Route Recommendation to Facilitate Carpooling

Christine Bassem
Department of Computer Science
Wellesley College
Wellesley, MA, USA
cbassem@wellesley.edu

Svitlana Honcharuk
Class of 2024
Wellesley College
Wellesley, MA, USA
sh102@wellesley.edu

Mohamed Mokbel
Department of Computer Science and Engineering
University of Minnesota - Twin Cities
Minneapolis, MN, USA
mokbel@umn.edu

2022 23rd IEEE International Conference on Mobile Data Management (MDM) | 978-1-6654-5176-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/MDM55031.2022.00025

***Abstract*—Recently ride-sharing platforms have struggled with a decreased supply of drivers, which has negatively impacted their passengers, by subjecting them to long delays and extremely high surge prices. An approach for mitigating these problems is for service providers to facilitate and coordinate carpooling via the recommendation of individually curated paths, not necessarily the shortest, for drivers towards completing their chosen rides. In this paper, we redesign the Weight Evolving Temporal graph structure to efficiently encode large dynamic road networks with temporal ride availability. Leveraging that graph structure, we efficiently define a polynomial-time optimal route recommendation algorithm that increases carpooling opportunities, taking into consideration the spatio-temporal constraints of both drivers and rides in such a highly-dynamic setting. Finally, we use simulations to demonstrate the effectiveness of these route recommendations, on both the driver and passenger experience.**

***Index Terms*—ride-sharing, carpool, routing, ride assignment, temporal, evolving graph.**

I. Introduction

As evident with the recent pandemic, the decrease in the number of drivers in ommercial ride-sharing platforms, such as Uber [7] and Lyft [10], led to price surges and increased delays [1], [15]. This has led to increased negative emotions towards these platforms [11], forcing companies to increasing driver incentives [12] and bringing back carpool options [13], in attempts to reduce prices. With the return of carpooling, customers can benefit from reduced fares, but drivers may still experience lower income and reduced gain due to wasted time deadheading, *i.e.*, circling around idly looking for the next passenger [3]. In this work, we argue that there is a need for a better model for coordinating ride-sharing in such platforms.

Although service providers cannot control drivers in crowd-based platforms, they still have full access to current and historical data on rider and driver availability. This data can be used to recommend "better" ride choices for drivers, accompanied by individually curated paths, not necessarily the shortest, that would facilitate carpooling. In such a model, drivers have the incentive to follow these recommendations, as they increase their chances of picking up multiple rides along their paths. Meanwhile, passengers have the incentive to carpool, as they pay reduced fares for shared rides with a delay that is constrained to some threshold of their choice.

This work is supported by the National Science Foundation, USA, under Grants IIS-1907855 and CSR-1755788.

In the context of ride-sharing, the route recommendation problem was defined in [21], in which drivers are recommended routes that have highest probability of finding more passengers, while remaining within their detour thresholds. However, due to the highly dynamic nature of ride availability in large road networks, efficient optimal route recommendation algorithms are yet to be defined. In this paper, we address that problem via the design of a temporal graph structure that efficiently encodes both the dynamic road network properties and the spatio-temporal ride availability over that network. Thus, facilitating the definition of optimal route recommendation algorithms.

Contribution. The first contribution of this paper lies in optimizing the design of the Weight Evolving Temporal (WET) graph structure [2], in which both vertex and edge weights evolve over time. In this work, we annotate the graphs with ride frequency information obtained from past ride traces, to represent the expected spatio-temporal ride availability in the network. Index-based labeling is adopted to efficiently retrieve and store these annotations, which allow for the definition of polynomial-time optimal routing algorithms that do not necessarily optimize for the shortest routes, but for the maximum benefit to be expected from picking up future rides along the route.

The second contribution lies in the definition of the optimal Dual-Path (DP) routing algorithm, which is a 3-endpoint routing algorithm that is well-suited for finding the optimal combination of dual-paths. With the first path from a driver's current location to a ride's pick up location, and the second path leading to the ride's drop off location. The algorithm is theoretically proven to find optimal routes in polynomial time, and it can be easily integrated into existing ride-sharing platforms.

Finally, we briefly propose a model of ride assignment, or rather recommendation, based on the results of these routing algorithms, and evaluate the effectiveness of these recommendations in simulated ride-sharing experiments. We compare our model to typical driver choice models in ride-sharing platforms; closest-first, max-revenue-first and least-detour, and show that it is more beneficial for drivers (and passengers) to heed the recommended paths.

Paper Outline. In Section II, we present our system model, followed by the definitions of our data structures and routing algorithms in Section III. In Section IV, we present our

2375-0324/22/$31.00 ©2022 IEEE
DOI 10.1109/MDM55031.2022.00025

evaluation framework and discuss experimental results. Then, we provide a brief review of related works in Section V and conclude the paper with a discussion of future work.

II. Route Recommendation Model in Ride Sharing

In this section, we start by defining the system model and main definitions.

A. System Model

In typical crowd-based ride-sharing platforms, the service provider acts as the broker between the passengers, represented by their rides, and the drivers. Drivers are assumed to be individual and rational participants, with personal schedules, constraints, and utilities. They have the flexibility to choose the rides to complete based on their personal preferences, or to accept ride recommendations from the platform. Moreover, drivers have the opportunity to specify their schedule constraints with the platform by indicating their destination location, associated with their preferred latest arrival time at that destination, implicitly defining their detour threshold. This personal schedule information can be leveraged by the platform to provide better individually-curated route and ride recommendations for the drivers.

Rides are associated with single passengers, and each ride is defined by its spatio-temporal endpoints, representing the ride's pick up and drop off location, as well as the threshold of delay based on the passenger's preferences. In this work, we assume that the spatial endpoints of a ride may not be negotiated, as opposed to its temporal constraints. Rides with no flexibility, *i.e.*, a zero-threshold of delay, expect to be picked up right away and routed through the shortest path to their destinations. On the other hand, rides with more flexibility, *i.e.*, with some maximum threshold on delay, tolerate longer routes, possibly allowing drivers to complete other rides along the way.

B. System Definitions

We start the definitions with that of the road network, on which both the drivers and rides are active.

Definition 1: The ***Road Network*** is modeled as the graph, $G = (V, E, c)$, in which the set of vertices V represents the various landmarks in the road network, and the set of edges E represents the roads between these landmarks. The cost of traversing the edge between pairs of locations is captured by the cost function $c : V \times V \rightarrow N$[1].

Rides arrive to the system in an online manner, with a definition of endpoints and an optional threshold of delay.

Definition 2: A ***Ride Request***, $r_i = < sloc_{r_i}, tloc_{r_i}, t_{r_i}, flex_{r_i} >$, represents the spatio-temporal endpoints of the ride request. The spatial constraints are represented by the pick up and dropoff locations, $sloc_{r_i} \in V$ and $tloc_{r_i} \in V$ respectively. The temporal constraints are represented by the earliest pick up time, $t_{r_i} < T$, and the delay threshold, $flex_{r_i}$.

[1]For the purposes of this work, the cost of an edge or path is measured as the time it takes to traverse that path, which is equivalent to its shortest distance. Time is modeled as a discrete sequence of time steps, with some maximum value of T, namely the recommendation period.

Similarly, drivers join the platform in an online manner. If a driver opts to share their spatial and temporal constraints, we define it as the driver span. If not, then only the current location of the driver is used in the routing process.

Definition 3: A ***Driver's Span***, $d_i = < sloc_{d_i}, t_{d_i}, tloc_{d_i}, flex_{d_i} >$, represents the spatio-temporal constraints of the driver. These constraints are in the form of the start and target locations, $sloc_{d_i}, tloc_{d_i} \in V$, associated with the earliest departure time, $t_{d_i} < T$, and the detour threshold, $flex_{d_i}$.

Definition 4: The ***Route Recommendation*** is the set of spatio-temporal stops throughout their span of availability, $PR_{d_i} = \{< sloc_{d_i}, t_{d_i} >, < loc_1, t_1 >, < loc_2, t_2 >, ...\}$. The path starts with the current location of the driver, and then each spatio-temporal stop could represent a pick up event, a drop off event, or a recommended stop to visit for a possible carpool opportunity.

Finally, we define the route recommendation problem.

Definition 5: ***Route Recommendation problem.*** Given a driver and a ride, the route recommendation problem is defined as that of finding the optimal driver recommendation from the driver's current (start) location to the ride's drop off location, passing by its pickup location. An optimal path is one which maximizes the chances of the driver picking up other rides (current or future) along the way.

III. Routing to Facilitate Carpooling

In this section, we start with the definition of Weight Evolving Temporal graphs to effectively represent the dynamic nature of the road network, and our proposed 3-endpoint optimal polynomial-time routing algorithm with a discussion on how to incorporate them into ride-sharing platforms to facilitate carpooling.

A. WET Graph for Large Networks

We redefine the WET graph structure first defined in [2] to include two main properties of realistic road networks; the very large size of these networks and the temporally-varying expected reward of visiting the nodes on such networks.

A Weight Evolving Temporal (WET) graph, over a recommendation period T, is defined as $G^T = (V, E, \delta, \omega)$, in which the set of temporal vertices V represents the various landmarks in the mobility field and the set of edges E represents the links, *i.e.*, streets, between these landmarks. The cost function, $\delta : V \times V \rightarrow N$, represents the duration of the shortest paths between the two locations $u, v \in V$. The temporal weight function, $\omega : V \times N \rightarrow \Re$, represents expected reward of visiting a vertex $v \in V$ at some time unit $0 \leq t \leq T$.

The novelty of the WET graph model lies in the temporal weight function, which links the weight of a vertex to the expected reward of visiting that specific location over time. This allows for various node scoring models that are typically encountered in dynamic networks, such as the reward for completing a task at a specific location as in crowd sensing platforms, and the probability of finding future rides to pick up or the similarity of existing rides to an already committed ride in ride-sharing platforms.

Our first optimization to the WET graph structure lies in tackling the extremely large pool of temporal weights on that graph, which emerges from the large size of V and the varying ride expectations from these locations over the duration of a day. We adopt an index-based approach to efficiently model the temporally-evolving node weights by re-designing the node weight function, ω, as a HashMap with $< key : (v, t), value : \mathbb{R} >$, in which the value represents the past frequency of ride pickups at that location, $v \in V$, at time step $t \leq T$.

The second optimization lies in the definition of the edge cost function, $\delta : V \times V \rightarrow N$. For the purposes of this paper, the cost function is implemented using a HashMap with $< key : (u, v), value : \mathbb{R} >$, in which the value represents the minimum duration to traverse the path between the two locations in the key, $u, v \in V$[2].

B. Routing with Three Endpoints

Given a WET graph and a trio of spatio-temporal endpoints, representing the driver's current location and a ride's endpoints, the objective of routing is to find the optimal path between these endpoints that maximizes the overall chance of picking up more rides along the way. In other words, and without loss of generality, the objective is to maximize the total expected reward collected by the driver from visiting various WET graph vertices, while not violating the spatio-temporal constraints of the ride.

To optimally find a path, we decompose the routing problem into two parts. The first part is to find the set of optimal routes between the first two endpoints, with the assumption that the rider can withstand a delay in pick up. The second problem complements the first one, by finding the set of optimal paths between the second two endpoints, with the assumption that the rider can be dropped off at their maximum drop off time.

Algorithm 1 Dual-Path routing algorithm

Input: $<$ *dloc*, *dt*, *sloc*, *t*, *tloc*, *flex* $>$
Output: Optimal set of vertices along the trip
1: Retrieve $G^T = (V, E, \delta, \omega)$: Annotated WET graph
2: $M1, Prev1$ = Algorithm2($<$ *dloc*, *dt*, *sloc*, $t + flex >$)
3: $M2, Prev2$ = Algorithm2($<$ *tloc*, $t + flex$, *sloc*, $t >$)
4: Choose $i*$ with $max\{M2[sloc][i*] + M1[sloc][t + flex - i* - dt]$
5: Backtrack from $Prev1[sloc][t + flex - i* - dt]$ to get $P1$
6: Backtrack from $Prev2[sloc][i*]$ to get $P2$
7: **return** $R = reverse(P1) \cup P2$

We define the Dual-Path routing algorithm, as shown in Algorithm 1, which is a combinatorial dynamic program that builds up two tables of optimal paths between the three endpoints, using Algorithm 2. The objective of Algorithm 2 is to compute the set of optimal paths between two spatio-temporal endpoints, and it is a direct implementation of the dynamic recurrence shown in (1), with the setup of the data structures that save the results computed by that recurrence.

[2]Future work is on also representing the temporal variation of edge costs.

$$OPT(v,t) = \omega(v,t) + max\Big\{OPT(v,t-1),\ max_{(u,v)\in E}\{OPT(u,t-\delta(u,v))\}\Big\} \quad (1)$$

,where $OPT(v, t)$ represents the optimal reward that can be collected by visiting vertex $v \in V$ on a WET graph G^T at time $0 \leq t \leq T$.

Algorithm 2 Dynamic program for reward-maximizing routing with two endpoints

Input: $<$ *sloc*, *t*, *tloc*, *flex* $>$
Output: Routing table for routes from *sloc* to *tloc*
1: Retrieve $G^T = (V, E, \delta, \omega)$: Annotated WET graph
2: Set $maxT = t + minDist(sloc, tloc) + flex$
3: Create array M of size $|V| \times (maxT + 1)$
4: Create array $Prev$ of size $|V| \times (maxT + 1)$
5: Set $M[v][t] = -1, \forall (v, t)$
6: Set $M[sloc][0] = 0$
7: **for** $t = 1$ to $maxT$ **do**
8: **for** each vertex $v \in V$ **do**
9: **for** each vertex $u \in V$ s.t. $(u, v) \in E$ **do**
10: Choose $u*$ with maximum $M[u][t - \delta(u, v)]$
11: **end for**
12: $M[v][t] = M[u*][t - \delta(u*, v)] + \omega(v, t + t_1^p)$
13: $Prev[v][t] = u*$
14: **end for**
15: **end for**
16: **return** $M, Prev$

We note that for the computation of the second rewards matrix $M2$, the path is reversed temporally, to allow for a start from a fixed point of time. Thus, ensuring the optimality of the result. To accommodate this, Algorithm 2 is tweaked to allow for backward advancement in time if the endpoints of the trip is reversed. After the optimal reward tables are obtained, the optimal pick up point in time is chosen, $i*$. The optimal pick up time is the value that maximizes the sum of rewards obtained from the two segments meeting at the pick up location, and it can be obtained by scanning the two tables $M1$ and $M2$.

Finally, after the optimal pick up time is chosen, the reverse path from the driver's location to the pick up location is built, $P1$, and the path from the pick up location to the drop off location is built, $P2$. Both paths are built by backtracking through the $Prev$ tables. Finally, the combined path, which joins them at time $i*$, is the final result returned by the algorithm leading to a total running time complexity of $\Theta(maxT \times |V|^2)$.

C. Route Recommendation within the Platform

The theoretical definitions of the routing algorithms above can be used in different contexts. In this section, we discuss the main aspects of incorporating them into a ride-sharing platform.

Endpoints to use for the routing algorithms. Route recommendations in the platform are requested in one of two scenarios; (1) a driver already chose a ride and is requesting a non-shortest path route to complete it, or (2) a driver is considering a ride and needs to inquire about the carpooling opportunities associated with that ride. Regardless of the reason for route inquiry, when a route recommendation is requested within the platform, the DP algorithm is executed. The input to the DP algorithm is the driver's current location and current time, as well as the properties of the ride request including its delay threshold. The output of the routing can be used in various ways depending on the requirements and design of the ride-sharing platform.

Pruning the graph to further improve efficiency. Although our defined routing algorithms have a polynomial running time complexity, it is a function of the number of edges in the WET graph, which is extremely large. This could lead to further delays when matching rides to drivers as the platform is running in real-time.

To further optimize the running time of the routing algorithms, the vertices of the WET graph are pruned based on the input to the routing algorithm. To be more specific, as the initial dynamic programming structure is being built in Algorithm 2, all vertices that are not reachable within the trip can be removed from the route computation process. Thus, reducing the number of vertices to some $k << |V|$, and drastically improving the running time of the routing algorithm to a complexity of $\Theta(maxT \times |k|^2)$. Reachability can be determined in constant running time, since all routes have temporal constraints that can be used to explicitly identify unreachable locations within these routes.

Leveraging the routes for ride recommendation. Although ride assignment is not the focus of this paper, there is an opportunity to define a simple approach for ride recommendation that leverages the routing algorithm defined above. If a driver is undecided on which ride to choose, the platform can recommend a ride based on its carpooling potential. This potential can be computed by executing the DP algorithm multiple times on behalf of the driver; once for each feasible ride. Finally, the ride that is associated with the route of highest reward is recommended to the driver, with its recommended route.

IV. Performance Evaluation

Although the routing algorithm defined in this paper is optimal, its effect on the quality of ride-sharing service requires further evaluation. In this section, we use simulations to evaluate the effect of drivers following their route recommendations on the quality of service of the platform.

A. Experimental Setup

We developed a discrete event Java simulator with the purpose of evaluating the effectiveness of various ride choice and routing algorithms on the quality of service in ride-sharing platforms. All simulations were executed on a machine running macOS version 12.1, with 8-Core Intel Core i9 machine and 32 GB of RAM.

1) Dataset: All ride information and driver spans are generated from real traces obtained from the San Francisco taxi dataset [14]. This dataset contains GPS coordinates of approximately 500 taxis collected over 30 days in the San Francisco Bay Area. The location coordinates are time stamped and are associated with the vacancy status of the taxi allows for efficient retrieval of realistic ride information, as well as driver information while deadheading.

2) Evaluation Metrics: For the purposes of this work, we measure the ride-completion rate, which is the ratio between completed rides and all rides arriving to the platform, and the carpool factor, which is the average number of rides picked up by a driver along a single trip. Moreover, we measure the average delay in pick up per ride, as well as the average detour encountered per driver for following the route recommendations [3].

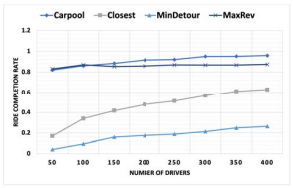
3) Ride-Choice Algorithms: In the experiments below, we evaluate the effect of drivers following ride recommendations while completing rides, as opposed to strictly using shortest paths. Moreover, for ride-choice, we implement various algorithms by which drivers can choose their rides, such as (1) **MaxCarpool**, in which the route recommendation algorithm is leveraged by the platform to recommend rides with higher opportunities of carpooling. (2) **ClosestPickup**, in which the drivers chooses the ride with the nearest pickup location to their current one, and breaking ties based on the higher revenue. (3) **LeastDetour**, in which the driver chooses the ride that incurs no extra cost or delays to their original personal schedules, and breaking ties based on the higher revenue. (4) **MaxRevenue**, in which the driver chooses a reachable ride with the maximum revenue associated with it, and minimizing detour when breaking ties.

B. Efficacy Experiments

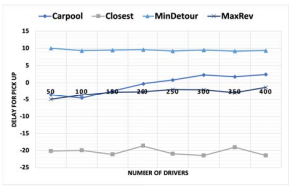
Since a theoretical analysis isn't possible, we reverse engineer optimal solutions, and then evaluate the performance of the system in various settings. To be more specific, we perform these experiments in a more controlled environment, in which driver spans are randomly generated on a 10×10 Manhattan Grid. Then, for each driver span, two rides are generated along that span, *i.e.*, the spatio-temporal properties of both rides fit within the spatio-temporal constraints of the driver, with extra time to spare. In an optimal scenario, 100% of the rides should be completed within a reasonable delay threshold and driver detours. Moreover, the experiments are designed such that further rides arrive earlier.

In the first experiment, we increase the number of drivers roaming around the mobility field, and assign them all the same detour threshold of 60, which means that each driver can add up to 60 units of time to their typical shortest path along their span. In Fig. 1, we demonstrate the effect

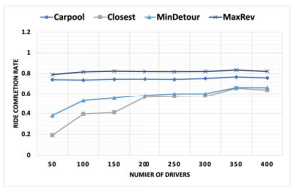
[3]We note that only a subset of the results are shown due to space limitations.



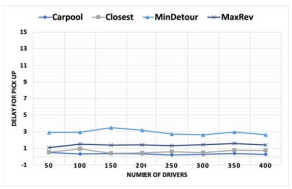
(a) Route recommendations lead to almost optimal ride completion.



(b) Route recommendations lead to minimal pickup delays.

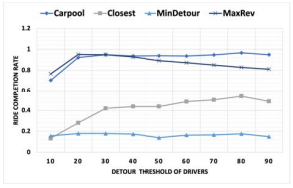


(c) Shortest paths don't reach optimal completion rates.

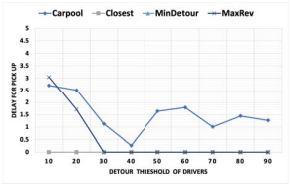


(d) Shortest paths lead to an increase in pickup delays.

Fig. 1. Positive impact of drivers following the route recommendations, as opposed to shortest paths, even when they make their own ride choices.



(a) Following recommended rides and routes allow for almost optimal ride completion rate.



(b) Almost no delays with increased detour.

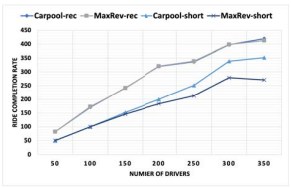
Fig. 2. An increased driver detour threshold leverages every bit of flexibility available, allowing for better ride completion.

of riders following the platform's ride recommendations, as opposed to taking the shortest path with the various ride choice algorithms.

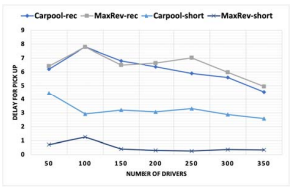
In general, closest-first provides the minimum delay in pickup and min-detour offers the minimum detours, but both ride choices perform the worst in terms of ride completion and revenue collected. Even with the optimal route recommendations, as demonstrated in Fig. 1a and Fig. 1c, these greedy models for choosing rides lead to missed opportunities for drivers.

Following the route recommendations allow for almost 98% ride completion with minimal pick up delays, even when drivers are greedily choosing the rides with maximum revenue. Moreover, as Fig. 1c demonstrates, better ride choice algorithms complete about 20% less rides when not coupled with optimal routes.

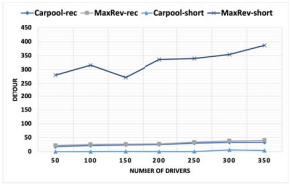
In the second experiment, we evaluate the impact of increased availability of drivers on the platform, even with drivers following recommended ride choices and routes. For all these experiments, 200 driver spans and 400 optimal rides were reverse engineered over the 10×10 grid, similar to the previous experiment. We vary the detour threshold of drivers



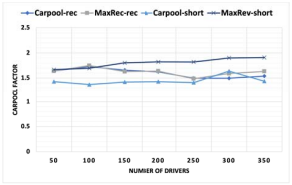
(a) Completion rate increases.



(b) Delays are minimally affected.



(c) Detour flexibility is leveraged only when needed.



(d) Carpool factor is almost at the maximum of 2 passengers per driver.

Fig. 3. Route recommendations are effective in a dynamic online setting, with a ride completion rate increase of about 45%.

from 10 to 90 units, and repeat the simulations for each value. The results in Fig. 2 indicate that our defined route recommendation algorithm leverages almost every bit of detour flexibility, allowing drivers to gain the most out of the rides completed, while achieving up to 97% completion rate when coupled with ride recommendation. Moreover, the coupling of optimal rides and routes allow for a high completion rate, with less detours and a slight increase in pickup delays.

C. Online Experiments

In these experiments, we use the traces of the first 28 days in the SF dataset to annotate a WET graph of $30,000$ vertices. Then, the traces from the last day is used to generate rides and driver spans in the simulated platform. Although in that dataset, drivers did not have spans, we decided to specify their spatio-temporal endpoints to effectively evaluate the route recommendation in a highly-constrained scenario. To add flexibility to the simulation, threshold values of 60 and 10 were added to the driver detour and ride delay respectively.

We run multiple simulations with a varying number of drivers for a duration of 360 time units (representing 3 hours), and focus on the effect of route recommendation on the Carpool and MaxRevenue ride choices. As shown in Fig. 3, the number of rides completed is on average 45% more when route recommendations are followed, with a minimal effect on the pickup delay and driver detour. This is an indication of the efficient routing, which leverages the detour thresholds of drivers only when needed. Moreover, we measure the carpool factor in the system, as shown in Fig. 3d, and confirm that the carpooling opportunities is as high as 1.8 when recommendations are followed.

V. Related Work

Existing work on ridesharing can be generally classified into three main categories based on the model of driver availability

in the system; taxi-based services [9], [19], dial-a-ride services [6], and ride-sharing with private cars [8]. Ride-sharing in platforms with private cars, usually referred to as dynamic ridesharing, assumes that both riders and drivers have their personal constraints and they join the platform at their own accord. For most of these platforms, the work on ride-sharing generally focuses on either ride assignments, route planning, or a hybrid of both. Our work falls in the second category.

Ride Assignment. Most works that explore ride assignment consider one-to-one matching, where a rider is matched to at most one driver, and a driver is matched to at most one rider at a time, as in [16]. Some works deviate from this model, as in SHAREK [4], in which a rider is matched to a set of drivers that they can choose from. Although our algorithms can be used for ride assignment, the focus of this paper is on the route planning aspect.

Routing Planning. The ridesharing works that focus on ride assignment generally use the shortest path for routing their matches [17]. Other routing algorithms are mostly studied in taxi services with the goal of sending drivers to areas with the greatest number of riders based on historical data to minimize their idle time [19].

Another group of routing algorithms are based on inserting the new ride (pick up and drop off locations) into an existing route of a vehicle while optimizing for a certain objective [5], [18]. However, this approach might not be as efficient long-term. So based on that observation, Wang et. al designed a dynamic programming algorithm that considers where rides may appear in the nearest future such as next 90 minutes based on historical data [20] to improve the routing results.

Recently, the problem of routing planning with detours have been proposed in the context of ride-sharing. For example, Yuen et. al proposed a routing recommendation algorithm that maximizes the chance to find compatible rides with a minimum detour using historical data [21]. Our work directly contributes to this category.

VI. Conclusion

In this paper, we investigate the benefit of coordinating carpooling within ride-sharing platforms with private drivers, by recommending for drivers routes that maximize their chances of picking up more rides along the way. We define an optimized graph structure, namely Weight Evolving Temporal (WET) graphs, which when annotated with ride availability information, it can be used to define efficient and optimal routing algorithms. Our simulations indicate the effectiveness of route recommendations on the quality of service of the ride-sharing platform, even when drivers make their own choice of rides. For future work, we aim to develop more efficient routing algorithms for WET graphs, and to define more comprehensive ride assignment and incentive mechanims to benefit both the drivers and passengers in the platform.

References

[1] Bobby Allyn. Lyft And Uber Prices Are High. Wait Times Are Long And Drivers Are Scarce. https://www.npr.org/2021/08/07/1025534409/lyft-and-uber-prices-are-high-wait-times-are-long-and-drivers-are-scarce, 2021. [Online; accessed January-28-2022].

[2] Christine Bassem. Redefining node centrality for task allocation in mobile crowdsensing platforms. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 323–331, 2019.

[3] Greg Bensinger. For Uber and Lyft, the Rideshare Bubble Bursts. https://www.nytimes.com/2021/10/17/opinion/uber-lyft.html, 2021. [Online; accessed January-21-2022].

[4] Bin Cao, Chenyu Hou, Liwei Zhao, Louai Alarabi, Jing Fan, Mohamed F. Mokbel, and Anas Basalamah. SHAREK*: A Scalable Matching Method for Dynamic Ride Sharing. *GeoInformatica*, 24(4):881–913, oct 2020.

[5] Peng Cheng, Hao Xin, and Lei Chen. Utility-aware ridesharing on road networks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, volume Part F1277, pages 1197–1210. Association for Computing Machinery, may 2017.

[6] Jean François Cordeau and Gilbert Laporte. The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, 153(1):29–46, may 2007.

[7] Uber Technologies Inc. Uber. http://www.uber.com. [Online; accessed January-28-2022].

[8] Qiulin Lin, Lei Deng, Jingzhou Sun, and Minghua Chen. Optimal demand-aware ride-sharing routing. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 2699–2707. IEEE, 2018.

[9] Zhidan Liu, Zengyang Gong, Jiangzhou Li, and Kaishun Wu. mt-share: A mobility-aware dynamic taxi ridesharing system. *IEEE Internet of Things Journal*, 9(1):182–198, 2021.

[10] Inc Lyft. Lyft: A ride whenever you need one. http://www.lyft.com. [Online; accessed January-28-2022].

[11] Syed Ahnaf Morshed, Sifat Shahriar Khan, Raihanul Bari Tanvir, and Shafkath Nur. Impact of covid-19 pandemic on ride-hailing services based on large-scale twitter data analysis. *Journal of Urban Management*, 10(2):155–165, 2021.

[12] Kari Paul. Heavy spending on driver incentives pushes Uber to bigger-than-forecast loss. https://www.theguardian.com/technology/2021/aug/04/uber-revenues-delivery-service-pandemic, 2021. [Online; accessed January-28-2022].

[13] Jay Peters. Uber reintroduces shared rides with a new name. https://www.theverge.com/2021/11/16/22786147/uber-uberx-share-rides-carpooling-new-name, 2021. [Online; accessed January-21-2022].

[14] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. CRAWDAD dataset epfl/mobility (v. 2009-02-24). Downloaded from https://crawdad.org/epfl/mobility/20090224, February 2009.

[15] Faiz Siddiqui. Where have all the Uber drivers gone? https://www.washingtonpost.com/technology/2021/05/07/uber-lyft-drivers/, 2021. [Online; accessed January-28-2022].

[16] Na Ta, Guoliang Li, Tianyu Zhao, Jianhua Feng, Hanchao Ma, and Zhiguo Gong. An Efficient Ride-Sharing Framework for Maximizing Shared Route. *IEEE Transactions on Knowledge and Data Engineering*, 30(2):219–233, feb 2018.

[17] Na Ta, Guoliang Li, Tianyu Zhao, Jianhua Feng, Hanchao Ma, and Zhiguo Gong. An Efficient Ride-Sharing Framework for Maximizing Shared Route. *IEEE Transactions on Knowledge and Data Engineering*, 30(2):219–233, feb 2018.

[18] Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Lei Chen, Jieping Ye, and Ke Xu. A unified approach to route planning for shared mobility. In *Proceedings of the VLDB Endowment*, volume 11, pages 1633–1646. VLDB Endowment PUB4722, jul 2018.

[19] Tanvi Verma, Pradeep Varakantham, Sarit Kraus, and Hoong Chuin Lau. Augmenting decisions of taxi drivers through reinforcement learning for improving revenues. In *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*, pages 409–417, 2017.

[20] Jiachuan Wang, Peng Cheng, Libin Zheng, Chao Feng, Lei Chen, Xuemin Lin, and Zheng Wang. Demand-aware route planning for shared mobility services. *Proceedings of the VLDB Endowment*, 13(7):979–991, mar 2020.

[21] Chak Fai Yuen, Abhishek Pratap Singh, Sagar Goyal, Sayan Ranu, and Amitabha Bagchi. Beyond shortest paths: Route recommendations for ride-sharing. In *The Web Conference 2019 - Proceedings of the World Wide Web Conference, WWW 2019*, pages 2258–2269. Association for Computing Machinery, Inc, may 2019.