2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems

Integration of Dynamic Road Condition Updates for Real-Time Ridesharing Systems

Joseph Hargrave, San Yeung, Sanjay Madria
Department of Computer Science
Missouri S&T
Rolla, Missouri, USA
jh9x8, syq3b, madrias@mst.edu

***Abstract*—Real-time ridesharing systems have been investigated as a potential method for improving congestions in overloaded road networks. In this paper, we specifically investigate integrating dynamic road conditions (DRCs) such as traffic accidents into the ridesharing systems, as the DRCs will cause unexpected delays during the route planning phase. We propose to use the CRP framework due to its capability of handling frequent road edge updates, and we will discuss the relevant issues on traffic data preprocessing, time conflict detection on ridesharing schedules, and the importance of utilizing a probability traffic model based on historical data. Our preliminary experiments show the impressive efficiency of CRP on DRC updates of varying batch sizes and distributions.**

***Keywords*-Dynamic road conditions; ridesharing systems**

I. Introduction

For a real-time ridesharing system, the goal is to assign multiple ride requests to a single vehicle so that we can serve many ridesharing requests with fewer taxi vehicles. This proposed scheme not only saves the ride fare for the passengers but also produces fewer greenhouse gases since the number of roaming taxis can be reduced and the seat utilization for a taxi is increased. Such benefits are extremely important for big cities that are suffering from traffic congestion. The economical cost of congestion is expected to rise surpassing 100 billion by the year 2030 for the whole United States [1], [2]. Hence, these concerns provide a large incentive to more efficiently transport people to their destinations using more creative approaches, like real-time ridesharing services. Additionally, each ridesharing request contains a pick-up and a drop-off time constraint for the corresponding pick-up location and destination. The processes of finding the optimal vehicle to assign to each request and planning for the optimal route is based on the time satisfaction. In literature, efficient vehicle searching and schedule planning algorithms have been proposed, but dynamic road conditions (DRCs) have not been fully considered in the context of a ridesharing system. Literally, DRCs can be unexpected in time and types; they can be caused by traffic accidents or nature-induced road blockages, both of which can have an adverse effect on the time-sensitive ridesharing schedules.

[1]The first two authors contributed equally in this work.

In an effort to solve this problem, we collected and validated real-time DRC data updates from different API's of traffic/map services, and we will explain how we can use the important traffic tags information to represent usable DRC updates. Secondly, in order to quickly process the DRC updates and reflect them in the underlying road network, we propose the employment of the Customizable Route Planning framework (CRP) developed in [3]. It is a route planning engine that can process frequent edge updates for a continental-sized road network. Using the design principles and data structures of CRP, we are then able to design efficient means of conducting time-conflict detection for the ridesharing schedules when there are DRCs present. Last but not the least, we also collected and parsed historical traffic data into different time intervals, as well as calculated the average speed for the road segments. These historical traffic data then can be used for constructing a probabilistic traffic prediction model that could be used to enhance the routing quality of ridesharing schedules by maximizing the probability of avoiding DRCs for a selected path. The integration work between such model and the CRP framework is an important task for the future. For evaluation, we used a large-scale map (8,448,453 nodes and 707,877 edges) to experiment on the effectiveness of the CRP's ability to handle a large amount of incremental DRC batch updates of varying sizes and distributions. The results show that CRP can handle the frequent updates with ease and is a suitable choice for our integration needs.

II. Related Work

The goal of making traffic to be less congested has always been an important research interest. However, the majority of early work focused on coming up with an optimal pick-up and drop-off plan given all of the ride requests beforehand [4]. It was not until the emergence of popular ridesharing services like Uber and Lyft that the research community started to push towards a new direction. Recently, many representative ridesharing systems have been proposed [5], [6]. However, the factor of handling dynamic road conditions has not yet been explored in the research field of ridesharing. In [7], the authors proposed several remedies for solving the delays experienced by onboard passengers when DRCs are present, including a delay-mitigating transferring algorithm

2155-6814/17 $31.00 © 2017 IEEE
DOI 10.1109/MASS.2017.94

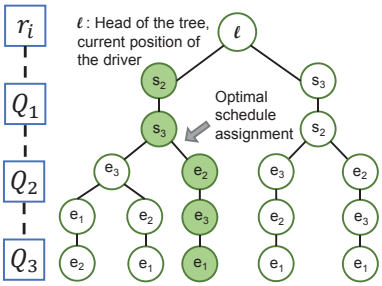

Figure 1. An example of ridesharing schedule using the dynamic tree representation where $R_i = (Q_1, Q_2, Q_3)$.

and an incentive algorithm. However, the details on how to accurately and promptly handle the data of the dynamic road conditions are not provided. Therefore, it is important for us to establish the technical foundation to how a ridesharing system needs to be capable of processing DRCs.

III. Preliminaries

Road Network: A road network $G = (V, E)$ is considered as a graph with a nonnegative cost function $d(u, v)$ associated with each edge $(u, v) \in E$, where V represents the set of vertices and E is the set of edges. The cost can be represented as travel time based on the concept of *shortest path*. The cost for a path $P = (v_0, ..., v_k)$, where $(v_i, v_{i+1}) \in E$, is denoted as $d(P) = \sum_{i=0}^{k-1} d(v_i, v_{i+1})$.

Ridesharing Requests and Schedules: In a real-time ridesharing system, one of the important tasks is to assign an optimal schedule to a vehicle upon receiving a ridesharing request. A request takes the form of $Q = (s, e, pw, dw)$ consisting of pick-up location, drop-off location, pick-up time, and drop-off time deadline, respectively. The optimal vehicle assignment for request Q_i is determined by which vehicle's current schedule would incur the least amount of additional travel distance while satisfying the time constraints of Q_i and previously assigned requests. A schedule of a driver r_i is denoted as $S_i = (v_1, ..., v_k)$, where v_i is either a pick-up or drop-off point from the request set ($R_i = (Q_1, ...Q_n)$) assigned to r_i. We employ the dynamic tree representation [6] to store a ridesharing schedule for each driver. In Figure 1, the optimal schedule for r_i is highlighted and all other branches represent different enumeration of possible schedules. This way, when a new request arrives, we can use the previously computed results stored in the tree to efficiently evaluate if the optimal schedule needs to be updated.

Dynamic Traffic Updates: A dynamic road condition (DRC) is defined as any road traffic event that is different from the normal traffic behavior. There are two types of DRC that we consider: 1) The DRC's that will cause delay to the normal traffic flow, such as traffic congestion due to rush hours and accidents. 2) The DRC's that have negligible effect, such as accidents that occur on roadsides. We will mostly focus on type 1 DRC in our research. The basic attributes for a DRC include $(p_s, p_d, type, m_d)$, representing the start and end locations, the type (or cause) of the traffic incident, and the magnitude of delay of the affected road(s).

IV. Challenges

There are several main challenges when we integrate DRCs into a real-time ridesharing system. The first one is **data acquisition and availability**. This stems mostly from sparse data. Often, real-time traffic update data is in the hand of a few sources, including large corporations that provide mapping services and traffic agencies that monitor traffic conditions from public traffic infrastructures. The complete stream of real-time traffic data is often difficult to obtain for use in other generalized applications. The second challenge arises from the need for the dynamic traffic update and query to be **accessible and scalable**. After a batch of DRCs is received, the system must quickly make the updated edge costs accessible to the incoming ridesharing requests in order to conduct optimal schedule planning. In addition, such updating components should be able to handle DRC batches of any size, preferably a road network of continental size.

V. Problem Statement

We want to solve the problem of *"integrating dynamic road condition updates into a real-time ridesharing system"* in which the unexpected delays caused by DRCs will potentially affect the time constraints of existing ridesharing schedules. We need to ensure that DRC data are are handled properly. The DRC updating component must achieve fast accessible time and be scalable to answer real-time route planning queries for the ridesharing vehicles. Additionally, we must consider how to efficiently detect the set of current assigned ridesharing schedules that are affected by the DRCs (violated time constraints for passengers) and how to re-plan for the optimal routes under the influence of DRCs.

VI. System Framework

In this section, we will introduce the components of dynamic traffic updates that we plan to integrate into a real-time ridesharing system. At its original form, a ridesharing system only requires **request handling**, **taxi scheduling**, and **route planning** (using a static road network map) components to fully operate. To enhance the system with dynamic DRCs, we propose an integration with the **CRP** (Customizable Route Planning) framework [3], which is proposed by Delling et al. as a robust routing engine that could process real-time traffic updates and personalized cost functions for a continental-sized road network. After DRCs are properly processed and useful information is extracted, they will be fed as input data into CRP such that the underlying road network can go through a series of edge

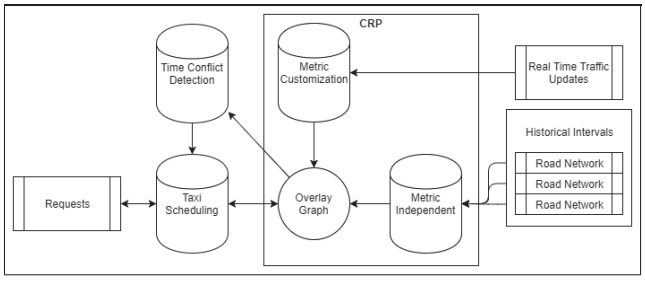

Figure 2. A framework overview of ridesharing and dynamic traffic update components.

cost updates. With the updated road network, the ridesharing system then will need to perform a **time conflict detection** among the existing ridesharing schedules to discover those that are affected by the delays caused by the DRCs. The updated map will also be used for answering routing queries coming from the new ridesharing requests.

Besides considering for the impact of DRCs on existing schedules, the inclusion of DRCs into a ridesharing system links to another major concern: *how can we utilize historical traffic data to optimize the ridesharing vehicles flow and reduce the chances of encountering DRCs among the assigned routes?* For example, given Q with $s = A$ and $e = B$, if the shortest path $P(A, B)$ tends to have a high chance of DRC occurrence within a time window t_w ($t_w = p_w + d(P(A, B))$), then we should take an alternative route with a relatively lower DRC risk provided that the time constraint of Q can still be met. This leads to the component of **historical traffic utilization**. Although this component has not been fully investigated at our current point of research, it serves as a crucial layer for achieving DRC-risk avoidance and more reliable (delay-mitigating) routing for a ridesharing system. Essentially, our initial attempt is to gather historical traffic datasets (mostly GPS logs and incident data) and perform traffic analysis. The results from the analysis will allow us to create several different historical traffic models with different traffic profiles to represent the different *states* of traffic, such as different hour-intervals throughout a day or different periods throughout a week that exhibit different traffic patterns than the rest.

VII. Methodologies and Current Accomplishments

A. Data Acquisition and Preprocessing

Real-time DRC data: Being able to obtain useful real-time DRC data is a critical task in actualizing the dynamic traffic update task. While there are many providers that give a general quick glance of road congestion or real-time traffic events, most of them (data feeds) are too ambiguous to apply in actual computation. Out of the several options that we have, we decide to utilize the traffic condition data API provided by TomTom™. An example of the API data

```
▼<tm id="1500556437009">
  ▼<poi>
     <id>northamerica_HD_US_M_TTR70661574374864</id>
    ▼<p>
       <x>-88.029044</x>
       <y>41.641567</y>
     </p>
    ▼<op>
       <x>-88.02767</x>
       <y>41.64156</y>
     </op>
     <ic>8</ic>
     <ty>4</ty>
     <cs>0</cs>
     <d>closed</d>
     <f>W 135th St</f>
     <t>Archer Ave</t>
     <l>1390</l>
   </poi>
```

Figure 3. A snippet of a DRC from the TomTom Traffic API response.

Table I
IMPORTANT TRAFFIC TAGS FROM TOMTOM TRAFFIC API RESPONSE.

id	Unique identifier.
op	Coordinates representing the position of the incident.
ic	Icon category use to identify the type of incident.
ty	A number used to represent the magnitude of delay.
d	Description of incident.
c	Cause of the incident, where available.
f	The name of the intersection where the delay of traffic starts from the incident.
t	The name of the intersection where the delay of traffic ends from the incident.
l	Length of the incident in meters.
dl	Delay the incident causes in seconds (not available in road closures).
r	The road number(s) affected by the incident. Multiple road numbers will delimited by slashes.
v	A vector representing the geometry of the incident. The vector is encoded with the Google Encoded Polyline Algorithm.

snippet shows the associated attributes for a recorded DRC is provided in Figure 3. The response contains a list of data attributes, and we have extracted the ones that we think are the most important. By using the provided information, we can sufficiently infer the type of the traffic incident (ic), its locations (op, f, and t), and the predicted delay magnitude (dl). Because not all attributes will have data for different kinds of traffic incidents, we can use the tag information to decide whether or not a DRC belongs to type 1 or type 2. It is also notable that the (f) and (t) tags simplify the process of mapping a DRC to its corresponding road segment. Although the DRCs obtained from this API provide valuable input, we are aware of the fact that the returned set of DRC might not be complete and may not include all crucial road condition information. As a result, we conducted a preliminary phase of DRC verification by comparing the data retrieved from the TomTom traffic API with other navigation service providers like Bing Maps and Google Maps traffic APIs. We found that the data from

974 lines (974 sloc) | 51.4 KB

1	00416	2007-01-06T03:19:13+00:00	37.761101	-122.389542
2	00416	2007-01-06T03:19:17+00:00	37.761473	-122.389570
3	00416	2007-01-06T03:19:21+00:00	37.761636	-122.389600
4	00416	2007-01-06T03:19:25+00:00	37.761706	-122.389793
5	00416	2007-01-06T03:19:29+00:00	37.761712	-122.390119
6	00416	2007-01-06T03:19:33+00:00	37.761693	-122.390394
7	00416	2007-01-06T03:19:37+00:00	37.761692	-122.390515
8	00416	2007-01-06T03:19:41+00:00	37.761673	-122.390696
9	00416	2007-01-06T03:19:45+00:00	37.761636	-122.391020
10	00416	2007-01-06T03:19:49+00:00	37.761582	-122.391320
11	00416	2007-01-06T03:19:53+00:00	37.761573	-122.391450
12	00416	2007-01-06T03:19:57+00:00	37.761580	-122.391520
13	00416	2007-01-06T03:20:01+00:00	37.761428	-122.391578
14	00416	2007-01-06T03:20:05+00:00	37.761177	-122.391581
15	00416	2007-01-06T03:20:09+00:00	37.761007	-122.391568

Figure 4. GPS logs snippet for the San Francisco region

TomTom traffic API is comparable with other APIs and the DRC dataset is thus inclusive overall.

Historical Traffic Data: The most common form of historical traffic data comes from the GPS trajectories collected from either vehicles or GPS-enabled electronic devices. An example of GPS logs that records the time, location, and speed information is shown in Figure 4. Challenges for processing GPS logs include handling the errors and increasing the accuracy of the data. We also need to clean the data for road segments that are associated with abnormalities (e.g., incorrect speed information). Additional concerns may also include accounting for turns. When matching GPS logs to road segments, we calculate the average speed from the GPS logs and set that as the segment speed. Outliers are removed to avoid skewing the results.

B. Dynamic Traffic Update Components

Customizable Route Planning Engine: Different from other route planning frameworks less effective in handling dynamic traffic updates, CRP adapts separator-based algorithms and is robust enough to incorporate multiple customized cost metrics very quickly. In its design structure, the road network G is stored as nested multilevel graph partitions. That is, a multilevel partition of V (the edge set) is a family of partitions $\{C^0, ..., C^L\}$ where l denotes the level of a partition C^l and $C^l = \{C_0, ...C_k\}$ (cell sets) such that $C_i \subseteq V$ with each $v \in V$ contained in exactly one cell. In addition, a boundary arc on level l is an arc with endpoints in different level-l cells; a boundary vertex on level l is a vertex with at least one neighbor in another level-l cell. To achieve fast route planning, it employs a partition-based approach that uses multiple levels of overlay graphs. An overlay graph H_i for each level i of the partition includes all boundary arcs, plus an overlay linking the boundary vertices within a cell. The shortest path query between two points in G can be run with a bi-directional version of Dijkstra's algorithm on the graphs consisting of the union of H and the partition cells containing the two points.

CRP performs two phases of graph preprocessing: the metric independent and metric customization phases. The first phase only considers the structure of the road network and is independent of the custom cost metrics desired. The metric customization phase runs much faster than the first phase and is designed to be rerun frequently to accommodate many network edge changes. When receiving DRC updates, we do not need to rerun the entire second phase. Instead, the framework will detect which cells in the partition are affected and only run the customization updates on those cells. The capability of CRP to minimize latency between DRC and road network updates suits the need for our intended integration. So far, we have fully implemented the code needed to integrate CRP into the ridesharing system developed in [7].

Time Conflict Detection: There are two tasks that we aim to achieve during such detection: (i) selecting the schedules that are potentially affected by the DRC delays, and (ii) validating the time constraint satisfaction of a schedule. In the simplest way, we can check the time constraint violation for every existing ridesharing schedule, but we can reduce the search space by taking advantage of the multilevel graph partitions. Once an optimal schedule is assigned to a driver using the dynamic tree, we will associate with the schedule a list of cells that the route belongs to, denoted as C^{S_i}. Since the customization phase invoked by DRC updates recomputes the shortcut costs for all boundary vertices one cell at a time for each overlay level, we can execute customization and schedule selection in parallel by checking if $C_i^l \in \bigcup_{i=1}^{k} C^{S_i}$: if cell i of level l appears in the union of the cell list from the existing k schedules in the ridesharing system, then only those schedules that contain C_i^l needs to be validated. For the retrieved set of schedules, we can check for time constraint violation by running the bi-directional Dijkstra's algorithm for the paths leading to each pick-up or drop-off point in S_i.

Historical Traffic Data Model: So far, we have implemented code to validate and parse historical GPS logs that we collected. We have also successfully partitioned GPS logs into an arbitrary set of time intervals (dividing 24 hours in a day into every hour interval) and computed the average speed for the road segments. This information will allow us to construct a probability traffic flow model by considering travel time, risk, and time deadline simultaneously [8]. As a future work, we plan to integrate this said probabilistic model that can maximize the probability of arriving on time into the ridesharing system, but we must consider the challenges that will arise from integrating it with DRC because the real-time traffic updates might be a new traffic pattern that has not been seen/accounted for by the

probabilistic model.

VIII. Experiment

As a preliminary experimental result, we first evaluate the performance of the CRP framework with DRC updates. The experiments were run on OS Ubuntu 64-bit, Intel Core i5-4200U CPU @ 1.60GHz x 4, and 8GB of RAM. We obtained a road network of Missouri, USA, from Open Street Map, which contains 8,448,453 nodes and 707,877 edges. The total time for the preprocessing phases of CRP took 12m, 9.866s. Figure 5 shows the time CRP took to run

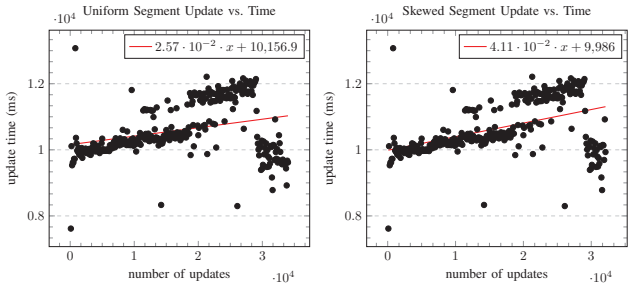

Figure 5. Uniform vs Skewed DRC segment distribution updates

the customization phase with varying DRC batch size and edge distribution. The red line represents the fitted trend of all the data recorded. The data points were generated by increasing the number of DRC updates by 100 every step. For the uniform distribution, DRC occurs on road segments evenly throughout the road network. The data from the second graph was computed by updating DRC segments that were sparse elsewhere, but were heavily distributed around a compact space from major cities across Missouri (Saint Louis, Kansas City, Springfield). The results show that increasing DRC batch size requires more computation time for customization, and different DRC segment distributions yield almost identical results. Even though we tested over 30,000 road segment updates in both graphs, CRP is apparent in its capability of handling frequent edge updates with no significant decrease in efficiency under different DRC sizes and distributions. We will experiment on the time conflict detection component with CRP in the near future.

IX. Future Work

1) Because there is no guarantee that the DRC updates from a single API can capture all crucial road conditions, one thing that we can try next is to collect DRC updates from heterogeneous sources. We could include social media platforms like Twitter [9], traffic authority websites, and traffic reported by crowdsourcing applications like Waze. 2) We must keep a close eye on the research advancements in smart traffic infrastructures and autonomous vehicles, as they imply a more extensive deployment of sensors to collect a much more diversified set of traffic data [10]. These new changes could immensely impact the ways we collect and process DRC updates. 3) There is much work to be done to integrate the CRP framework with a probability traffic model for enhanced routing quality. For example, when the DRC updates do not agree with the traffic prediction, the ridesharing system needs to have an online component that can quickly infer any significant impact caused by some DRCs that the probability model has never seen before.

X. Conclusion

In this paper, we discussed the challenges that arise from integrating DRCs into a real-time ridesharing system, which includes data processing on DRC and historical data, why we selected the CRP framework to be the DRC update engine, and the importance of a traffic predictive model to be used alongside CRP in order to enhance routing quality. We also proposed methods for handling the issue of time conflict detection among ridesharing schedules using advantages offered by CRP. Our experimental results based on a real road map prove the robustness and effectiveness of CRP's capability for handling DRC updates of varying sizes and distributions.

References

[1] O'Toole, Randal. "Solving the Problem of Traffic Congestion." National Center for Policy Analysis. National Center for Policy Analysis, 8 Oct. 2012. Web. 12 Apr. 2017.

[2] Levy, Jonathan I., et al. "Evaluation of the public health impacts of traffic congestion: a health risk assessment." Environmental health 9.1 (2010): 65.

[3] Delling, Daniel, et al. "Customizable route planning in road networks." Transportation Science (2015).

[4] Xiang, Zhihai, Chengbin Chu, and Haoxun Chen. "A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints." European journal of operational research 174.2 (2006): 1117-1139.

[5] Ma, Shuo, Yu Zheng, and Ouri Wolfson. "Real-time city-scale taxi ridesharing." IEEE Transactions on Knowledge and Data Engineering 27.7 (2015): 1782-1795.

[6] Huang, Yan, et al. "Large scale real-time ridesharing with service guarantee on road networks." Proceedings of the VLDB Endowment 7.14 (2014): 2017-2028.

[7] Yeung, San, Evan Miller, and Sanjay Madria. "A Flexible Real-Time Ridesharing System Considering Current Road Conditions." Mobile Data Management (MDM), 2016 17th IEEE International Conference on. Vol. 1. IEEE, 2016.

[8] Cao, Zhiguang, et al. "Maximizing the Probability of Arriving on Time: A Practical Q-Learning Method." AAAI. 2017.

[9] Chen, Po-Ta, et al. "Road traffic congestion monitoring in social media with hinge-loss Markov random fields." IEEE International Conference on Data Mining, 2014.

[10] Liggins II, Martin, David Hall, and James Llinas, eds. Handbook of multisensor data fusion: theory and practice. CRC press, 2017.