivers in the system. MDVRP optimizes the travel time for all drivers (i.e., plan their routes), while satisfying the free curbside parking conditions (i.e., provide parking guidance). It is cost-effective, as it does not rely on any sensing infrastructure. Its design has two components: a mobile app running on the drivers' smart phones and a server running in the cloud. The app submits real-time route requests to the server, receives optimized routes from the server, and guides the drivers toward destinations. In addition, the app reports to the server when and where a car is parked and when it leaves its parking space. This allows the server to manage the parking information and assign parking spaces to drivers. The server's main job is to interact with the mobile apps of all drivers and to optimize the routes for these drivers to reduce their travel time, while managing traffic congestion. The optimization determines the best order to visit the destinations and finds the best free curbside parking spaces for the drivers.

MDVRP uses TDTSP-FPA, a novel algorithm that combines a solution for the Time-Dependent Traveling Salesman Problem (TDTSP) [22] to find the fastest route for the next destination with our Free Parking Assignment Algorithm (FPA) [13] to find free curbside parking that minimizes the driving plus walking time for all drivers in the system. TDTSP-FPA manages the incoming requests in two steps: first, it finds the shortest path to the next destination in a trip in such a way as to minimize the total travel time. Second, it solves driver contention for the same parking spaces in such a way as to minimize the total travel time for all drivers. The travel time for one driver is the sum of: (1) driving time from the moment the driver submits a parking request to the moment she parks, and (2) walking time from the parking space to the destination and back. TDTSP-FPA's optimization goal is to reduce the total travel time for all drivers.

The main contributions of this paper are:

- We define a new instance of the multi-destination route planning problem, which has significant practical applicability. To the best of our knowledge, this is the first work on route planning that considers simultaneously the real-time conditions of vehicular traffic and free parking availability.
- We propose a novel system, MDVRP, and an algorithm, TDTSP-FPA, to solve this problem. The optimization goal of the algorithm is to minimize the total travel time for all drivers, where this time includes both the driving time to parking spaces and walking time between parking spaces and destinations. The design of MDVRP is modular and, thus, other algorithms for time-dependent route planning and parking assignment can be used to replace TDTSP-FPA.
- We build a new experimental platform for realistic simulations of multi-destination routing. We use real vehicular mobility traces from over two million drivers from the city of Cologne, Germany to learn the spatio-temporal distribution of real driver destinations. Our platform then uses a new method to generate realistic multi-destination route requests, exploiting Cologne's road network along with many destinations and curbside parking spaces in the city's downtown.
- We perform extensive experiments to demonstrate the performance of our system. According to the experimental results, TDTSP-FPA reduces the total travel time by 34% when compared to the solution that represents current driver habits and by 29% and 26% when compared to baseline solutions for TSP and TDTSP, respectively. TDTSP-FPA scales well, as it works better when a larger fraction of drivers in the road network are MDVRP drivers. For example, TDTSP-FPA's travel time reduction compared with TDTSP's is 25% when 5% of drivers are part of MDVRP vs. 19% when only 3% of the drivers are part of MDVRP. The system is robust and provides benefits even when drivers do not comply with the recommended visiting order, but accept the parking assignment.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents an overview of the MDVRP system. Section 4 defines the optimization criteria for time-dependent route planning and free parking assignment, and it describes TDTSP-FPA. Section 5 presents our new experimental platform and the experimental results obtained on top of this platform. Section 6 presents conclusions and future work.

2 Related Work

As urban population grows, cities face many challenges related to transportation, resource consumption, and the environment. Vehicle route planning has been proposed as a strategy to decrease road traffic congestion and implicitly reduce the travel times for drivers. Most of the previous studies on route planning focused on single-destination scenarios. Unlike these studies, our work focuses on a new and practical problem: many drivers have to go to several destinations in a trip, but do not care about the visiting order of these destinations. Furthermore, our problem needs to satisfy real-time constraints regarding vehicular traffic and free curbside parking availability.

The Traveling Salesman Problem (TSP) is a well-known multi-destination route planning problem that aims to find the shortest route (i.e., in terms of distance) that visits each destination once [16]. Although this problem is NP-hard, there is a large number of algorithms that can solve the problem exactly for practical number of destinations or approximately for very large number of destinations. However, these algorithms assume that the travel times are constant throughout the day. The Time-Dependent Traveling Salesman Problem (TDTSP) is a variation of TSP in which the amount of time it takes the salesman to travel from one destination to another fluctuates depending on the time of the day. By allowing the travel time between destinations to vary, the TDTSP can better model real world conditions such as heavy traffic, road repairs, and automobile accidents. We are interested in the time dependent problem introduced by [10, 5, 27] which strives to find the shortest route when the travel time depends on the time of day when the route is traversed.

traveling salesman problem with multiple time windows and time-dependent travel and service time

In these real-world TDTSP problems, there are frequently additional constraints such as time-windows or precedence constraints. TDTSP with time windows [21] deals with finding a set of optimal routes for a fleet of vehicles in order to serve a set of customers, each one with a specified time window. Hurkala [15] proposes a novel algorithm that computes the minimum route duration for the TDTSP with multiple time windows and time-dependent travel and service/visit time constraints. Different constraints are addressed in Huang et al. [29] to efficiently plan a route that satisfies deadlines and cost requirements. The work finds an objective-optimized route where the user-specified destinations are visited before their corresponding deadlines. It also considers multiple deadlines for multiple destinations as well as optimizing the trip cost simultaneously. Melagarejo et al. [20] proposes a set of benchmarks for TDTSP based on real traffic data and shows the importance of handling time dependency in the problem. The authors present a new global constraint (an extension of no-overlap) that integrates time-dependent transition times and show that this new constraint outperforms the classical Constraint Programming approach.

In addition to academic research, route planning apps such as Route4Me and GSMtasks [1, 2] aim to optimize driver's route when traveling to multiple destinations. These apps are able to efficiently manage driver fleets as well as business and delivery drivers.

None of these works considers finding free curbside parking for drivers and does not consider the influence parking availability and parking locations on the traffic conditions. To the best of our knowledge, our MDVRP system is the first work on multiple-destination route planning that considers real-time parking and traffic conditions for multiple destinations, while optimizing the total travel time for all drivers.

MDVRP chooses to learn the parking information from the drivers because infrastructure such as ParkNet [19] and SFpark [3], which is installed to detect and monitor the availability of curbside parking, is expensive to deploy and maintain. Nevertheless, MDVRP can integrate information from other parking monitoring solutions. For example, Nawaz et al. [23] proposed a smart phone based sensing system that leverages the ubiquity of WiFi beacons to monitor the availability of street parking spaces. Salpietro et al. [26] developed Park Here!, a smart curbside parking system based on smart phone-embedded sensors and short range communication technologies. Arnott and Rowse [6] developed an integrated model for curbside parking and traffic congestion control in a downtown area.

Most previous research on parking in a city focused on managing information about parking availability and sharing it with the drivers, but let the drivers make their own parking decisions. However, this still leads to traffic congestion because multiple drivers will attempt to park in the same space. A better approach would be to assign the parking spaces automatically to the drivers and let them concentrate on their trips. Several works studied parking assignment solutions from different perspectives. Boehle [9] presented a centralized reservation system. A parking service constantly gathers traffic data from participating vehicles. This data is then used to determine time-optimal routes from the vehicles' current position to the parking spaces. Mackowski et al. [18] developed a demand-based real-time pricing model to optimally allocate parking spaces in busy urban centers. In [13], we introduced FPA for on-the-fly curbside parking assignment. Unlike other parking assignment systems, FPA adapts on-the-fly to new parking requests and optimizes parking space allocation to maximize a social welfare objective (i.e., minimizing the total travel time for all drivers). However, none of these works was designed to optimize parking space allocation for multiple-destination route planning over time. MDVRP leverages FPA in its TDTSP-FPA algorithm to optimize the travel time for all drivers in our system.

3 System Overview

This section presents an overview of MDVRP, focusing on how to plan a multi-destination route that satisfies real-time traffic and parking conditions. MDVRP aims to reduce the total travel time (i.e., driving and walking times) for all drivers in the system.

Figure (1) shows the system design of MDVRP system, which consists of two components, namely *Driver Manager* (DM) and *Route Planning Manager* (RPM).

DM is a mobile app that runs on each driver's smart phone, which consists of three modules: *driver request initiator*, *tracker*, and *driver guidance*. DM is in charge of submitting a multi-destination route request and reporting parking status to the RPM. Once it receives a route from RPM, it guides the driver in their trip. The reporting of parking status relies on the app, which can learn this status from an activity recognition service running on the phone [11].

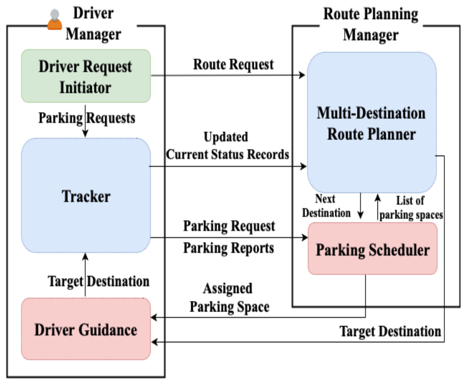

Figure 1: MDVRP System Overview

RPM runs on a central server and consists of two modules, the *multi-destination route planner* and the *parking scheduler*. RPM manages the incoming route requests, aggregates the DM parking reports to determine the available parking spaces, and provides multi-destination route planning services to drivers. The services are invoked upon the initial request for trip planning from a driver, and are re-invoked at each destination to plan the remaining route for the driver based on her current location. The TDTSP-FPA algorithm running at RPM combines a solution for the Time-Dependent Traveling Salesman Problem (TDTSP) [22] to find the fastest route for the next destination with our Free Parking Assignment Algorithm (FPA) [13] to find free curbside parking that minimizes the driving plus walking time for all drivers in the system. MDVRP is designed to first consider traffic conditions, and then consider the parking conditions, as drivers approach their destinations.

We now describe the lifecycle of a multi-destination route request in MDVRP, from generation to completion. When a driver submits a request, the *driver request initiator* on her phone generates two types of requests: a route request and several parking requests (corresponding to the multiple destinations). The route request contains the destinations chosen by the driver and the driver's current-status record, *i.e.*, (driver's current road segment, position on road segment, observation time). The route request is sent to RPM, where all incoming route requests are streamed into a queue by the *multi-destination route planner* module and are processed on a first-come-first-serve basis. The parking requests are forwarded to the *tracker* at the DM, which sends them individually to the *parking scheduler* at the RPM each time the driver approaches a new destination and needs a parking space near that destination. The number of parking requests equals the number of the driver-specified destinations. Each parking request contains a driver-specified destination and the amount of time the driver wants to spend at the destination (*i.e.*, parking duration).

The *multi-destination route planner* manages and serves incoming route requests. It plans routes in a way that optimizes the total travel time. Specifically, for each route request, it uses a time-dependent graph representation of the road network and applies a Time-Dependent Traveling

Salesman Problem (TDTSP) solution to compute the fastest path between two given locations. The travel time over a road segment depends on its traffic congestion status, which in turn depends on the time instant at which the road segment is traversed. Thus, knowledge about real-time traffic information over the road network is required. Even though speed profiles extracted from history data can provide a good estimation of long-term traffic dynamics, the short and mid-term forecast of travel times on road segments, particularly the time instant at which the segments are traversed must be made dynamically. Thus, we obtain the time cost of a road segment from existing open source historic trajectory data [28] and real-time traffic information from drivers who are part of our system (*i.e.*, MDVRP drivers) [24]. As shown in [14], drivers' smart phones can form a traffic sensing infrastructure, and a 2-3% penetration of smart phones in the driver population is enough to provide accurate measurements of the velocity of the traffic flow.

The initial routes determined by TDTSP are adjusted after visiting each destination based on the locations of available parking spaces around the next destination. This is done to minimize the total cost of traversing the route, which includes the time spent on both driving to parking spaces and walking to destinations from parking spaces. Since parking spaces may be taken without notice by drivers who are not part of MDVRP, we consider the (k) closest parking spaces to each destination when computing the routes. To select the next destination, the *multi-destination route planner* averages the travel times between driver's origin location and the k selected parking spaces around each destination. It then selects the destination with the shortest average travel time. Once the next destination is computed, the corresponding route and the destination are sent to the DM' s *driver guidance* module.

Given the driver's next target destination, the *driver guidance* module guides the driver to the destination. It also forwards the destination to the *tracker*, which then submits a parking request to the *parking scheduler* when the driver approaches the target destination. If the parking request is sent when the driver is far away from the destination, drivers who are not part of our system (*i.e.*, unsubscribed drivers) have a high likelihood of taking the assigned space. If the request is sent when the driver is too close to the destination, the system may not be able to find a parking space close enough to the destination.

Therefore, as the driver approaches the target destination, we use a Request Distance (see Figure (2)) to determine when the driver's parking request has to be sent by the *tracker* to the *parking scheduler* in order to be assigned a parking space. This distance defines a circle centered around the destination and its radius was determined experimentally to be initially set to the average length of the roads within the whole region (i.e., zip code). The radius can be adjusted periodically based on the parking occupancy rate in the area which is learned from the RPM (*i.e.*, the radius is increased when the occupancy becomes higher). RPM may over-estimate the number of available parking spaces as it uses only information from MDVRP drivers. This is because unsubscribed drivers may take parking spaces presumed to be available by our system. This problem is solved in our previous work [13] based on keeping track of spaces occupied by unsubscribed drivers and on avoiding assigning these spaces for a period of time.

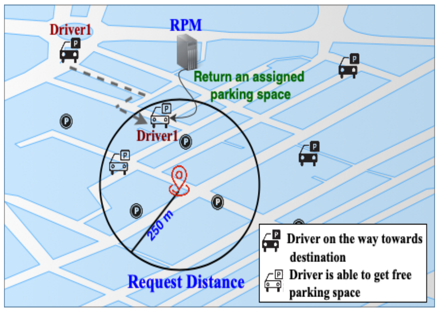
Driver1
RPM
Return an assigned parking space
Driver1
250 m
Request Distance
Driver on the way towards destination
Driver is able to get free parking space

Figure 2: Parking Search Region

After receiving the parking request, the *parking scheduler* enqueues it for parking scheduling and assignment. The parking assignment decision is made once the Request Distance is reached in such a way as to minimize the total travel time (driving and walking times) of all drivers in MDVRP. The parking assignment algorithm is described in Section 4. Once the driver parks in the assigned space, the *parking scheduler* deletes the parking request from the queue. The *tracker* reports the status of the parking space to the *parking scheduler* when the driver is going to either park at or leave the assigned space. When the driver leaves the space, the *tracker* also updates the driver's current-status record and sends it to the *multi-destination route planner* to find the fastest path toward the next target destination in the trip. The aforementioned process is repeated until all the driver-specified destinations are visited.

Both the *multi-destination route planner* and the *parking scheduler* aim to minimize the total travel time; however, the *multi-destination route planner* minimizes the travel time toward the next destination (up to the Request Distance) for each driver. Then, once the Request Distance is reached, the *parking scheduler* minimizes the total travel time (driving from the Request Distance to the parking space and walking from the parking space to the destination and back) for all the drivers.

The design of our MDVRP system is modular and, thus, other time-dependent route planning and parking assignment algorithms can be used. Even though we use the TDTSP's point-to-point shortest path algorithm [22] and the Free Parking Assignment algorithm (FPA) [13], they can be replaced with other such algorithms.

4 Travel Time Optimization

We consider the multi-destination route planning problem with parking and traffic constraints defined as follows. Given a sequence of route requests ordered by generation time, we aim to serve each request by finding the fastest route leading drivers to their destinations while considering the real-time traffic and providing free parking assignment service at each destination in the route.

The salient character of our problem lies in that we aim to reduce the total travel time of all drivers as much as it is practically possible. The travel time for each driver is split into: 1) The driving time from the current location to the parking's Request Distance of the next target destination; 2) The driving time from the moment the driver reaches the Request Distance to the moment it parks; 3) The walking time between the parking space and destination (forth and back).

To achieve this goal, we develop the TDTSP-FPA algorithm. TDTSP-FPA uses a solution to TDTSP to solve a multi-destination route planning problem in such a way as to minimize the travel time toward destinations (point 1 above). FPA solves drivers' contention for the same parking spaces in such a way as to optimize the total travel time to each destination in their trips (points 2 and 3 above). TDTSP helps FPA in the sense that it finds the fastest route that avoids traffic congestion to the destination, which implicitly means it is easier to find a parking space along the path. FPA helps TDTSP by reducing the traffic congestion due to cruising while looking for parking.

4.1 Optimization Formulation

The optimization objective for our problem is to reduce the total travel time for all drivers. Specifically, the problem targets a set of requesting drivers $V = \{v_1, v_2, ..., v_n\}$; and each driver v_i has a set of target destinations $D = \{d_1, ..., d_z\}$. When planning routes, we also consider curbside parking spaces, which are denoted by $S = \{s_1, s_2, ..., s_m\}$, and the parking occupancy periods for each destination, which are described by w_{s_j}, i.e., the time duration that parking space s_j will be occupied by a driver and cannot be utilized for any other driver.

The drivers are assumed to be moving independently based on legal speeds and on the congestion levels on different road segments. All the geographical locations, including the addresses of destinations and the locations of parking spaces, are converted into latitude and longitude coordinates in the system.

The optimization solves two problems together, TDTSP and FPA, which are as described as follows.

4.1.1 TDTSP Definition

TDTSP is a well-known route planning problem for multiple destinations. TDTSP extends the original Traveling Salesman Problem (TSP) with the specific goal of finding the fastest connection on time-dependent road networks. The travel time on the road networks depends on the traffic congestion. All drivers travel along a road network that is modeled as a directed graph $G(N, E)$. Each directed edge $e \in E$ represents a road segment and each node $n \in N$ represents the intersection of two or more roads. Given a segment e_i, it takes time t_i for a driver to travel from one intersection to another along e_i. Note that traffic conditions represented by t_i can be incorporated in the model by introducing weights on graph edges [24]. If a trip begins or ends in the middle of a road segment, we approximate the location to the nearest intersection node. This approximation works well in our city settings, where the road segments are a mix of medium-length and short.

Next, we formally define the concept of path, travel time function, timed-path in a graph, travel time of sub-tour, and we then give an alternative formal definition of the TDTSP.

Definition 1 (Path). *A path $P = (n_1, ..., n_k)$ in a graph $G = (N, E)$ is a sequence of nodes such that $(n_i, n_{i+1}) \in E, \quad \forall i \in \{1, ...k-1\}, k \geq 2$.*

Definition 2 (Travel Time Function). *A travel time function $f : E \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$ is a function such that for a given edge $(n_i, n_j) \in E, f(n_i, n_j, t)$ is the travel time from n_i to n_j when leaving n_i at time t.*

The travel time function dynamically associates travel times to road segments at the time when the segment is traversed, *i.e.*. MDVRP does this based on historical speed profiles as well as frequent updates received from drivers in the system.

Definition 3 (Timed-Path). *Given a graph G=(N,E), a path starting time $\tau \in \mathbb{R}^+$ and a travel time function $f \colon E \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$, a timed-path $P_{\tau,f}$ in G is a path $(n_i, ..., n_k)$, in which each node n_i has an associated start time $t(n_i, P_{\tau,f})$ such that:*

$$t(n_i, P_{\tau,f}) \geq \tau, \quad \forall i \in \{1, ..., k\}$$

$$t(n_{i+1}, P_{\tau,f}) \geq t(n_i, P_{\tau,f}) + f(n_i, n_{i+1}, t(n_i, P_{\tau,f}))$$

Next, we define the travel time to parking, which is the time between the current location of the driver (origin or current parking space) and its next parking space (i.e., for the next destination). Recall that we do not know which parking space will be available when the car approaches the next destination, and thus consider the k closest parking spaces to the destination in our system.

Definition 4 (Travel Time to Parking.) *Given a graph G=(N,E), two nodes (n_i, n_j) that represent a driver's current location (n_i) and the next target destination (n_j) in a driver's route, a current time t, and a travel time function $f : E \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$, a travel time to parking T_{ij} is the average of the minimum costs (i.e., time) timed-paths between the origin n_i and the k available parking spaces closest to the destination n_j.*

MDVRP calculates the k available parking spaces at the time the vehicle is ready to drive toward the next destination (i.e., MDVRP does not predict the parking availability at the time the vehicle arrives at destination). The parking spaces are calculated based on the occupancy period w_{s_j} and the travel time to the parking space s_j from the current location n_i. If by the time the driver approaches the destination, some of the k parking spaces become unavailable, MDVRP is able to adapt and find other parking spaces.

Definition 5 (TDTSP). *Given a graph G=(N,E), a path starting time $\tau \in \mathbb{R}^+$, a travel time function $f : E \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$, and a timed-path $P_{\tau,f}$, TDTSP finds the fastest route which starts from the origin $(n_1 = o)$ and visits each destination exactly once. The route is computed using the travel times to parking, T_{ij}, computed between each pair of (n_i, n_j) nodes.*

4.1.2 FPA Definition

A parking assignment of spaces to drivers is defined as Y: $V \rightarrow S$, where y_{ij} is the assignment of a driver $v_i \in V$ to a parking space $s_j \in S$:

$$y_{ij} = \begin{cases} 1, & \text{if } v_i \text{ is assigned to } s_j \\ 0, & \text{otherwise} \end{cases} \quad 1 \leq i \leq n, 1 \leq j \leq m \quad (1)$$

$$\sum_{i=1}^{n} y_{ij} \leq 1, \quad 1 \leq j \leq m \ (i.e., s_j \in S) \qquad (2a)$$

$$\sum_{j=1}^{m} y_{ij} = 1, \quad 1 \leq i \leq n \ (i.e., v_i \in V) \qquad (2b)$$

Constrains 2a and 2b ensure that a driver receives at most one space and that a space is not assigned to more than one driver, respectively. Further, the space is forbidden to be reassigned during the time occupancy period of the current assignment, such that the assignments do not overlap.

For a set of drivers and a set of parking spaces, there may exist a large number of assignments. The algorithm seeks to find an assignment that can minimize the total travel time (driving and walking) of the drivers to each destination in their trips. The travel time $T(v_i)$ toward one destination in a driver v_i's trip is calculated in real-time and consists of two parts, the driving time and the walking time:

- T_d (O_{v_i}, s_j) is the driving time of driver v_i from the moment she submits her request from location O_{v_i} until she parks at the parking space s_j.
- T_w (s_j, d_{v_i}+s_j) is the walking time of the driver between the parking space s_j and the destination d_{v_i} (forth and back).

4.2 A Solution for the TDTSP

In the RPM component of our system, we deploy the time-dependent point-to-point shortest path solution [22] to compute a timed-path with minimum travel time to the next destination. This is a bidirectional search algorithm on time-dependent road networks, based on the A* algorithm. The given network is modeled as a directed graph with time-dependent travel time functions for all edges. The algorithm procedure leverages a modified generalization of Dijkstra's algorithm, made bidirectional and improved in several aspects. As for the backward search in A*, the arrival times are not known in advance. Thus, the reversed graph has to be weighted by a lower bound cost (constant travel time for all time instants i.e., edge length/maximum speed limit).

Given a graph G=(N,E) and origin and destination nodes o, d ∈ N, the algorithm for computing the fastest o-d path works in three phases.

1. A bidirectional A* search occurs on G, where the forward search is run on the graph weighted by the travel time function, and the backward search is run on the graph weighted by the lower bound cost. All nodes settled by the backward search are included in a set M. Phase one terminates as soon as the two search scopes meet.
2. Suppose that node $n \in N$ is the first node in the intersection of the forward and backward searches, where a time cost of the path going from o to d passing v is an upper bound cost of the path of (o, d, t). In the second phase, both search scopes are allowed to proceed until the backward search queue contains only nodes associated with costs less than the upper bound. Again, all nodes settled by the backward search are added to M.
3. Only the forward search continues, with the additional constraint that only nodes in M can be explored. The forward search terminates when d is settled.

4.3 The FPA Algorithm

The *parking scheduler* component runs the FPA algorithm periodically to assign parking spaces to outstanding parking requests in the queue. We determined experimentally, based on simulations, that running FPA every 2 seconds provides a good trade-off between performance and overhead. In each period, FPA first pre-allocates to the driver of each outstanding request an available parking space that is closest to her destination. The pre-allocation adapts the solution to the flow problem described in [7] to minimize the total walking time of these drivers. The actual assignment of parking spaces takes place based on the urgency of the demands for parking spaces, which is measured by how close the corresponding drivers are to their destinations or their pre-allocated parking spaces. Specifically, in each period, the drivers with the most urgent demand (i.e., they may pass their destination if a parking assignment is not made quickly) are selected and their pre-allocated parking spaces are officially allocated to them. For more details, we direct the reader to a description of the FPA algorithm [13].

Algorithm 1 TDTSP-FPA Pseudo-code Executed for Each Visited Destination

```
1: Phase one
2: Input: a driver's origin $o_v$, set of target destinations
   $D_v = d_1, ..., d_z$, a value $k$ for the closest parking spaces
   to each destination, and a starting time $\tau$
3: $curr\_orig \longleftarrow o_v$ // current origin of the trip
4: $rem\_D_v \longleftarrow D_v$ //set of remaining destinations to be
   visited
5: for each destination $d_v^i \in D_v$ do
6:    Define a list of $k$ parking spaces $L_{d_v^i}$ which are the
      closest available spaces to $d_v^i$ at the approximate time
      of arrival to $d_v^i$
7:    $Origin\_set \longleftarrow D_v - d_v^i + curr\_orig$
8:    for each parking space $s_j \in L_{d_v^i}$ do
9:       for each $o$ in $Origin\_set$ do
10:         Compute travel time $\alpha_o^{s_j}$ of the timed-path be-
            tween $o$ and $s_j$ at time $t$
11:      end for
12:   end for
13:   for each $o$ in $Origin\_set$ do
14:      Compute the travel time to parking $T_i$ between $o$
         and $d_v^i$ by averaging the travel times $\alpha s_{j_o}$ between
         $o$ and the $k$ parking spaces
15:   end for
16: end for
17: $fastestRoute \longleftarrow TDTSP(D_v, T)$
18: Send first destination, $d$, in $fastestRoute$ to FPA pro-
    cedure to assign parking space
19: Phase two //executed once the driver reaches the Re-
    quest Distance for parking assignment
20: Input: a driver's current location $c_v$ and the destination
    $d$
21: Create the list of current available parking spaces $L_d$ in
    the proximity of $d$
22: $s_v \longleftarrow FPA(c_v, d, L_d)$ //assigned parking space for
    driver $v$
23: Guide $v$ to $s_v$.
24: $rem\_D_v \longleftarrow rem\_D_v - d_v^i$
25: $curr\_orig \longleftarrow s_v$
```

4.4 The TDTSP-FPA Algorithm

The procedure of serving drivers' request in TDTSP-FPA algorithm is divided into two phases, as shown in Algorithm 1, and each phase requires a list of parking spaces that are located in a destination's region. These lists are static, as defined by the municipality data on streets with free curbside parking. Therefore, for each destination, we define an ascending list of parking spaces offline where each parking space is ordered according to the road distance to its associated destination.

The first phase invokes the TDTSP procedure to find the shortest route that starts from a driver's current origin and visits all the destinations once in such a way as to minimize the total travel time. We compute the travel time to parking according to Definition 4 (lines 5-16 in Algorithm 1) for each pair of nodes in the graph (i.e., the union of current origin and the set of remaining destinations not visited yet). Then, we apply TDTSP according to Definition 5 (line 17), and select the first destination in the fastest route generated by TDTSP (line 18). This will be the next destination, for which FPA will assign parking. In order to reduce the time spent on computing paths, we re-use the paths that have been computed in the past x minutes for drivers who share the same locations and destinations, where x is determined experimentally. In this second phase, the FPA procedure is invoked when a driver reaches the Request Distance (line 22). Once a parking space is assigned, the driver's phone will guide the driver toward this space (line 23). Lines 24-25 update the set of visited destinations and sets the new current origin of the driver. The whole algorithm is executed again to determine the next destination after the parking duration at the current destination expires.

5 Experimental Evaluation

We have evaluated the performance of MDVRP and TDTSP-FPA algorithm using simulation with real traffic traces in a real-world road network, which provide us with realistic constraints in terms of traffic and parking.

5.1 Evaluation Goals

Our evaluation aims to determine:

- The overall effectiveness of the TDTSP-FPA algorithm on reducing *the average travel time.* The travel time of a driver includes the time spent on driving to the assigned parking locations and the time spent on walking from the parking locations to destinations and then back to the parking locations. It does not include parking duration. The travel times of all drivers in each experiment are averaged to reflect the overall performance.
- Contributions of *driving time* and *walking time* in the total reduction of travel time.
- The *scalability of TDTSP-FPA*, as the percentage of the MDVRP's drivers among all drivers on the roads increases.
- The effectiveness of MDVRP on reducing the travel times of individual drivers. We want to know how many drivers use less time to finish their trips and how many drivers spend more time when MDVRP is used. We calculate the *improvement rate*, which is the proportion of drivers with travel time reduced by MDVRP, to reflect its effectiveness.
- The robustness of the system under a varying compliance rates (i.e., percentage of drivers who follow the suggested visiting order).

5.2 Comparison Algorithms

- *Highest Transition Probability Order (HTPO)* represents human mobility habits without careful route planing: a driver always picks the destination that is closest to her current location as her next stop.
- *Traveling Salesman Problem (TSP)* is a classical routing strategy that aims to minimize the total travel distance; it does not consider any constraints. The problem is NP-hard, but a heuristic algorithm for solving the TSP problem is used in the experiments [12].
- *Time-dependent Traveling Salesman Problem (TDTSP)* uses travel time as a metric to select the shortest path between driver's origin and destination that yields the provably fastest route. Paths can be evaluated by considering simply point-to-point shortest paths [22] and real-time traffic density on the road segments [24].

In HTPO, TSP, and TDTSP, a driver searches for the closest free parking spaces using breadth-first search.

5.3 Experimental Platform

5.3.1 Real-World Traffic and Road Network Dataset

We use the TAPAS Cologne driver trace [28], which contains the traffic records of over two million drivers in the city of Cologne, Germany during a period of tow hours from 6:00 am to 8:00 am. Each trip record includes a departure time, an origin location and a destination (the IDs of the corresponding road segments), and the route from the origin to the destination. We map the trips to the road network in the same city, which contains 31,584 road intersections and 71,368 road segments.

5.3.2 Request Generation

The requests used to drive the simulation are derived from the trip records in the TAPAS Cologne dataset. This process allows us to 1) control the number of drivers in simulation experiments; 2) select only the destinations in Cologne downtown, which is the most congested area in the city, since we are most interested to evaluate TDTSP-FPA in crowded areas with enough vehicular traffic and contention for parking; and 3) have requests with multiple destinations in simulation experiments.

To generate realistic route requests with specific departure times and multiple destinations, we use the method proposed in [17]. We first divide the trips in the dataset into short-time bins, denoted by b_i and denote all road segments by r_i. Then all trips are assigned into bins based on the departure time of the trips. We assume that the destinations of trips on each road segment approximately follow a Poisson distribution [25] during time frame f_j, where each frame has a fixed length spanning L time bins. Thus, the Poisson distribution parameter λ_{ij} is computed for each road segment r_i during time frame f_i. Specifically, for each road segment r_i, we count the number of trips that originated from r_i within time frame f_i, denoted by c_{ij}, and learn the probability distribution of the destination road segments of these trips, denoted by p_{ij}. Then, we calculate λ_{ij} based on c_{ij} using Equation (3) and generate a target route request that follows a Poisson process.

$$\lambda_{ij} = c_{ij}/L. \tag{3}$$

For each route request generated in frame f_i with the origin road segment r_i, a destination road segment is generated according to the probability distribution p_{ij}. We only consider the destination road segments with high probability distribution in the Cologne downtown area to ensure enough vehicular traffic and enough contention for parking spaces. Note that the dataset only reveals one destination in a trip; however, in reality there are more destinations. To keep the characteristics of a realistic scenario, we repeat the operation and select from the list more trip records with the origin of each trip record being the destination of the previous trip record.

Since trip records are selected according to the probability distribution, they reflect the real distribution of trip destinations in Cologne downtown and the mobility patterns of the drivers. Also note that the drivers that submit requests are not the only drivers in the road network in the simulation, since background traffic is also included in the simulation, as we will discuss in this section. The route requests contain only the trips that we are interested to evaluate.

The route requests have different numbers of destinations (e.g., 1∼7). We set 40% of the routing requests to the largest number of destinations to induce more traffic congestion and to resemble the case of delivery drivers. The rest of the requests are set with fewer destinations to resemble individual drivers. For example, in an experiment with 1∼4 destinations, 40% of requests are set with 4 destinations, 30% with 3 destinations, 20% with 2 destinations, and 10% with one destination. To obtain a diverse workload, different simulations have different upper limits.

The length of parking duration is randomly chosen within [10 min, 25 min], to keep the duration reasonable. Note, the time needed to walk from the parking location to the destination and back to the parking location is not included in the parking duration, as it is an important factor in our optimization objective.

We set the value of k, the number of closest available parking spaces to each destination considered in TDTSP, to 3. We found that a small value of k is sufficient to deal with the problem of parking spaces taken by cars that are not part of MDVRP, while avoiding an increase in the computation time. Furthermore, k cannot be very large in order to ensure that the parking spaces are close to the destinations.

5.3.3 Simulation Setup

We use SUMO [8] to run vehicular traffic simulations, and use TraaS [4] to send commands to drivers and direct them in their routes. We use the NetEdit tool in SUMO to create travel destinations and parking spaces on the Cologne map. The total number of parking spaces around the destinations is 2400.

To simulate the scenarios with real traffic conditions, we varied the background traffic by including different numbers of additional drivers (40k∼80k). These drivers make single-destination trips, which are randomly selected from the TAPAS Cologne dataset. Background traffic is introduced because we do not assume that all or even a large fraction of drivers will use the MDVRP system. However, we assume that MDVRP drivers are generally representative of the entire driving population.

The background traffic simulates realistic traffic conditions, but it is not used for parking contention for two reasons. First, we selected only a small number of parking spaces for the drivers that we control; there are many more parking spaces that could be used by drivers in the background traffic. Second, we are not interested to evaluate the effect of unsubscribed drivers (i.e., drivers not subscribed to MDVRP) on parking contention in this paper. We proposed a solution to this problem elsewhere [13].

All experimental results show averages over five runs.

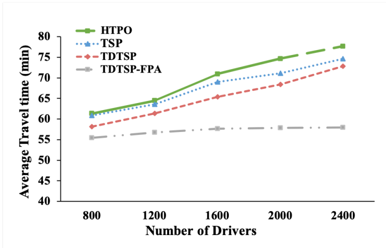

Figure 3: Average Travel Time with Different Number of Drivers and Number of Destinations [1∼4]

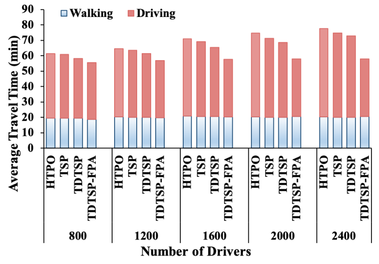

Figure 4: Walking and Driving Time for Different Numbers of Drivers and Numbers of Destinations [1∼4]

5.4 Experimental Results

Figure (3) compares the performance of HTPO, TSP, and TDTSP with TDTSP-FPA with the number of drivers varied from 800 to 2400. The background traffic is generated with 60K drivers. As the figure shows, TDTSP-FPA outperforms the competing solutions consistently, and its performance advantage is more prominent when the number of drivers increases. When the number of drivers is 2400, TDTSP-FPA reduces the average travel time by 34%, 29%, and 26%, respectively, compared to HTPO, TSP, and TDTSP. The results demonstrate the substantial impact MDVRP can have on driving and parking in the cities.

The figure also shows that the average travel time grows quickly for HTPO, TSP, and TDTSP when the number of drivers increases. There are two reasons for this behavior. First, traffic conditions are not considered in HTPO and TSP; thus, they may select congested road segments. The comparison between TDTSP and TSP shows the benefits from taking traffic conditions into consideration. Second, drivers in HTPO, TSP, and TDTSP need to travel more to search for parking, which further increases traffic congestion. TDTSP-FPA directs drivers to parking spaces that are likely to be available. Thus, drivers travel shorter distances looking for parking spaces. This reduces not only their travel time but also the traffic in the road network.

Figure (4) breaks down the travel time into two parts: driving time and walking time. The figure shows that drivers spend most time on driving and TDTSP-FPA reduces the average travel times by mostly reducing the driving time. With 2400 drivers, TDTSP-FPA can reduce driving time by up to 54%. Reducing the driving time is very important, as this reduces traffic congestion and implicitly the gas cost and pollution. Since the number of parking spaces in the centroid area is limited, TDTSP-FPA can hardly reduce walking time. We expect that, with the technology developing toward self-driving cars that can drop off drivers at the locations closest to their destinations, the impact of walking time can be ignored in the future. In such a scenario, a self-driving car finds its way to the assigned parking space after dropping off its passenger.

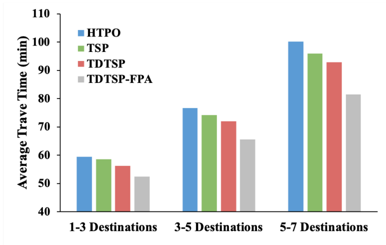

Figure 5: Average Travel Time with Different Number of Destinations and Constant Number of Drivers (1200)

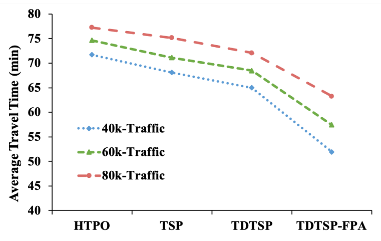

Figure 6: Average Travel Time for 2000 Drivers with Different Patterns of Background Traffic and Number of Destinations [1∼4]

The next set of experiments investigate how the travel times change when the number of destinations is varied. Figure (5) shows the average travel time for 1200 drivers and 60K background traffic drivers. As the figure shows, TDTSP-FPA reduces the average travel time by larger percentages when the number of destinations increases. For the experiments with 1∼3 destinations, TDTSP-FPA reduces the average travel time by 13% and 7%, respectively, relative to HTPO and TDTSP. For 5∼7 destinations, the percentages increase to 23% and 14% respectively. TDTSP-FPA shows more advantage with more destinations in each trip not only because the traffic in the road network increases, but also because there is more optimization space for TDTSP-FPA to improve parking performance.

We have also investigated how TDTSP-FPA scales when the percentage of MDVRP's drivers increases. To model this scenario, we varied the number of background traffic drivers and kept the number of MDVRP's drivers constant at 2000. The background traffic is generated with 40K, 60K, and 80K drivers. Figure (6) shows that TDTSP-FPA decreases the average travel time by 25%, 19%, and 14%, relative to TDTSP, for 40k, 60k, and 80k background drivers, respectively. We observe that TDTSP-FPA scales well, as it reduces the average travel time by larger percentages when the percentage of MDVRP's drivers increases. With more MDVRP drivers, TDTSP-FPA can collect more information from these drivers and affect the traffic more effectively. These results confirm what we observed in Figure (3), where we varied the number of MDVRP's drivers, but kept the number of background drivers constant.

While the reduction of average travel time reflects the overall benefits for the drivers in the road network, we also want to find out if most individual drivers spend less time for their trips. Thus, for each driver, we calculate an improvement ratio between the travel time obtained with TSP and the travel time obtained by TDTSP-FPA. A ratio higher than 1 indicates that the driver has benefited from TDTSP-FPA and spent less time with TDTSP-FPA. Then, we sort the drivers based on their ratios, and show the ratios in Figure (7). In the experiments, there are 2000 MDVRP drivers

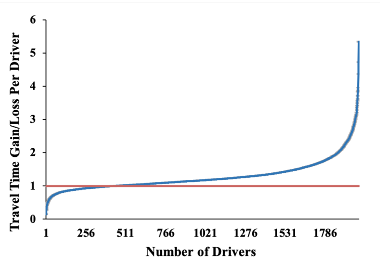

Figure 7: Distribution of Travel Time Gain/Loss for 2000 Drivers in the System with Number of Destinations [1∼4]. Gains are Values Greater Than 1, and Losses are Values Less Than 1

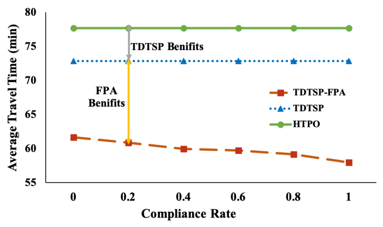

Figure 8: Average Travel Time as a Function of the Compliance Rate for 2400 Drivers with Number of Destination [1∼4]

with 1∼4 destinations and 60k drivers in background traffic.

As shown in the figure, TDTSP-FPA manages to reduce the travel time for a large majority of drivers (over 85%). However, there are still some drivers who cannot experience improvements. In real-life, these drivers may not know that their time increased, but a few bad experiences could impact the system adoption. Thus, we plan to investigate limiting the number of drivers who experience performance losses and bound performance loss to avoid the worst user experiences.

While it is in the drivers' interest to follow the MDVRP's guidance, it is possible that some drivers will not comply with the guidance (i.e., they will not follow the recommended visiting order of destinations). Therefore, we vary the compliance rate (percentage of drivers who follow the recommended visiting order) to test the system robustness. In this experiment, all drivers (including the non-compliant ones) accept the FPA parking assignments. The non-compliant drivers follow their own routes, according to HTPO. Figure (8) indicates that MDVRP is robust; compared to TDTSP and HTPO, TDTSP-FPA still offers good improvement, even under a low compliance rate. This is due to the fact that, even at a 0% compliance, drivers still receive benefits from FPA, which in turn can improve the travel time. Conversely, at the higher compliance rate, both FPA and our updated version of TDTSP provide benefits to drivers. The figure shows that the FPA benefits range from 19% to 27%, and the TDTSP benefits are 7% when compared to HTPO.

6 Conclusion and Future Work

This paper has addressed a novel problem, namely multi-destination route planning with parking and traffic constraints. This problem has practical applications in many real-life situations, such as package delivery or people visiting multiple destinations in one trip. We formulated this problem analytically in order to optimize the travel time for all drivers. To solve the problem, we designed a novel system, MDVRP, which finds the sequence of destinations that result in the shortest driving and walking time for the drivers. To the best of our knowledge, this is the first work on multi-destination route planning that considers real-time traffic and parking conditions to optimize the travel time for all drivers in the system. We evaluated the optimization algorithm of MDVRP, namely TDTSP-FPA, over a new and realistic experimental platform that leverages millions of real-life vehicular traces. The experimental results demonstrated that TDTSP-FPA outperforms the comparison baselines, scales well when the number of drivers in MDVRP increases, and is robust to non-compliant drivers. For future work, we plan to optimize the travel time by considering destination arrival deadlines as an additional constraint to our problem.

7 Acknowledgment

This research was supported by the U.S. National Science Foundation (NSF) under Grants No. DGE 1565478, SHF 1617749, and CNS 1801430. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

8 References

[1] https://route4me.com/.

[2] https://gsmtasks.com/.

[3] SFpark. http://sfpark.org/.

[4] TraCI-TraaS. http://https://sumo.dlr.de/wiki/TraCI/TraaS/.

[5] H. Abeledo, R. Fukasawa, A. Pessoa, and E. Uchoa. The Time Dependent Traveling Salesman Problem: Polyhedra and Branch-Cut-and-Price Algorithm. In *Experimental Algorithms*, pages 202–213. Springer Berlin Heidelberg, 2010.

[6] R. Arnott and J. Rowse. Downtown Parking in Auto City. *Transportation Systems Engineering and Information Technology*, 39(1):1–14, 2007.

[7] D. Ayala, O. Wolfson, B. Xu, B. Dasgupta, and J. Lin. Parking Slot Assignment Games. In *Proc. of the 19th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*, pages 299–308, 2011.

[8] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. SUMO - Simulation of Urban Mobility: An Overview. In *Proceedings of the Third International Conference on Advances in System Simulation (SIMUL)*, pages 63–68, 2011.

[9] J. Boehle. *City-based Parking and Routing System.* PhD thesis, Erasmus University Rotterdam, 2007.

[10] M. S. Daskin. Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms. *Transport. Science*, 26(3):185–200, 1992.

[11] S. Dernbach, B. Das, N. C. Krishnan, B. L. Thomas, and D. J. Cook. Simple and Complex Activity Recognition through Smart Phones. In *Proceedings of the 8th International Conference on Intelligent Environments (IE)*, pages 214–221, 2012.

[12] A. R. K. D. S. E.L. Lawler, J.K. Lenstra. *The Traveling Salesman Problem*, volume 18. Wiley, New York, 3 edition, 1988.

[13] A. Hakeem, N. Gehani, X. Ding, R. Curtmola, and C. Borcea. On-The-Fly Curbside Parking Assignment. In *Proceedings of the 8th EAI International Conference on Mobile Computing, Applications and Services*, MobiCASE'16, pages 1–10, 2016.

[14] J. C. Herrera, D. B. Work, R. Herring, X. J. Ban, Q. Jacobson, and A. M. Bayen. Evaluation of traffic data obtained via gps-enabled mobile phones: The mobile century field experiment. 2009.

[15] J. Hurkała. Time-Dependent Traveling Salesman Problem with Multiple Time Windows. volume 6, pages 71–78. Annals of Computer Science and Information Systems, 2015.

[16] S. Lin and B. W. Kernighan. An Effective Heuristic Algorithm for the Traveling Salesman Problem. *Oper. Res.*, 21(2):498–516, Apr. 1973.

[17] S. Ma, Y. Zheng, and O. Wolfson. T-share: A large-Scale Dynamic Taxi Ridesharing Service. In *IEEE 29th International Conference on Data Engineering (ICDE)*, pages 410–421, 2013.

[18] D. Mackowski, Y. Bai, and Y. Ouyang. Parking Space Management via Dynamic Performance-Based Pricing. *CoRR*, abs/1501.00638, 2015.

[19] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe. ParkNet: Drive-by Sensing of Road-side Parking Statistics. In *Proc. of MobiSys*, 2010.

[20] P. A. Melgarejo, P. Laborie, and C. Solnon. A Time-Dependent No-Overlap Constraint: Application to Urban Delivery Problems. In *CPAIOR*, 2015.

[21] A. Montero, I. Méndez-Díaz, and J. J. M. Bront. An Integer Programming Approach for The Time-Dependent Traveling Salesman Problem with Time Windows. *Computers OR*, 88:280–289, 2017.

[22] G. Nannicini. *Point-to-point Shortest Paths on Dynamic Time-Dependent Road Networks.* PhD thesis, Ecole Polytechnique, Palaiseau, France, 2009.

[23] S. Nawaz, C. Efstratiou, and C. Mascolo. ParkSense: A Smartphone Based Sensing System for On-street Parking. In *Proc. of MobiCom*, pages 75–86, 2013.

[24] J. Pan, I. Sandu Popa, and C. Borcea. DIVERT: A Distributed Vehicular Traffic Re-routing System for Congestion Avoidance. *IEEE TMC*, 16:1–1, 01 2016.

[25] R. Routledge. Poisson Distribution. https://www.britannica.com/topic/Poisson-distribution.

[26] R. Salpietro, L. Bedogni, M. Di Felice, and L. Bononi. Park Here! A Smart Parking System Based on Smartphones' Embedded Sensors and Short Range Communication Technologies. In *Proc. of the 2015 IEEE 2Nd World Forum on Internet of Things (WF-IoT)*, pages 18–23. IEEE Comp. Soc., 2015.

[27] B. Strasser. Dynamic Time-Dependent Routing in Road Networks Through Sampling. In *ATMOS*, 2017.

[28] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas. Generation and Analysis of a Large-Scale Urban Vehicular Mobility Dataset. *IEEE Trans. on Mobile Computing*, 13(5):1061–1075, 2014.

[29] V. S. T. Yu Huang, Bo-Hau Lin. Efficient Multi-Destinations Route Planning with Deadlines and Cost Constraints. In *18th IEEE International Conf. on Mobile Data Management (MDM)*, 2017.