Zhou, Ze; Roncoli, Claudio

Computationally efficient dynamic assignment for on-demand ridesharing in congested networks

Published in:
2021 7th International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS 2021

DOI:
10.1109/MT-ITS49943.2021.9529283

Published: 16/06/2021

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Zhou, Z., & Roncoli, C. (2021). Computationally efficient dynamic assignment for on-demand ridesharing in congested networks. In *2021 7th International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS 2021* IEEE. https://doi.org/10.1109/MT-ITS49943.2021.9529283

Computationally efficient dynamic assignment for on-demand ridesharing in congested networks

Ze Zhou, Claudio Roncoli
Department of Built Environment
Aalto University
Espoo, Finland
{ze.zhou, claudio.roncoli}@aalto.fi

***Abstract*—On-demand ridesharing service has been recognized as an effective way to meet travel needs while significantly reducing the number of required vehicles. However, most previous studies investigating dynamic assignment for ridesharing systems overlook the effects on travel times due to the assignment of requests to vehicles and their routes. To better assign the ridesharing vehicles while considering network traffic, we propose a framework that incorporates time-dependent link travel time into the request-vehicle assignment. Furthermore, we formulate an optimal assignment problem that considers multiple path options and that accounts for the congestion potentially caused by assigned routes. A set of simulations reveals that using an appropriate congestion avoidance ridesharing strategy can remarkably reduce passenger average travel and waiting time by alleviating traffic congestion in the network.**

***Index Terms*—traffic control, dynamic ridesharing, network traffic**

I. Introduction

In densely populated areas, urbanization followed by the increasing need for mobility has been constantly deteriorating the existing transportation systems. Meanwhile, the situation is further aggravated by the increasing number of private cars and the consequent low car occupancy rate. Statistics show that there are only 1.7 passengers, on average, on board of private cars, with even fewer passengers for work-based trips [1]. As a mobility supplement of public transit and private cars, the on-demand ridesharing service is noted for its flexibility and potentially low cost. In particular, by grouping requests with similar itineraries and time schedules [2], ridesharing is potentially an effective and flexible way to alleviate congestion by serving the heavy demand with a reduced number of vehicles.

Furthermore, automated driving technology matches perfectly with ridesharing and carsharing services for two reasons. Firstly, autonomous vehicles (AVs) are more compliant and reliable than human drivers, especially when dealing with real-time requests. Secondly, automated driving enables a more efficient and precise relocation of vehicles, without incurring much labor costs. Shared autonomous vehicle (SAV), a fleet of AVs offering short-term carsharing service [3], is currently a prevailing trend in research. Compared to a traditional ride-hailing service, half of the SAV fleet with a ridesharing service is enough to satisfy the same demand [4]. Previous studies [5]–[7] also show that one SAV can replace between 1.18 and 11 traditional private cars, which suggests the potential of SAV in reducing car ownership [8].

Large-scale ridesharing is another crucial research branch as it requires more efficient algorithms to solve this complicated problem. In this branch, minimising fleet size was investigated, e.g., in [9], where an innovative shareability network method is proposed to obtain an optimal solution for a large-scale network. The simulation in [9] shows that 70 percent of vehicles is enough to fulfill the travel needs. Authors of [10] extended the previous work [9] by scaling the problem size and introducing a 4-steps algorithm to cope with high-capacity ridesharing issues. The key idea is to generate a request-trip-vehicle graph considering various time and capacity constraints. Then, the author of [10] solved an integer linear programming to calculate the optimal assignment plan. Their results are even more optimistic than those in [9], showing that 3000 vehicles can meet 98% of the travel demand in New York city. Based on [10], the authors of [11] developed a federated optimization architecture to solve a similar multi-vehicle and -request matching and routing problem. Compared to [10], the algorithm in [11] is capable of obtaining similar results with a considerably lower computational burden. Moreover, the computation can be distributed among different platforms, making it more applicable when dealing with multi-company competition.

Nevertheless, in order to simplify the prediction of the travel time among different origins and destinations, all these studies are based on travel times that are either static or consider historical data. In practice, link travel times are dynamic and depend on the traffic state in each link. Some recent studies have shown signs of shifting from unrealistic assumptions to more precise travel time. For example, the authors of [12] developed an event-based framework to simulate SAVs by a dynamic network loading (DNL) model. Surprisingly, the results indicate that the empty repositioning trips by SAVs cause more congestion than conventional vehicles. However, only simple ridesharing rules are incorporated into their framework and no optimization is implemented to minimize travel costs.

This research is partly funded by the Henry Ford Foundation Finland and the FINEST Twins Center of Excellence (H2020 European Union funding for Research & Innovation grant 856602).

978-1-7281-8995-6/21/$31.00 ©2021 IEEE

Furthermore, [13] investigates the SAV routing problem by developing a linear program based on the link transmission model (LTM) [14]. A similar work has been performed in [15], where, instead of LTM, the cell transmission model is employed to update the traffic states. Nonetheless, both models fails to incorporate a ridesharing service, mainly because including shared rides would further complicate the model and make it hard to get the desired results. On the other hand, the authors of [16] proposed an optimisation problem that considers both congestion and SAVs ridesharing service, where congestion is modelled via BPR functions; nevertheless, due to the high complexity of the nonlinear optimisation problem, the method is computationally expensive.

In this work, we design an operational framework for ridesharing services, which is based on a combination of components with decentralised calculations, which incorporates a congestion prediction module based on a dynamic network loading (DNL) algorithm. Moreover, we propose a computationally efficient optimal assignment problem that allows assigning travel requests to shared vehicles, as well as vehicle routes, while also aiming at minimising congestion occurrence.

The remainder of this paper is structured as follows. Section II elaborates on the proposed operational framework for the ridesharing systems, while Section III presents a vehicle assignment optimisation problem considering the predicted routes overlapping. In Section IV, a simulation experiment is implemented to evaluate the performance of the proposed methodology. Finally, we summarize the main findings and provide some suggestions for future research in Section V.

II. Operational framework for ridesharing

A. Problem statement

We briefly describe the fundamental research problem in this section. Let us consider a real-time vehicle dispatching center where ridesharing requests may arrive randomly. Given a directed graph $G = (N, E)$, where N is the set of nodes and E is the set of edges, a request i is defined as a tuple $r_i = \{t_i^{\mathrm{a}}, t_i^{\mathrm{p}}, t_i^{\mathrm{d}}, n_i^{\mathrm{o}}, n_i^{\mathrm{d}}\}$, which contains request time t_i^{a} (assumed equal to the required pickup time), latest pickup time t_i^{p}, latest drop-off time t_i^{d}, origin $n_i^{\mathrm{o}} \in N$, destination $n_i^{\mathrm{d}} \in N$. We can then introduce the acceptable waiting time $\Delta_i = t_i^{\mathrm{p}} - t_i^{\mathrm{a}}$ and the maximum delay $\Omega_i = t_i^{\mathrm{d}} - t_i^{\mathrm{a}}$. A vehicle j is defined as a tuple $v_j = \{l_j, m_j, p_j\}$, where l_j represents the current vehicle location, m_j denotes the number of on-board users, and p_j is the designated vehicle path.

The following assumptions regarding users and vehicle behaviours are introduced to outline the boundaries of the problem:

- the network is redundant, i.e., there may be several routes for any arbitrary origin-destination (OD) pair;
- unserved requests, including those violating vehicle capacity, maximum waiting and travel time constraints, leave the system automatically;
- the origin and destination of the request belong to the node-set N;

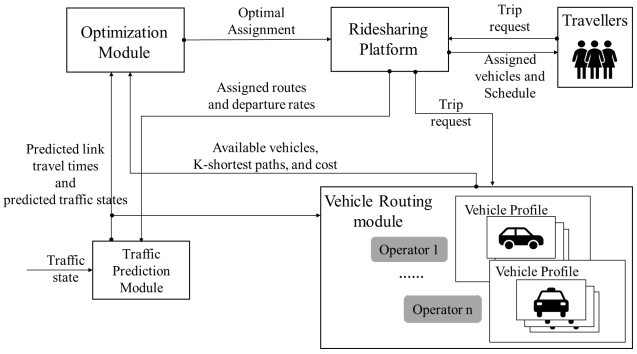


Fig. 1: Operational framework for ridesharing

- there is only one passenger behind each request;
- if there are no passengers on board, the vehicle will stay at the nearest parking lot until a new task is assigned;
- only ridesharing vehicles are considered;
- predicted link travel times are different in different time steps. However, the travel time remains the same during each horizon.

In general, the research problem can be stated as follows: given a ridesharing request set $R = \{r_1, r_2, \ldots\}$ and a vehicle fleet $V = \{v_1, v_2, \ldots\}$, we define an optimal assignment and routes to minimize space-time varying travel cost while satisfying constraints associated to each request.

B. Federated architecture

The operational framework employed in this paper follows a "federated" architecture, similar to the one proposed in [11],which is illustrated in Fig. 1. The strengths of such architecture consists in the fact that the overall (complex) problem is decomposed in separated sub-problems, which are processed in separated modules, where communication and sharing of information is limited. As the internal logic of each module is assumed as essentially independent, a certain level of privacy is maintained. Finally, the decentralised nature of such architecture allows a reduced computational complexity.

As travel requests may arrive at any time into the *Ridesharing platform*, we adopt a batch assignment strategy to group requests every time interval ΔT, i.e., at time t, all requests arrived during the time slot $(t - \Delta T, t]$ are grouped and processed altogether. We normally define ΔT in the order of 10 s–30 s.

These requests are forwarded to a *Vehicle Routing module*, which has the role of identifying the set of vehicles that may potentially serve each of the requests, as well as their respective routes and costs. Existing works, including [11] achieve this by solving a single vehicle Dial-a-Ride-Problem (DARP) [17], [18] for each request-vehicle pair. However, since many alternative paths exist for a single OD pair, considering only the shortest path may cause severe congestion on a particular link. Fig. 2 gives an example to demonstrate the shortcomings of assigning each vehicle to its shortest path. In this abstracted network, where each link has the same travel time, we consider

two vehicles each with a traveller on board: Vehicle 1 needs to transport the traveller from node 0 to node 5, whereas Vehicle 2 needs to transport the traveller from node 6 to node 5. If the system merely outputs the shortest route, for example, Vehicle 1 may be assigned to path $\{0, 1, 4, 5\}$ and Vehicle 2 may be assigned to path $\{6, 7, 4, 5\}$, they may eventually share some links. On the other hand, there was possibility of selecting another path with the same cost, but without route overlapping, e.g. Vehicle 2 may have been assigned to path $\{6, 7, 8, 5\}$. If this behaviour is repeated for a high number of assigned trips, there is a possibility of over-utilising some of the links, whereas some other may remain underutilised. To avoid this happening, multiple paths are calculated for each request-vehicle pair, which entails finding the k-shortest paths that satisfy the set of constraints associated to the request, as well as their related cost. For each request r_i and vehicle v_j, the required ingredients include (i) the request information $\{t_i^{\mathrm{a}}, t_i^{\mathrm{p}}, t_i^{\mathrm{d}}, n_i^{\mathrm{o}}, n_i^{\mathrm{d}}\}$; (ii) vehicle current state $\{l_j, m_j, p_j\}$; and (iii) vehicle pickup cost c_{ij}^{p}, i.e., the cost for traveling from l_j to n_i^{o}. The output is defined as the k-shortest path set R_{ij} and the corresponding trip cost D_{ij}. In order to take into account the sequence of pick-ups and drop-offs, we employ here the same insertion algorithm proposed in [11] to solve the DARP.

As travel time in the network links is time-dependent and, in particular, affected by the assigned trips and routes, we include a *Traffic Prediction module*, which aims at predicting the time-space evolution of traffic based on the current traffic state and the routes and departure rates that are assigned at each step. We propose here to utilise a DNL procedure, such as, e.g., [19]. The required input for this module is a path departure rate matrix, where rows contain the number of paths and columns contain the predicted steps. In our framework, the rate matrix results from the routes assigned at each time step. As vehicles are designated to serve the request along the route, which may change at each step, also the rate matrix needs to be updated at each iteration to reload the network traffic. The output of this module consists in the link travel time and number of vehicles in each link for the predicted horizon.

Finally, the *Optimisation module* computes the optimal combination of request-vehicle pairs that minimise a given cost, by utilising information on the available vehicles, their routes, and costs, as well as the predicted traffic states. The details on this module are provided in the following section.

III. Optimization Module: Extended Linear Assignment Problem

This section presents a novel linear assignment problem to determine the request-vehicle assignment and the corresponding routes, which is solved at each step. As previously mentioned, for each request-vehicle pair, we assume available a set of k-shortest paths $\Phi_{i,j} = \{R_{i,j,1}, R_{i,j,2}, ..., R_{i,j,k}\}$ and related travel time set $\Psi_{i,j} = \{D_{i,j,1}, D_{i,j,2}, ..., D_{i,j,k}\}$, where k denotes the number of shortest path. A time tolerance ε may be included, to control the time difference between shortest and longest path costs, so that $\max(\Psi_{i,j}) - \min(\Psi_{i,j}) \leq \varepsilon$.

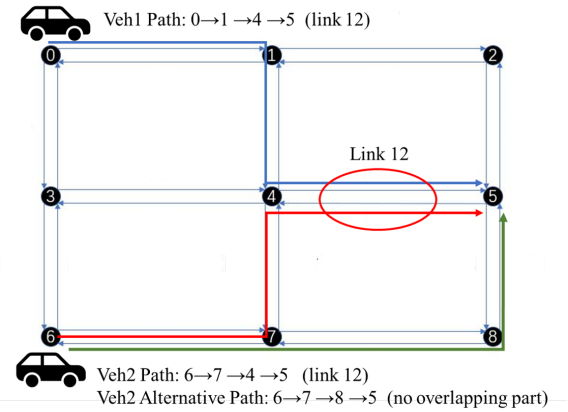


Fig. 2: Shortest path overlap versus multiple path

The problem is formulated by considering integer decision variables $x_{v,r,a}$, which indicate if vehicle v is assigned to serve request r along the path alternative a. Introducing elements $\beta_{v,r,a}$ of the path cost matrix, which can be computed for each request-vehicle pair from Ψ, we can calculate the total travel and waiting cost as

$$\sum_{v}\sum_{r}\sum_{a} x_{v,r,a}\beta_{v,r,a}. \tag{1}$$

We now introduce binary parameter $\alpha_{v,r,a,l}$, which can be derived from Φ, and indicates if link l is included in the alternative path a considered for request r and vehicle v. Consequently, we can calculate the total number of vehicles assigned to each link l as

$$A_l = \sum_{v}\sum_{r}\sum_{a} x_{v,r,a}\alpha_{v,r,a,l}, \quad \forall l. \tag{2}$$

We denote ρ_l as the relative remaining capacity for a link l with respect to the other links in the network, defined as

$$\rho_l = \frac{C_l - q_l^t}{\sum_l C_l - q_l^t}, \tag{3}$$

where C_l is the link capacity and q_l^t denotes the link traffic at time t. We then introduce the following cost to prioritize vehicle assignment to links with higher capacity available

$$\sum_{l}(1-\rho_l)A_l. \tag{4}$$

As such, the multiple path ridesharing and vehicle assignment problem can be formulated as the following mixed integer linear programming (MILP) problem

$$\min_{x}\left(\theta\sum_{l}(1-\rho_l)A_l + (1-\theta)\sum_{v}\sum_{r}\sum_{a} x_{v,r,a}\beta_{v,r,a}\right) \tag{5}$$

subject to

$$\sum_{r}\sum_{a} x_{v,r,a} \leq 1, \quad \forall v \tag{6}$$

$$\sum_{v}\sum_{a} x_{v,r,a} \leq 1, \quad \forall r \tag{7}$$

$$\sum_{v}\sum_{r}\sum_{a} x_{v,r,a} = \min\{m, n\} \tag{8}$$

$$x_{v,r,a} \in \{0, 1\}, \tag{9}$$

where θ is a trade-off parameter ranging from 0 to 1. Note that the cost function (5) degenerates to a minimum assignment problem when $\theta = 0$. Constraints (6) guarantees that one vehicle can serve at most one request via one route at a time; similarly, (7) guarantees that a passenger can be picked up at most by one vehicle along a path. Eq. (8) ensures that the number of assignment results is equivalent to the smaller number between requests and fleet size. Finally, (9) indicates $x_{v,r,a}$ is a binary decision variables. This MILP problem can be solved directly by existing optimization solvers, such as Gurobi [20].

IV. Numerical simulation

A. Implementation set-up

The proposed strategy is tested via simulation using the network shown in Fig. 2, which consists of 9 nodes and 24 uni-directional links. The link attributes includes capacity, length, free-flow time, which are set to 0.13 veh/s, 2 km, and 0.04 h, respectively. We define a request arrival rate following the Poisson distribution for a 1-hour simulation horizon, which is further divided into three periods. The request arrival intensity λ, namely the expected number of requests entering the system during a unit time, is 30, 60, and 30 requests per minute, respectively. In total, there are 2400 requests that wait to be served, while 600 vehicles with four seats are randomly introduced in the network. We assume each request has some tolerance in maximum waiting time and delay, both set as 5 minutes. The requests are grouped every 10 s, considering a trade-off between waiting time and solution quality. Although the simulation step is 10 s, the link travel time is updated every 10 steps to improve computational efficiency.

Three scenarios, each with two variants that consider ridesharing or not, are devised to evaluate the proposed strategy.

- Scenario 1: static travel time + minimizing travel cost. While assigning requests to vehicles, the shortest path cost between arbitrary two points is calculated based on static travel time, which equals to the link length divided by the free-flow speed; only one path is computed for each request-vehicle pair; whereas, within the optimization module, we set $\theta = 0$, implying that only travel cost minimisation is considered. However, the simulation still takes into account realistic link travel time and congestion phenomena, calculated via the DNL algorithm.
- Scenario 2: time-dependent travel time + minimizing travel cost. In contrast to Scenario 1, spatio-temporal varying travel time is considered when assigning and dispatching vehicles. Namely, when we calculate the travel times, the predicted link travel time is employed to capture the actual traffic state. Nevertheless, we still consider only one path for each request-vehicle pair and we employ $\theta = 0$.
- Scenario 3: time-dependent travel time + congestion avoidance. In this scenario, not only dynamic travel time is considered, but also the multi-path congestion avoidance strategy is adopted. In particular, we consider 3 possible paths for each request-vehicle pair and $\theta = 0.8$ to emphasize the congestion avoidance objective term.

All the experiments are coded in Python 3.7 and run on an i7-8750H 2.20 GHz computer with 8 GB RAM. For the DNL algorithm, we adopt the Matlab code package provided in [19]. The Python Matlab engine is thus used to bridge the code gap. Gurobi Optimizer is used to solve the optimisation problem.

B. Results and discussion

Table I shows a comparison of the simulation results in terms of waiting time, travel (in-vehicle) time, network congestion, request-response, and request-completion ratio. The network congestion is evaluated by index μ, which is computed as follows

$$\mu = \frac{\sum_l (\frac{\sum_1^T t_i^l}{n} - t_0^l)}{t_0^l}, \tag{10}$$

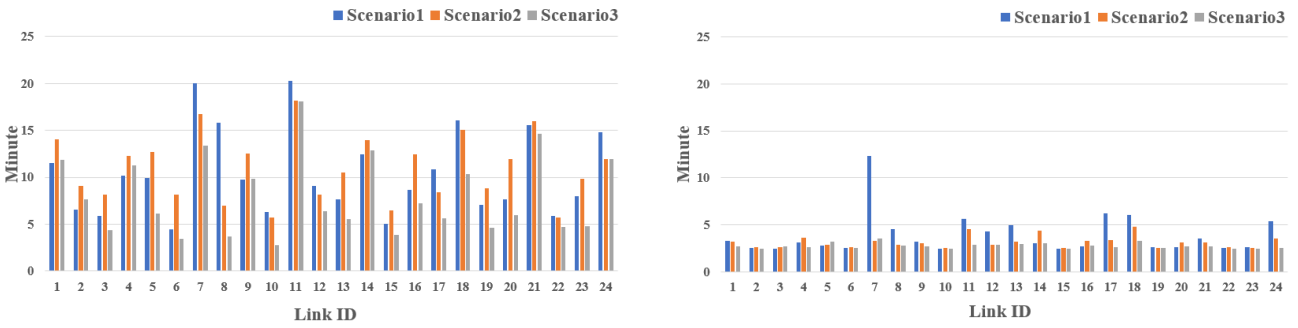
where t_0^l is the free-flow time in link l; t_i^l represents the link travel time during $[t_{i-1}^l, t_i^l]$ for link l; n is the number of time steps; and L denotes the total number of links. Across all the links and time steps, the average travel time is μ times the average free-flow time $\frac{\sum_l t_0^l}{L}$. If we consider, for example, Scenario 1 without ridesharing, a ratio $\mu = 3.33$ means averagely travel time is 3.33 times longer compare to free-flow time; that is, since free-flow time for all links is 2.4 minutes, the average travel time in the network is 10.4 minutes.

1) Passenger waiting and travel time

In Scenario 1 without ridesharing, users wait, on average, 5.39 minutes until they are picked up by a vehicle. Such high value is mainly because vehicles are assigned based on static travel time and, when congestion grows, the real pick-up time increases significantly from the forecasted one. On the other hand, in the same scenario with ridesharing, this number remarkably decreases to 0.93 minutes. As rideshring reduces the congestion ratio from 3.33 to 0.63, the error between the actual travel time and the predicted time is small. A similar trend can be seen also in terms of average travel time: without ridesharing, it takes, on average, 19.04 minutes to transport a single passenger to his or her destination, whereas, if they are willing to share rides, the average in-vehicle time drops to 8.49 minutes. In Scenario 2, the average waiting time drops to 0.1 minutes without ridesharing, which is 0.5 minutes shorter than with the ridesharing service. That is reasonable because many vehicles may be located at each passenger's origin and a vehicle can pick up the traveller virtually immediately, if it only serves one user. In comparison, ridesharing vehicles need to visit to several origins to pick up different users, causing a longer waiting time. In Scenario 3, the average waiting time is slightly longer than Scenario 2, no matter if it offers

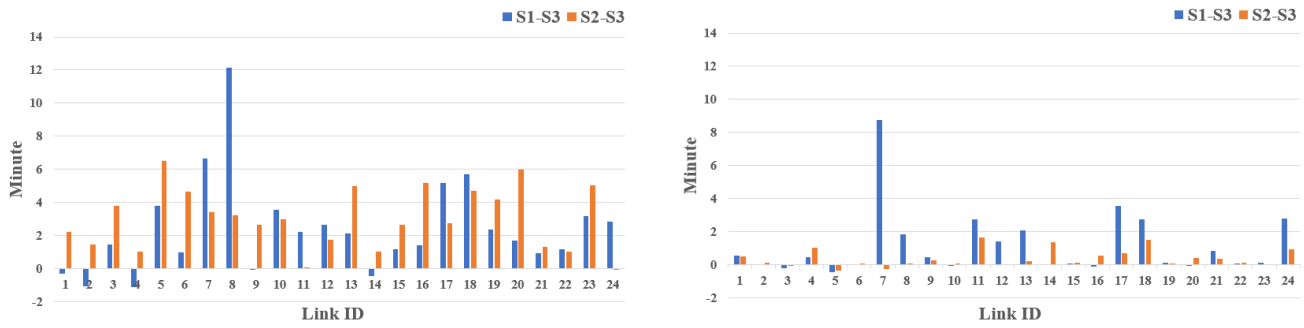
TABLE I: Comparison results between different scenarios

Scenario	Ridesharing	Traveler waiting time (min)		Traveler in-vehicle time (min)		Network congestion ratio	Request response ratio	Request completion ratio
		Mean	Std.	Mean	Std.			
1	N(Capacity 1)	5.39	8.43	19.04	14.3	3.33	0.78	0.65
	Y(Capacity 4)	0.93	1.66	8.49	6.48	0.63	1	0.93
2	N	0.1	0.81	18.67	11.28	3.58	0.65	0.74
	Y	0.61	0.84	5.93	2.91	0.32	1	0.94
3	N	0.29	1.06	14.55	8.96	2.85	0.80	0.88
	Y	1.14	1.02	5.16	2.54	0.14	1	0.94



(a) No ridesharing (b) Ridesharing

Fig. 3: Average link travel time comparison



(a) No ridesharing (b) Ridesharing

Fig. 4: Link travel time difference compare to scenario 3

ridesharing or not, since only the shortest path is considered in Scenario 2. Nevertheless, travel time decreases by 4.12 and 0.77 minutes with a 0.19 and 0.53-minute sacrifice in waiting time. Comparing Scenarios 1 and 3, both passenger waiting time and in-vehicle time drop significantly, which is quantified in 5 minutes and 4.5 minutes in waiting time and travel time separately for no shared rides. With ridesharing, the waiting time slightly increases by 0.21 minutes, whereas the user travel time decreases by 3.33 minutes. Finally, it is worth noting that, in Scenario 3, the standard deviation of the in-vehicle time is always lower than the other scenarios, meaning that the reliability of the travel time is increased.

2) Link and network congestion

As shown in table I, the usage of ridesharing is capable of reducing network congestion by approximately 90%. In Scenario 1, the ratio drops from 3.33 to 0.63 with the help of shared rides. We classify the network congestion ratio into two groups based on whether the system offers ridesharing service or not. For the non-ridesharing group, time-dependent and congestion avoidance strategy can only slightly reduce the congestion by 15%. However, the strategy performed better in a ridesharing setting. Apart from the overall network congestion rate, we compare the average link travel time across 360-time steps in different links. Fig. 3 shows the time-mean link travel time across different scenarios and we can observe that, irrespectively from using static or time-dependent travel

times, ridesharing significantly reduces the average travel time. To evaluate the performance of the proposed strategy, we compare both Scenarios 1 and 2 with Scenario 3. In Fig. 4, we show that congestion avoidance strategy can significantly reduce link travel time. This is particularly evident for highly congested links, such as link 7, 8, 17, 18, where the congestion avoidance strategy alleviate congestion by assigning vehicles according to the real-time available link space.

3) Request response and completion rate

The request-response ratio is calculated by dividing the number of answered requests by the total request number, while the number of completed requests (customer arrives at his/her destination) to the total request number ratio is the request completion index. All ridesharing cases can achieve a 100 percent response rate. In Scenario 1, this ratio is even higher than in Scenario 2 without ridesharing, since vehicles in Scenario 1 may optimistically estimate the travel time. On the other hand, only 65 % customers arrived at their destination by the end of the simulation horizon.

V. Conclusions

Despite the fact that previous studies overlooked the congestion effect caused by ridesharing service vehicles or SAVs, when vehicle fleets will be large enough, these vehicles would remarkably disrupt the transportation system. In this paper, we proposed a computationally efficient strategy to solve the dynamic ridesharing vehicle assignment problem in the presence of dynamic travel time. In particular, we proposed a framework that includes a DNL module to consider time-dependent link travel time in the request-vehicle assignment. Furthermore, we formulated an optimal assignment problem that considers both multiple path options and the congestion potentially caused by assigned routes. The proposed strategy is tested via simulation in an artificial network and the results indicate that our model can significantly reduce congestion comparing to static or shortest path assignment. Moreover, the simplified problem settings guarantees that the assignment result is achieved within reasonable computation time.

Nevertheless, several open questions remain unsolved. The current study assumes that vehicle stops once arrived at their destination if there are no passengers on board; however, how to effectively rebalance those empty cars is also crucial for congestion reduction. We also assume that there are only ridesharing vehicles in the network, while other travel modes, such as private cars and buses, exist in a real transportation system, with implications on both travellers choices and congestion impact. Finally, incorporate pricing and travellers preferences into the model would make these results more practical for ridesharing companies.

References

[1] A. Santos, N. McGuckin, H. Y. Nakamoto, D. Gray, S. Liss *et al.*, "Summary of travel trends: 2009 national household travel survey," United States. Federal Highway Administration, Tech. Rep., 2011.

[2] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Optimization for dynamic ride-sharing: A review," *European Journal of Operational Research*, vol. 223, no. 2, pp. 295–303, 2012.

[3] W. Zhang, S. Guhathakurta, J. Fang, and G. Zhang, "The performance and benefits of a shared autonomous vehicles based dynamic ridesharing system: An agent-based simulation approach," in *Transportation Research Board 94th Annual Meeting*, no. 15-2919, 2015.

[4] J. Farhan and T. D. Chen, "Impact of ridesharing on operational efficiency of shared autonomous electric vehicle fleet," Tech. Rep., 2018.

[5] L. Martinez and P. Crist, "Urban mobility system upgrade–how shared self-driving cars could change city traffic," in *International Transport Forum, Paris*, 2015.

[6] H. Dia and F. Javanshour, "Autonomous shared mobility-on-demand: Melbourne pilot simulation study," *Transportation Research Procedia*, vol. 22, pp. 285–296, 2017.

[7] D. J. Fagnant and K. M. Kockelman, "Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in austin, texas," *Transportation*, vol. 45, no. 1, pp. 143–158, 2018.

[8] S. Narayanan, E. Chaniotakis, and C. Antoniou, "Shared autonomous vehicle services: A comprehensive review," *Transportation Research Part C: Emerging Technologies*, vol. 111, pp. 255–293, 2020.

[9] M. M. Vazifeh, P. Santi, G. Resta, S. H. Strogatz, and C. Ratti, "Addressing the minimum fleet problem in on-demand urban mobility," *Nature*, vol. 557, no. 7706, pp. 534–538, 2018.

[10] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proceedings of the National Academy of Sciences*, vol. 114, no. 3, pp. 462–467, 2017.

[11] A. Simonetto, J. Monteil, and C. Gambella, "Real-time city-scale ridesharing via linear assignment problems," *Transportation Research Part C: Emerging Technologies*, vol. 101, pp. 208–232, 2019.

[12] M. W. Levin, K. M. Kockelman, S. D. Boyles, and T. Li, "A general framework for modeling shared autonomous vehicles with dynamic network-loading and dynamic ride-sharing application," *Computers, Environment and Urban Systems*, vol. 64, pp. 373–383, 2017.

[13] M. W. Levin, "Congestion-aware system optimal route choice for shared autonomous vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 82, pp. 229–247, 2017.

[14] I. Yperman, S. Logghe, and B. Immers, "The link transmission model: An efficient implementation of the kinematic wave theory in traffic networks," in *Proceedings of the 10th EWGT Meeting*. Poznan Poland, 2005, pp. 122–127.

[15] P. Venkatraman and M. W. Levin, "A congestion-aware tabu search heuristic to solve the shared autonomous vehicle routing problem," *Journal of Intelligent Transportation Systems*, pp. 1–13, 2019.

[16] X. Liang, G. H. de Almeida Correia, K. An, and B. van Arem, "Automated taxis' dial-a-ride problem with ride-sharing considering congestion-based dynamic travel times," *Transportation Research Part C: Emerging Technologies*, vol. 112, pp. 260–281, 2020.

[17] N. Marković, R. Nair, P. Schonfeld, E. Miller-Hooks, and M. Mohebbi, "Optimizing dial-a-ride services in maryland: benefits of computerized routing and scheduling," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 156–165, 2015.

[18] S. C. Ho, W. Szeto, Y.-H. Kuo, J. M. Leung, M. Petering, and T. W. Tou, "A survey of dial-a-ride problems: Literature review and recent developments," *Transportation Research Part B: Methodological*, vol. 111, pp. 395–421, 2018.

[19] K. Han, G. Eve, and T. L. Friesz, "Computing dynamic user equilibria on large-scale networks with software implementation," *Networks and Spatial Economics*, vol. 19, no. 3, pp. 869–902, 2019.

[20] Gurobi Optimization LLC, "Gurobi optimizer reference manual," 2020. [Online]. Available: http://www.gurobi.com