

# Sink Proximity: A Novel Approach for Online Vehicle Dispatch in Ride-hailing

Ruiting Wang, Jiaman Wu, Fabio Paparella, Scott J. Moura, Marta C. Gonzalez

**Abstract**—Ride-hailing platforms have a profound impact on urban transportation systems, and their performance largely depends on how intelligently they dispatch vehicles in real time. In this work, we develop a new approach to online vehicle dispatch that strengthens a platform's ability to serve more requests under demand uncertainty. We introduce a novel measure called *sink proximity*, a network-science-inspired measure that captures how demand and vehicle flows are likely to evolve across the city. By integrating this measure into a shareability-network framework, we design an online dispatch algorithm that naturally considers future network states, without depending on fragile spatiotemporal forecasts. Numerical studies demonstrate that our proposed solution significantly improves the request service rate under peak hours within a receding horizon framework with limited future information available.

## I. INTRODUCTION

Shared mobility services have greatly transformed urban transportation and provided a convenient alternative to reduce the use of private vehicles [1]. The global market size of ride-hailing was 106.66 billion in 2023 and is projected to grow from \$123.08 billion in 2024 to \$480.09 billion by 2032 [2]. It is fundamental for these shared mobility services to become both efficient and profitable. Platform-based operators without vehicle ownership primarily rely on increasing the volume of successfully matched driver–rider pairs to boost revenues [3]. A core challenge is the platform's ability to effectively assign drivers to incoming ride requests, a problem commonly referred to as the *Vehicle Dispatch Problem* (VDP).

From an economic standpoint, the dispatch algorithm significantly influences how effectively the platform balances demand from riders and supply of drivers, thereby reducing driver idle time and lowering passenger waiting time, and optimizing revenue for the platform [4], [5]. A good matching strategy also influences energy consumption and environmental impact, particularly in electrified fleets where smart coordination with power grids is essential [6], [7].

Besides financial incentives, there are important environmental and social motivations for improving the VDP. Strategically matching drivers to riders can minimize unnecessary vehicle movements, reduce overall energy consumption, and

decrease emissions [8]–[11]. Such strategies also help mitigate traffic congestion during peak hours [12], [13]. Thus, enhancements to the VDP can support wider sustainability goals by making transportation systems cleaner, more efficient, and more resilient.

This paper proposes a novel network science approach to the VDP, which is formulated as a network flow problem and solved in a receding horizon control (RHC) fashion. By incorporating a specialized measure derived from network science, we demonstrate how the request service rate (i.e., the fraction of riders served) can be improved compared to a standard RHC formulation.

### A. Vehicle Dispatch in Ride-hailing

With the increasing size of the market and the large number of drivers and riders in the platform, assigning drivers to real-time ride requests becomes increasingly complex. Besides the sheer scale of the problem, the inherent unpredictability of future demand adds another layer of difficulty. Research on VDP typically splits into two main methodological approaches:

1) *Optimization-Based Methods*: Combinatorial optimization algorithms often present the problem as a variant of the vehicle routing problem. This format is more flexible and allows for customized constraints. To address the lack of information in online optimization and uncertainty on the demand side, studies have used several approaches, such as Bayesian framework [14], multi-stage stochastic optimization [15], sampling [16], etc. The primary challenge in employing combinatorial optimization techniques also lies in the computational complexity, which is crucial for online implementation in large-scale systems.

2) *Machine Learning and Reinforcement Learning*: In recent years, studies have based their work on machine learning and reinforcement learning (RL) models, to handle uncertain and fast-evolving environments, where dispatch decisions must be updated within seconds [17]–[19]. These methods can be adept at learning from data in real time but often struggle with transferability if new conditions differ significantly from training data [20], [21]. Readers can refer to these survey papers [22], [23] for a comprehensive overview of how different methods are utilized in different problems in ride-hailing, including but not limited to VDP.

A middle-ground solution uses network flow formulations, which can be solved efficiently by exploiting graph-theoretic properties [24]–[26]. A variety of complex and realistic situations can be implemented by modifying the structure, and

The authors are with the Department of Civil and Environmental Engineering, University of California, Berkeley, CA, 94720 USA. {rtwang, fabio\_paparella, smoura, martag}@berkeley.edu

J.Wu is with the Department of Data Science, City University of Hong Kong, Hong Kong SAR, China. {jm.wu}@cityu.edu.hk

F. Paparella is also with the Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, 5600 MB, the Netherlands. f.paparella@tue.nl.

recent studies also incorporate ideas from learning frameworks into these classical algorithms to balance computational tractability with adaptive capabilities [16], [27]–[29].

Regardless of the algorithms used, the choice of objective can also vary, reflecting the diverse needs and constraints of ride-hailing platforms. To highlight a few directions, besides matching algorithms, some studies focus on: dynamic pricing [30], electric fleets [7], interaction with public transit [8], the fairness among drivers [31], and evaluation of environmental, social, and economic benefits of ride-hailing [10].

### B. Network Science for Vehicle Dispatch

Network science offers a variety of measures (e.g., in-degree, betweenness, closeness [32]–[34]) to analyze the structure of complex networks. However, these measures typically assume static graphs. Real-world shareability networks in ride-hailing are dynamic, meaning nodes (ride requests) and edges (potential matches) evolve over time. Although some work has studied measures for dynamic networks (e.g., communication [35], social networks [36]), no existing research has linked network science measures to dynamic shareability networks in ride-hailing.

Addressing this gap, specifically, how future network states may influence current matching decisions—can yield valuable insights and more efficient online algorithms for the VDP. This work aims to fill the gap by designing a tool that can be leveraged in network flow problems in a dynamic network setting. In particular, we focus on the temporal evolution of the shareability network and design a novel network science measure, *sink proximity* (SP), that captures the potential importance of each node in the network flow.

### C. Statement of Contribution

This study utilizes the network topology of a shareability network in ride-hailing problems through the application of network science to enhance the efficiency of ride-hailing platforms. We claim the following contributions:

1) *Novel Measure—Sink Proximity*: We introduce *sink proximity* in the context of network flow problems, which quantifies the longest path distance from any given node to the sink node in a network flow. Within shareability networks, this measure helps to account for downstream opportunities and incorporates elements of future information regarding the network's expansion.

2) *Efficient Computation in a DAG*: We demonstrate how *sink proximity* can be efficiently approached as a single source longest path problem in a directed acyclic graph (DAG), which can be solved in polynomial time. This ensures that real-time ride-hailing systems can feasibly integrate the measure into online algorithms.

3) *Application to Vehicle Dispatch*: We incorporate sink proximity into a network-flow-based receding horizon framework (RHC-SP). Simulation results show how this approach boosts the request service rate relative to standard RHC solutions. This algorithm represents a significant advancement in optimizing the dispatch process through the application of network science principles.

Without loss of generality, we focus on the standard VDP, though the proposed method can be adapted to variations, such as profit maximization [37] or energy-aware dispatching [7]. The objective of this study is to enhance the operational efficiency of ride-hailing platforms by optimizing the Request Service Rate (RSR), i.e., the number of orders served normalized by the overall number of orders of the vehicle dispatch problem.

## II. PROBLEM FORMULATION

In this section, first we formally define the orders and drivers. Then we define the dispatching problem as an integer linear problem as Problem 1. Then, we leverage a vehicle-shareability network approach to transform the Problem 1 into a directed acyclic graph formulation (Problem 2). This allows us to use polynomial-time machinery developed for minimum/maximum-cost flow problems to solve the dispatching problem efficiently.

### A. Vehicle Dispatch Problem Formulation

The dispatch problem within a ride-hailing platform involves three key participants: the platform itself, drivers, and passengers. The platform aggregates a collection of orders and drivers, which we define in the following. We refer to this challenge as the “vehicle dispatch problem”.

**Definition 1.** We define an order as a tuple  $\mathcal{O}_i = (t_i^p, t_i^d, o_i, d_i) \in \mathcal{T} \times \mathcal{T} \times \mathcal{Y} \times \mathcal{Y}$ , where  $\mathcal{T}$  is the simulation time period and  $\mathcal{Y}$  is the set of locations in which an order can start and end the trip. Then,  $t_i^p$  and  $t_i^d$  are the anticipated pick-up and drop-off times, while  $o_i$  and  $d_i$  indicate the origin and destination locations, respectively. Last, we define the set of orders  $\mathcal{O} := \{\mathcal{O}_i\}_{i \in \mathcal{N}}$ , where  $\mathcal{N} = [1, 2, \dots, N]$  is the set of indices of such orders.

It is assumed that if a passenger  $i^1$  places an order at time  $t_i^s$  with a maximum waiting time of  $t_i^w$ , then the expected pick-up time is  $t_i^p = t_i^s \leq t_i^w$ .

**Definition 2.** We define a driver as a tuple  $D_k = (t_k^r, p_k^t) \in \mathcal{T} \times \mathcal{Y}$ , where  $t_k^r$  indicates the time they begin to accept orders, and  $p_k^t$  signifies their location at time  $t_k^r$ . Last, we define a set of drivers  $\mathcal{D} := \{D_k\}_{k \in \mathcal{M}}$ , where  $\mathcal{M} = [1, 2, \dots, M]$  is the set of indices of such drivers.

Importantly, drivers have a maximum idling time, formally defined by  $t_k^{idle} \leq T_p, \forall k \in \mathcal{M}$ .

The platform seeks to maximize the number of served orders while satisfying the maximum idle time constraints. The resulting integer linear programming (ILP) [29] is defined in the following.

### Problem 1. (Vehicle Dispatch Problem)

<sup>1</sup>For the remainder of the paper, we use interchangeably  $\mathcal{O}_i$  and  $i$  to indicate the  $i$ -th order, and  $D_k$  and  $k$  to indicate the  $k$ -th driver.

Given a set of orders  $\mathcal{O}$  and a set of drivers  $\mathcal{D}$ , the decision variables  $x_{ij}^k, s_i^k, t_i^k$  that maximize the RSR, result from

$$\max \frac{\sum_{i,j,k} x_{ij}^k - 1}{N} \quad (1)$$

$$\text{s.t.} \quad \sum_{k,j} x_{ij}^k \leq 1, \quad \forall i, \quad (2)$$

$$\sum_j x_{ij}^k - \sum_j x_{ji}^k = s_i^k - t_i^k, \quad \forall i, k, \quad (3)$$

$$\sum_i s_i^k \leq 1, \quad \sum_i t_i^k \leq 1, \quad \forall k, \quad (4)$$

$$x_{ij}^k \leq \hat{x}_{ij}^k(T_p), \quad x_{ij}^k, s_i^k, t_i^k \in \{0, 1\} \quad \forall i, j, k, \quad (5)$$

where the objective is the RSR (the  $-1$  takes into account the trips to the virtual nodes), and where  $\hat{x}_{ij}^k(T_p)$  is an upper-bound of variable  $x_{ij}^k$ , and it is defined as follows:

$$\hat{x}_{ij}^k = \begin{cases} 1, & \min(v(t_i^d - t_j^p), vT_p) \geq L(d_i, o_j) \wedge t_i^p \geq t_k^r, \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

The decision variables of Problem 1 are  $x_{ij}^k \in \{0, 1\}$ ,  $s_i^k \in \{0, 1\}$ , and  $t_i^k \in \{0, 1\}$ . They respectively denote i) whether driver  $k$  picks up order  $j$  after serving  $i$ ; ii) whether order  $i$  is the first or iii) the last order for driver  $k$  in the operational period. The upper-bound  $\hat{x}_{ij}^k(T_p)$  represents the compatibility between order  $i$  and  $j$ , and driver  $k$  with maximum idle time  $T_p$ , and indicates if order  $j$  can be reached after a driver finishes serving order  $i$ . The parameter  $v$  denotes the average speed of the vehicles, while  $L(d_i, o_j)$  denotes the distance between location  $d_i$  and  $o_j$ . The constraints (2) indicate that one order can only be fulfilled by one driver. Meanwhile, the constraints (3) allow each driver to serve a sequence of orders, except for their first or last order. Additionally, the constraints (4) ensure that each driver has at most one first order and last order. Finally, the constraints (6) eliminate unfeasible connections in Eq. (5).

This stylized vehicle dispatch problem can be solved with a set of heuristics that provide good feasible solutions [38].

Note that the platform can chose to not assign a vehicle to a particular order if it is not convenient. In that case, the order remains unfulfilled, and the user can either leave the platform or ask again for the service. However, there is no guarantee that an user will be served.

### B. Network Flow Formulation

To cast Problem 1 as a network flow problem, we define an extended shareability network,  $\mathcal{G}^{\text{ext}} = (\mathcal{V}^{\text{ext}}, \mathcal{E}^{\text{ext}})$ . The set of nodes is defined as  $\mathcal{V}^{\text{ext}} := \mathcal{O} \cup \mathcal{D} \cup s \cup t$ , and it is the union of the order nodes  $\mathcal{O}$ , driver nodes  $\mathcal{D}$ , virtual source  $s$  and virtual sink  $t$ . Each order node in the extended shareability network  $\mathcal{G}^{\text{ext}}$  is further detailed by two nodes, which represent the origin and destination nodes, as shown in Fig. 1(c) and in Fig. 1(d). The set of links is defined as  $\mathcal{E}^{\text{ext}} := \mathcal{E}^v \cup \mathcal{E}^c \cup \mathcal{E}^i$ .

Thus, there are three types of links in the extended shareability network: i) the virtual links that connect the virtual node  $s$  to drivers, or orders/drivers to  $t$ ; ii) the connectivity links between drivers and orders, and between orders; and iii) the internal links inside each order node.

The problem can be represented as the following maximum-cost flow problem (MCFP).

### Problem 2. (Maximum Cost Flow Representation of VDP)

Given a directed flow network  $\mathcal{G}^{\text{ext}} = (\mathcal{V}^{\text{ext}}, \mathcal{E}^{\text{ext}})$ . For each edge  $(i, j) \in \mathcal{E}^{\text{ext}}$ , a flow  $f_{ij}$ , an upper bound  $u_{ij} = 1$ , a lower bound  $l_{ij} = 0$ , and a weight  $w_{ij}$  are defined.

$$\max_{f_{ij}} \sum_{(i,j) \in \mathcal{E}^{\text{ext}}} w_{ij} f_{ij} \quad (7)$$

$$\text{s.t.} \quad l_{ij} \leq f_{ij} \leq u_{ij}, \quad \forall (i, j) \in \mathcal{E}^{\text{ext}}, \quad (8)$$

$$\sum_{j \in \mathcal{V}^{\text{ext}}} f_{ij} = 0, \quad \forall i \in \mathcal{V}^{\text{ext}} \setminus \{s, t\}, \quad (9)$$

$$\sum_{j \in \mathcal{V}^{\text{ext}}} f_{sj} = d, \quad \sum_{i \in \mathcal{V}^{\text{ext}}} f_{it} = d \quad (10)$$

The upper and lower bounds ensure that each order is allocated to a maximum of one driver, similar to constraint (2). The weights  $w_{ij}$  of the internal links are set to 1, while the ones of all the others are set to 0. The element  $d$  can arbitrarily be either a variable or a parameter, representing the total number of drivers dispatched.

Fig. 1 depicts a toy example of the shareability network, and visually demonstrates how Problem 1 is cast into Problem 2. In Fig. 1(a), the green and red routes represent two subsequent orders. The first order (green) starts at 9 am and ends at 10 am, while the second order (red) starts at 11 am and ends at noon. The yellow route indicates that it takes 30 min to travel from the drop-off location of the first order to the pickup location of the second order. Since it is possible to travel from the destination of the first order to the origin of the second order before the pick-up time of the second order, we abstract the two orders with two squares connected with a link that goes from square 1 to square 2. The two trips correspond to two nodes in the vehicle-shareability network, and are connected by a ‘‘connectivity link’’. Fig. 1(b) shows a simplified shareability network without internal links. The simplified network allows to decrease the number of nodes and links in the network while maintaining the connectivity information between order nodes. Fig. 1(c) shows the structure of the order nodes. The attributes  $(1, 0, w)$  on the internal link are arranged in the following sequence: upper bound, lower bound, edge weight. Finally, Fig. 1(d) presents the extended shareability network with orders, drivers, virtual nodes, as well as internal links, connectivity links, and virtual links. Last, the required number of drivers is the overall flow from the source  $s$  to the sink  $t$ . The solution to the maximum cost flow problem corresponds to the optimal dispatch. Various efficient solution techniques have been well investigated for minimum-cost flow problems [39], which guarantee computational tractability.

### III. ALGORITHMS

In the previous section we detailed how to formulate the VDP leveraging network flow models, i.e., formulating

<sup>2</sup>Link attributes are in this order: upper bound, lower bound, edge weight.

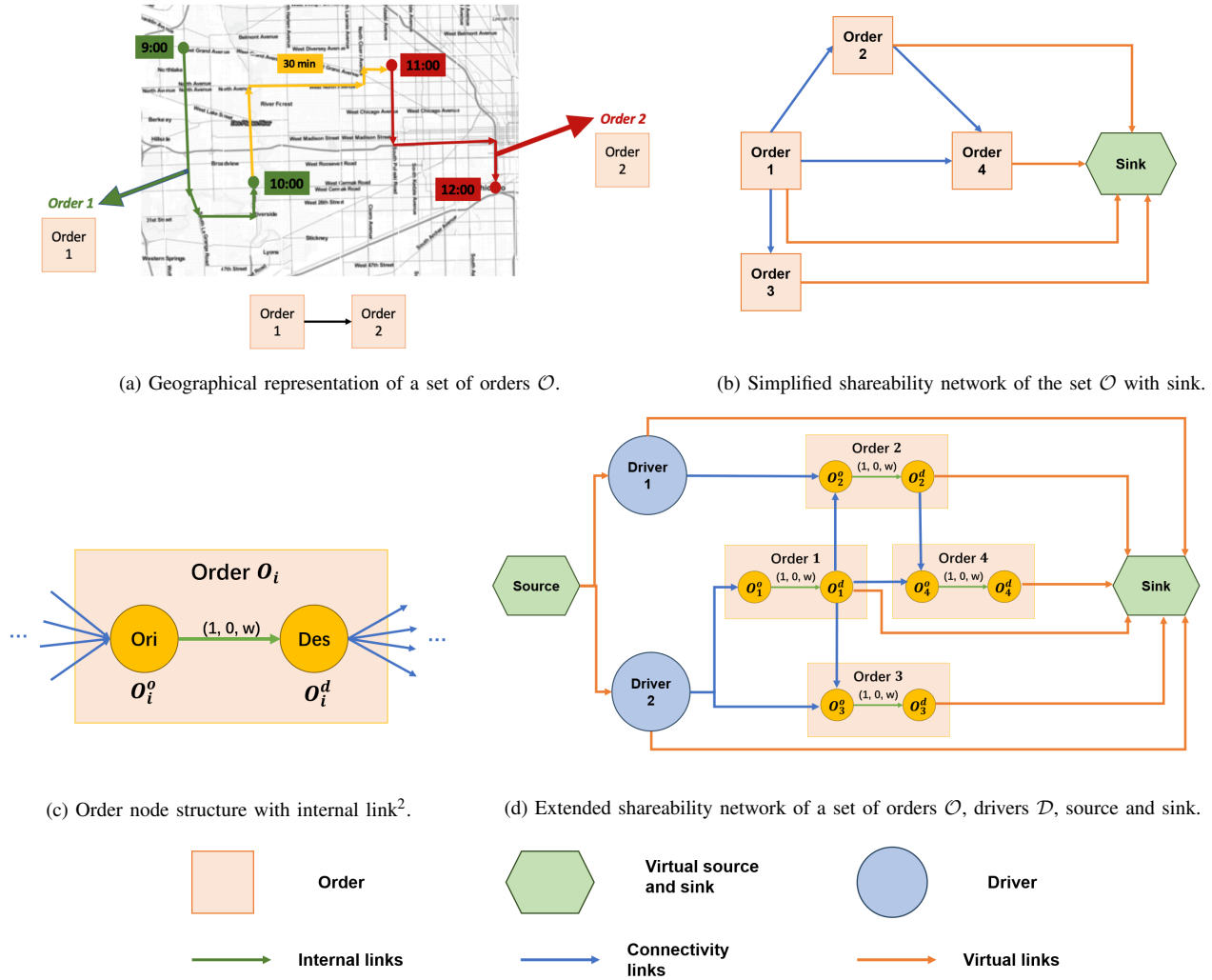


Fig. 1. Geographical representation, and shareability networks for a set of orders  $\mathcal{O}$  and drivers  $\mathcal{D}$ . The type of links is detailed at the bottom of the figure.

Problem 1 into Problem 2. This section investigates different algorithmic settings in which the maximum-cost flow optimization problem can be approached. First, in Section III-A and III-B we introduce two intuitive methodologies, offline and online, to solve the problem. We then present the formal definition of *Sink Proximity* in Section III-C and show in Section III-D how to leverage it to improve the previously proposed methodologies.

### A. Offline Algorithm

In the offline approach, in which all the information is known *a priori*, it is possible to compute a globally optimal solution, which can be used as a benchmark to assess the quality of online approaches. Here we solve the problem as a maximum cost network flow problem, using *NetworkX* package in Python [40]. The construction of the network, and the result retrieving process are detailed in Algorithm 1.

### B. Online Algorithm

In online implementations, future orders and drivers are not fully known, especially ones far off in time. Thus, online vehicle dispatch is more challenging than the offline counterpart

### Algorithm 1 Offline VDP

**Require:** The *FULL* set of orders  $\mathcal{O}$  and drivers  $\mathcal{D}$

**Ensure:** Vehicle dispatching strategy  $\mathcal{S}$ .

- 1: Add all driver and order nodes to shareability network  $\mathcal{G}$
- 2: **for** each driver/order pair  $(d_i, o_j)$  in sets **do**
- 3:   **if**  $t_i^d + t_{ij} \leq t_j^o$  **then**
- 4:     Add edge  $(i, j)$
- 5:   **end if**
- 6: **end for**
- 7: **for** each order/order pair  $(o_i, o_j)$  in sets **do**
- 8:   **if**  $t_i^o + t_{ij} \leq t_j^o$  **then**
- 9:     Add edge  $(i, j)$
- 10:   **end if**
- 11: **end for**
- 12: Solve the maximum cost problem in the network  $\mathcal{G}$
- 13: Recover driver order matching  $\mathcal{S}$  from network flow
- 14: Return  $\mathcal{S}$

due to limited information and forward-looking ability. In this section we leverage an RHC framework [41] for online VDP.

Fig. 2 illustrates the RHC framework of the online algo-

rithm<sup>3</sup>. In RHC, the VDP problem is solved iteratively in a short optimization time window,  $t_o$ <sup>4</sup>. In each step, a vehicle is dispatched for a locked time window  $t_l$ , after which the time horizon rolls by  $t_r$ <sup>5</sup>.



Fig. 2. Illustration of online receding horizon control algorithm structure.

Due to the limited forward-looking horizon, we ignore the future order connectivity in the shareability network. This leads to a greedy suboptimal solution for the online algorithm [42].

### C. Sink Proximity

To address the suboptimality in the online RHC algorithm, we propose a new network science methodology that overcomes the shortsightedness of the RHC framework, *sink proximity*.

The goal of this methodology is to create an attribute for each node (order) of the network that captures future connectivity of the network from this node while being easy and efficient to compute. In other words, this attribute captures the importance of each order from a network-wide perspective and can be leveraged in the online VDP to compensate for the lack of future information. We introduce the formal definition of *sink proximity*.

**Definition 3.** Given a shareability network  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $\mathcal{V} := \mathcal{O} \cup t$ , and the set of all the paths from node  $v_i$  to  $t$ ,  $P(v_i, t)$ , Sink Proximity of node  $v_i$  is defined as

$$SP_{v_i} = \max P(v_i, t). \quad (11)$$

In a shareability network, the sink proximity of an order is the maximum number of orders that a driver can be matched with after serving this order. Fig. 3 provides a schematic representation of this concept, with each color denoting the longest route and its corresponding SP value from node A (red), C (purple), E (orange), and the terminal node  $t$  (SP = 0).

In addition, *sink proximity* can be efficiently computed using Proposition 1.

**Proposition 1.** Finding the maximum number of orders a driver can take in a shareability network can be expressed as a single-source DAG longest path problem.

*Proof.* Given a shareability network  $\mathcal{G}$ , each edge represents the connectivity between a driver and an order or between orders. A driver can start from any node and take orders

<sup>3</sup>Due to page limitations, algorithmic implementations are in the appendix.

<sup>4</sup>The order information in  $t_o$  is known by scheduling or by forecasting.

<sup>5</sup>In the setting of this paper,  $t_o \geq t_l \geq t_r$  always holds. This is to avoid the “dead zone” at the beginning of each optimization time window, where experience delays as vehicles are dispatched to each request’s start location

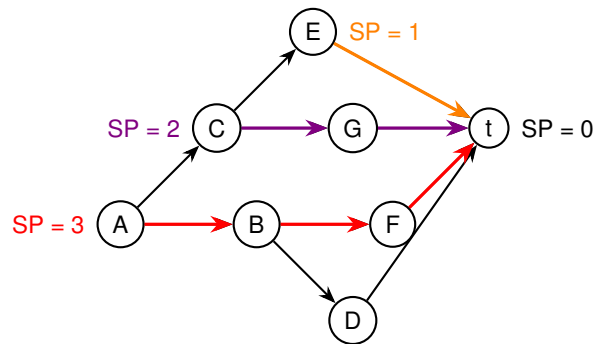


Fig. 3. Schematic representation of a simplified shareability network with the SP attribute of Node A, C, E,  $t$ . The color denotes the longest route and its corresponding SP value.

along any edge, as long as it has not been used before. The driver’s objective is to maximize the total number of orders taken in the future, or order-to-go. This problem is equivalent to a single source DAG longest path problem. Note that any feasible solution for the driver corresponds to a path in  $\mathcal{G}$ , and vice versa. Moreover, the total number of orders taken by the driver is equal to the length of the corresponding path. Therefore, maximizing the number of orders is equivalent to maximizing the length of the path. As  $\mathcal{G}$  is a shareability network, the acyclic nature of time guarantees that it has no cycles [28]. This means that it is possible to solve the problem in linear time by processing the vertices in topological order, usually, with a complexity of  $\mathcal{O}(|\mathcal{V}| + |\mathcal{E}|)$ .  $\square$

Computing sink proximity of a node requires full knowledge of the network. However, it is possible to efficiently estimate this attribute using a data-driven approach as shown in Appendix B.

### D. Sink Proximity-Aided Algorithm

In this section, we demonstrate how to leverage *sink proximity* to improve the quality of the solution obtained by Algorithm 3. The attribute *sink proximity* of a node indicates the number of additional nodes that can be visited after that particular node. It is reasonable to assume that the nodes with higher *sink proximity* should have higher priority. Thus, in the context of this work, given the *sink proximity* value of an order, we assign its value to the weight of the internal link of that order. The detailed algorithm that leverages sink proximity in an RHC framework is presented in Algorithm 2, called RHC-SP. Compared to Algorithm 3, the only difference lies in the additional “for loop” in lines 4 to 7, in which the sink proximity attribute of each node is used to modify the weight of the internal links for order nodes (illustrated in Fig. 1 (c)). Note that in the extended shareability network used as input in Algorithm 2, still only the internal links can have non-zero weights because the successful fulfillment of an order is only represented by one unit of flow in the internal link.

## IV. NUMERICAL STUDY

In this section we conduct a case study of Manhattan, NYC, USA. For the orders we use the NYC Taxi and Limousine

**Algorithm 2** Online VDP with RHC-SP

**Require:** A set of orders  $\mathcal{O}$  and a set of drivers  $\mathcal{D}$  in an optimization time window  $t_o$ . A rolling time window  $t_r$ , and a locked time window  $t_l$  where decisions are locked.

**Ensure:** Vehicle dispatching strategy  $\mathcal{S}$ .

- 1: Initialize  $t \leftarrow t_{\text{start}}$
- 2: **while** the system is running **do**
- 3:   Get order set  $\mathcal{O}_t$  and driver set  $\mathcal{D}_t$  in  $[t, t + t_o]$
- 4:   **for** each order node  $o$  set  $\mathcal{O}$  **do**
- 5:     Estimate sink proximity  $\text{SP}_o$
- 6:     Assigned  $\text{SP}_o$  as the weight to interal edge of  $o$
- 7:   **end for**
- 8:   Construct shareability network  $\mathcal{G}_t$
- 9:   Solve the maximum cost problem in the network  $\mathcal{G}_t$
- 10:   Recover driver order matching  $\mathcal{S}_t$  from network flow
- 11:   Apply actions for matching in  $[t, t + t_l]$  and add to  $\mathcal{S}$
- 12:   Update order set  $\mathcal{O}_t$
- 13:   Update driver set  $\mathcal{D}_t$
- 14:   Update  $t = t + t_r$
- 15: **end while**
- 16: Return  $\mathcal{S}$

Commission (TLC) trip record data [43]. The dataset contains the start time, start location (as an area code), end time, and end location (as an area code) of each taxi order. The city is divided into 260 non-overlapping areas during data processing. Thereafter, the peak hours of two mornings (8:00 to 10:00 am) on June 1 and 2, 2022, are selected. There are 11,483 and 11,825 requests in these two days, respectively. The code is available at: <https://github.com/TIVV424/SPMoD>.

Table I summarizes the parameters used in the numerical experiment. The algorithm's performance is evaluated based on the RSR and running time.

TABLE I  
PARAMETER SETTINGS IN NUMERICAL EXPERIMENT

Parameter	Note	Value
$t_o$	Optimization time window	10, 20, 30, 60 min
$t_r$	Rolling time window	5, 10, 20, 30 min
$t_l$	Locked time window	8, 10, 15, 20, 30 min
$n_D$	Number of drivers	2000 <sup>6</sup>
$t_{\text{void}}$	Driver idle time	30 min

A. Analyzing Algorithm Performance

This section showcases the performance of our proposed approach. First, it is crucial to forecast the values of *sink proximity* (see Appendix B) for each node, and modify the weights of the internal links of the orders, as described in Section III-D. For a given simulation, the optimal objective of the maximum cost network flow problem, from which it is possible to retrieve the optimal objective of the VDP problem, i.e., the RSR, is given by the offline approach, Algorithm 1, which yields global optimality.

Fig. 4 compares two algorithms, RHC with and without *sink proximity*. We also conduct a sensitivity analysis of several

parameters, namely the optimization time window  $t_o$ , the locked time window  $t_l$ , and the rolling time window  $t_r$ . Fig. 4 demonstrates that RHC-SP algorithm consistently outperforms standard RHC in terms of RSR. It is important to note that the complexity for both algorithms with the same parameter settings is the same, and it takes approximately comparable time to compute. This is because the original structure of Problem II-B remains unchanged, regardless of the usage of *sink proximity*. Thus, RHC-SP permits the selection of more aggressive parameters, such as a shorter optimization interval. This is thanks to a more efficient computation, which is vital for ride-hailing platforms that have to operate online.

The results indicate that the proposed RHC-SP algorithm reduces the run time by 4.55%<sup>7</sup>, and increases the RSR by 1.95% on average across all parameter settings tested.

TABLE II  
OBJECTIVE VALUE IMPROVEMENT OF RHC-SP COMPARED TO RHC, WITH DIFFERENT PARAMETER SETTINGS.

$t_o$	$t_r$	$t_l$	RSR improvement
10	5	8	7.41 %
10	5	10	6.98 %
20	5	8	3.35 %
20	5	10	4.00 %
20	10	15	6.80 %
20	10	20	6.47 %
30	5	8	0.62 %
30	5	10	0.94 %
30	10	15	1.76 %
30	10	20	3.13 %
30	20	30	5.75 %
60	5	8	0.15 %
60	10	30	0.17 %
60	10	40	0.67 %
60	20	30	0.12 %
60	30	40	0.18 %

Table II illustrates the performance comparison between the RHC-SP and RHC Algorithms in terms of RSR improvement. The RHC-SP algorithm consistently outperforms the RHC algorithm, especially when  $t_o$  is small. To enhance clarity, the magnitude of improvement is visually represented using color coding: green indicates large gains, while red indicates small ones.

To gain deeper insights into the results, We examine the rolling RSR through the time horizon, as shown in Fig. 5, which depicts the rolling ratio of the RSR, up to that point in time, i.e., at the end of each rolling iteration, between the two algorithms. When the lines are above the gray dashed line (ratio of 1), it indicates that RHC-SP outperforms the RHC algorithm up to that point in time. Among all parameter settings, RHC-SP with the  $t_o = 20, t_r = 10, t_l = 20$  parameter combination provides the most performance improvement w.r.t. the standard RHC framework.

We highlight that, RHC-SP, at the end of the simulation, always outperforms the standard RHC in terms of RSR. The benefit of RHC-SP is generally more significant when the optimization time window is smaller and the rolling/locked windows are larger, i.e., when future information is limited.

<sup>7</sup>The reported number demonstrates that the algorithm does not compromise computational efficiency, but it does not indicate an increase in computational efficiency.

<sup>6</sup>The driver-to-order ratio is approximately 0.2

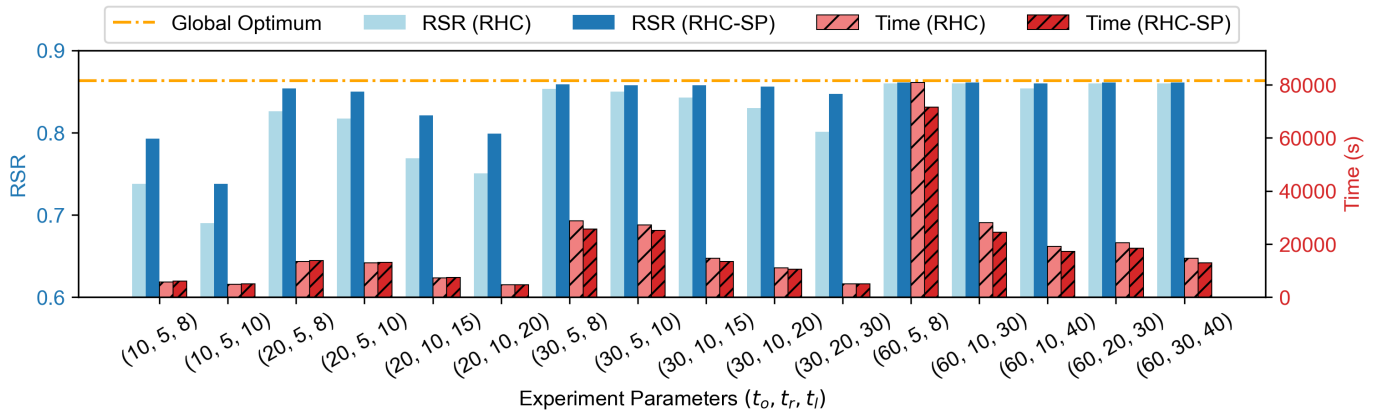


Fig. 4. Comparison of RSR (left axis), i.e., objective of Problem 1, and computation time (right axis) for online algorithms with different parameter settings. The global optimal solution is computed by solving the same problem with the offline algorithm.

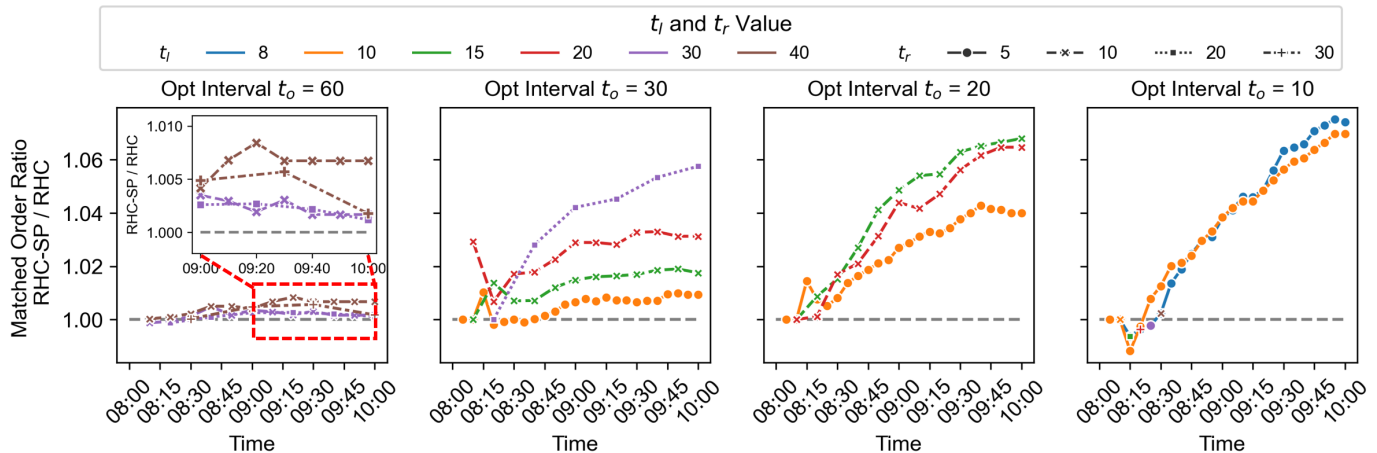


Fig. 5. Comparison of rolling request service rate (Problem 1 objective, RSR) using RHC-SP and RHC across parameter settings. Ratios above 1 indicate that the proposed RHC-SP algorithm outperforms RHC.

This indicates that the algorithm is especially valuable when computational power is restricted and a larger rolling time window and locked time window are necessary for quick dispatch and decision-making. Interestingly, we highlight that in Fig. 5, the cumulative RSR with parameter  $t_o = 10$  does not constantly outperform RHC during the whole simulation time, even if at the end it does. This result confirms that the sink proximity attribute allows the RHC algorithm to not be myopic because of the horizon limit, allowing to overall outperform, even if, in the short-term, it might underperform because of the additional future information. From a physical point of view, the result can be interpreted as a preventive realignment to meet future orders by sacrificing current ones. To summarize, sink proximity complements the missing information on the network's future structure and evolution when solving an online problem with RHC.

## B. Discussions

RL has recently been applied to various combinatorial problems, including the VDP. Although this work is orthogonal to RL, it is worth exploring how the proposed approach could be integrated into an RL framework. Notably, the authors in [44], [45] show that, in a similar setting compared to ours, RL outperforms in terms of computation time with respect to an RHC framework but it under performs in terms of overall profit

generated by the fleet operator. The RHC framework, in turn, is sub-optimal w.r.t. the “oracle” case, i.e., the equivalent of our offline approach.

In online VDP, the most significant challenge arises from uncertainty regarding future demand. Rather than attempting to predict the precise timing and location of future orders as in many CO works, the concept of sink proximity allows us to estimate aggregated network information. The latter is a simpler task that enhances the reliability of the predicted data. The concept of sink proximity also resembles value functions in RL. While the value function captures the expected future reward of a given state, sink proximity captures the projected value of a given order node. In future work, it would be interesting to investigate if our proposed approach can be integrated in an RL framework, and to what extent it would be beneficial to the quality of the solution obtained.

## V. CONCLUSIONS

This paper examines how to dispatch vehicles more effectively in ride-hailing platforms to ease traffic flow, elevate service quality, and strengthen the resilience of urban mobility systems. We introduce a novel approach to improve the request service rate on ride-hailing platforms by incorporating network science measures into the shareability network. Sink proximity, a measure for the longest path distance to the sink node within

a network flow context, differs from existing measures by focusing on the global behavior of the network rather than local features of individual nodes or regions. By converting the problem into its equivalent form of finding the longest path in a DAG, we show that sink proximity can be efficiently calculated and predicted from order node information in a shareability network.

Building upon this, we propose a new online dispatch algorithm, RHC-SP, which assigns edge weights to the nodes based on their sink proximity. This allows the algorithm to incorporate network expansion information, even in a limited-time, partial network. By integrating this into the online optimization process, the proposed algorithm improves performance, increasing the number of matched orders by up to 7% compared to a standard RHC framework, without additional computational overhead. The inclusion of sink proximity as a weight in the dispatch problem resembles the terminal constraints used in RHC problems, addressing the suboptimality that can arise when the optimization horizon is limited.

Future work could explore scenarios with more volatile request distributions or investigate a sliding sink node approach to simulate infinite-horizon problems. Additionally, analyzing sink proximity in other dynamic network applications holds promising potential for further research. While a direct numerical comparison with other VDP algorithms is not possible due to differences in problem assumptions and modeling frameworks, another promising direction for future work is to incorporate SP into other online algorithms and evaluate its impact on performance.

## APPENDIX A ONLINE VDP

We present Algorithm 3 as the online version of Algorithm 1 using RHC techniques. In this algorithm, we detail how to iteratively construct the network and simplify the representation of the network construction process. Steps 1 to 11 in Algorithm 1 are reduced to Step 4 in Algorithm 3.

---

### Algorithm 3 Online VDP with RHC

---

**Require:** A set of orders  $\mathcal{O}$  and a set of drivers  $\mathcal{D}$  in an optimization time window  $t_o$ . A rolling time window  $t_r$ , and a locked time window  $t_l$  where decisions are locked.

**Ensure:** Vehicle dispatching strategy  $\mathcal{S}$ .

- 1: Initialize  $t \leftarrow t_{\text{start}}$
  - 2: **while** the system is running **do**
  - 3:   Get order set  $\mathcal{O}_t$  and driver set  $\mathcal{D}_t$  in  $[t, t + t_o]$
  - 4:   Construct shareability network  $\mathcal{G}_t$
  - 5:   Solve the maximum cost problem in the network  $\mathcal{G}_t$
  - 6:   Recover driver order matching  $\mathcal{S}_t$  from network flow
  - 7:   Apply actions for matching in  $[t, t + t_l]$  and add to  $\mathcal{S}$
  - 8:   Update order set  $\mathcal{O}_t$
  - 9:   Update driver set  $\mathcal{D}_t$
  - 10:   Update  $t = t + t_r$
  - 11: **end while**
  - 12: Return  $\mathcal{S}$
- 

## APPENDIX B SINK PROXIMITY FORECASTING

In online dispatch with receding horizon, the network structure remains unknown after the optimization window. Thus, forecasting is necessary to estimate the sink proximity value over a period longer than the time window. We leverage data-driven prediction methods to estimate the value of sink proximity, where the inputs include order start area, start time, end area, and end time or the orders.

### A. Time-standardized Sink Proximity

The sink proximity metric measures the step-wise reachability of nodes to the sink in a network with a longer horizon and captures information about how the shareability network expands. To illustrate this concept, in the context of the shareability network, an order node with a higher degree of sink proximity is more probable to be an order in a demand-heavy area during peak hours.

If the end time of a node is closer to the end time of the network construction horizon, the sink proximity will naturally be smaller. Thus, to mitigate this phenomenon, the time-standardized sink proximity (TSSP) is defined as follows:

**Definition 4.** For each node  $v_i$  in a shareability network  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , the time-standardized sink proximity is computed as the sink proximity divided by the time it takes to reach the sink node, with a buffer. Thus it is defined as

$$\text{TSSP}_{v_i} = \frac{\text{SP}_{v_i}}{t_e + t_{\text{buffer}}}. \quad (12)$$

These measurements quantitatively represent the importance of each order node in the network. Note that if a sink node has a time attribute  $t$ , and the time it takes to reach the sink node is a small time period  $t_e$ , most of the orders with an end time  $t_i^d \geq t - t_e$  would have a very small sink proximity value. Thus, computing TSSP allows for eliminating the impact of the sink node time attribute, facilitating the prediction of the network metric.

### B. Sink Proximity Forecasting Model

We use  $\epsilon$ -SVR [46] with radial basis function kernel to predict the value of TSSP, and tune the hyperparameters<sup>8</sup>, shown in Table III, using grid search and cross-validation. Then, we reverse the time standardization and round the value to recover the sink proximity for each order, which are integers by definition.

TABLE III  
GRID SEARCH RANGE AND BEST PARAMETERS

Parameter	Grid search ranges	Best value
kernel	[poly, rbf, sigmoid]	rbf
L2 regularization parameter	[100, 200, 300, 400, 500]	300
gamma	[0.001, 0.01, 0.1]	0.01
epsilon	[0.01, 0.05, 0.1, 0.2, 0.5]	0.1

<sup>8</sup>See "<https://scikit-learn.org/1.5/modules/generated/sklearn.svm.SVR.html>" for parameter explanations.

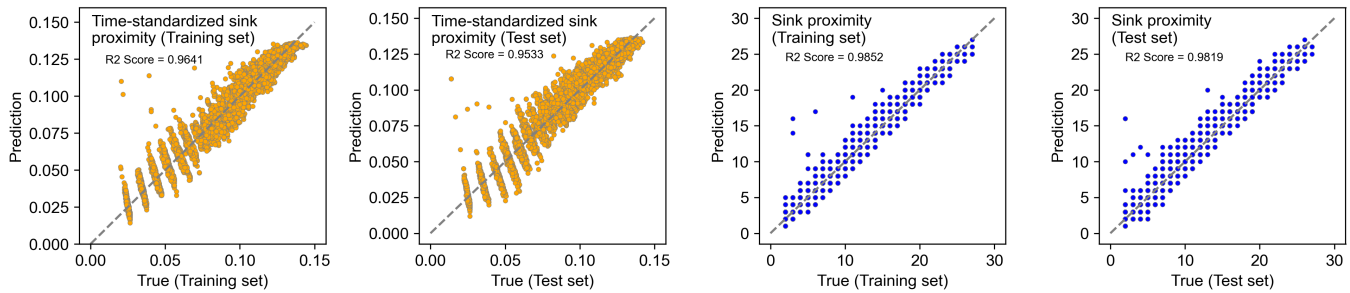


Fig. 6. Prediction accuracy of TSSP and SP in both training and testing sets for the case study of Manhattan, NYC. Data used for training is the morning peak hour of June 1st, 2022. Data used for validation is the morning peak hour of June 2nd, 2022.

### C. Sink Proximity Forecasting Results

To train the model to forecast sink proximity, we leverage data of the morning peak hour dataset of June 1st, 2022, and test on the same period on the following day. The buffer time for TSSP calculation is selected to be 20 minutes. Fig. 6 illustrates the prediction accuracy of sink proximity in training and testing dataset. We highlight that for an order node that is closer to the sink in time with a low sink proximity, the prediction accuracy is also lower. This is due to the relative magnitude of the value, as shown in the lower-left corner of Fig. 6, where the strip pattern is displayed. However, this lower accuracy can be improved by reversing the standardization to recover the integral sink proximity. After rounding, the prediction accuracy increased. Using the SVR model, the predicted TSSP achieves an  $R^2$  score of 0.95, while the recovered SP after rounding reaches an  $R^2$  score of 0.98. Using the SVR model, the predicted sink proximity achieves an  $R^2$  score of 0.985 in the training set, and 0.982 in the testing set.

### REFERENCES

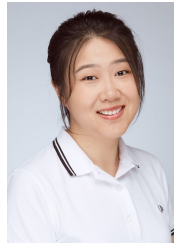
- [1] S. Banerjee and R. Johari, "Ride Sharing," in *Sharing Economy: Making Supply Meet Demand*, M. Hu, Ed. Cham: Springer International Publishing, 2019, pp. 73–97.
- [2] "Ride Sharing Market Size, Share, Growth | Industry Report, 2032," <https://www.fortunebusinessinsights.com/ride-sharing-market-103336>.
- [3] Theverge.com, "Uber ends the year in the black for the first time ever," Tech. Rep., 2024, <https://www.theverge.com/2024/2/8/24065999/uber-earnings-profitable-year-net-income>.
- [4] G. Qin, Q. Luo, Y. Yin, J. Sun, and J. Ye, "Optimizing matching time intervals for ride-hailing services using reinforcement learning," *Transportation Research Part C: Emerging Technologies*, vol. 129, p. 103239, 2021.
- [5] J. C. Castillo, D. Knoepfle, and E. G. Weyl, "Matching and pricing in ride hailing: Wild goose chases and how to solve them," *Management Science*, 2024.
- [6] F. Marzbani and A. Abdelfatah, "Economic dispatch optimization strategies and problem formulation: A comprehensive review," *Energies*, vol. 17, no. 3, 2024.
- [7] F. Rossi, R. Iglesias, M. Alizadeh, and M. Pavone, "On the interaction between Autonomous Mobility-on-Demand systems and the power network: Models and coordination algorithms," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 1, pp. 384–397, 2020.
- [8] S. Wollenstein-Betech, M. Salazar, A. Houshmand, M. Pavone, C. G. Cassandras, and I. C. Paschalidis, "Routing and rebalancing intermodal autonomous mobility-on-demand systems in mixed traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 12 263–12 275, 2021.
- [9] M. Salazar, N. Lanzetti, F. Rossi, M. Schiffer, and M. Pavone, "Intermodal autonomous mobility-on-demand," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3946–3960, 2020.

- [10] Z. Jin, Y. Li, D. Gruyer, and M. Tu, "Enhancing the Carbon Reduction Potential in Ridesplitting through Evolutionary Game Strategies of Tripartite Stakeholders under Carbon-Inclusive Policy," *Energies*, vol. 17, no. 16, p. 4103, Jan. 2024.
- [11] W. Li, T. Yu, Y. Zhang, and X. M. Chen, "A shared ride matching approach to low-carbon and electrified ridesplitting," *Journal of Cleaner Production*, vol. 467, p. 143031, Aug. 2024.
- [12] K. Wei, V. Vaze, and A. Jacquillat, "Transit Planning Optimization Under Ride-Hailing Competition and Traffic Congestion," *Transportation Science*, vol. 56, no. 3, pp. 725–749, May 2022.
- [13] J. Ke, H. Yang, and Z. Zheng, "On ride-pooling and traffic congestion," *Transportation Research Part B: Methodological*, vol. 142, pp. 213–231, Dec. 2020.
- [14] L. Zhang, T. Hu, Y. Min, G. Wu, J. Zhang, P. Feng, P. Gong, and J. Ye, "A taxi order dispatch model based on combinatorial optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: Association for Computing Machinery, Aug. 2017, pp. 2151–2159.
- [15] M. Lowalekar, P. Varakantham, and P. Jaillet, "Online spatio-temporal matching in stochastic and dynamic domains," *Artificial Intelligence*, vol. 261, pp. 71–112, 2018.
- [16] Y. Xu, W. Wang, G. Xiong, X. Liu, W. Wu, and K. Liu, "Network-flow-based efficient vehicle dispatch for city-scale ride-hailing systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5526–5538, 2021.
- [17] S. S. Eshkevari, X. Tang, Z. Qin, J. Mei, C. Zhang, Q. Meng, and J. Xu, "Reinforcement Learning in the Wild: Scalable RL Dispatching Algorithm Deployed in Ridehailing Marketplace," Feb. 2022.
- [18] Z. Liu, J. Li, and K. Wu, "Context-aware taxi dispatching at city-scale using deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 1996–2009, 2022.
- [19] Y. Tong, D. Shi, Y. Xu, W. Lv, Z. Qin, and X. Tang, "Combinatorial optimization meets reinforcement learning: Effective taxi order dispatching at large-scale," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 10, pp. 9812–9823, Oct. 2023.
- [20] X. Tang, F. Zhang, Z. Qin, Y. Wang, D. Shi, B. Song, Y. Tong, H. Zhu, and J. Ye, "Value Function is All You Need: A Unified Learning Framework for Ride Hailing Platforms," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD '21. New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp. 3605–3615.
- [21] E. Liang, K. Wen, W. H. K. Lam, A. Sumalee, and R. Zhong, "An Integrated Reinforcement Learning and Centralized Programming Approach for Online Taxi Dispatching," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 9, pp. 4742–4756, Sep. 2022.
- [22] D. Wen, Y. Li, and F. C. M. Lau, "A Survey of Machine Learning-Based Ride-Hailing Planning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 6, pp. 4734–4753, 2024.
- [23] Y. Liu, R. Jia, J. Ye, and X. Qu, "How machine learning informs ride-hailing services: A survey," *Communications in Transportation Research*, vol. 2, p. 100075, Dec. 2022.
- [24] L. R. Ford and D. R. Fulkerson, "Maximal Flow Through a Network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, Jan. 1956.
- [25] J. Orlin, "A polynomial time primal network simplex algorithm for minimum cost flows," *Math. Prog.*, vol. 78, pp. 109–129, Jan. 1996.
- [26] R. E. Tarjan, "Dynamic trees as search trees via euler tours, applied to the network simplex algorithm," *Mathematical Programming*, vol. 78, no. 2, pp. 169–177, Aug. 1997.

- [27] X. Zhan, X. Qian, and S. V. Ukkusuri, "A graph-based approach to measuring the efficiency of an urban taxi service system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 9, pp. 2479–2489, Sep. 2016.
- [28] 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.
- [29] J. Wu, C. Lu, C. Wu, Y. Qin, Q. Li, N. Ma, and J. Fang, "Mobility data-driven complete dispatch framework for the ride-hailing platform," in *Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers*, 2021, pp. 684–690.
- [30] C. Yan, H. Zhu, N. Korolko, and D. Woodard, "Dynamic pricing and matching in ride-hailing platforms," *Naval Research Logistics (NRL)*, vol. 67, no. 8, pp. 705–724, Dec. 2020.
- [31] D. Shi, Y. Tong, Z. Zhou, B. Song, W. Lv, and Q. Yang, "Learning to Assign: Towards Fair Task Assignment in Large-Scale Ride Hailing," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD '21. New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp. 3549–3557.
- [32] L. C. Freeman, "A Set of Measures of Centrality Based on Betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977.
- [33] A. Bavelas, "Communication Patterns in Task-Oriented Groups," *The Journal of the Acoustical Society of America*, vol. 22, no. 6, pp. 725–730, Nov. 1950.
- [34] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, Mar. 1953.
- [35] B. ERNO, B. BUECHEL, and V. VINCENT, "The Dynamics of Closeness and Betweenness," *The Journal of Mathematical Sociology*, vol. 37, no. 3, pp. 159–191, Jul. 2013.
- [36] M. Elmezain, E. A. Othman, and H. M. Ibrahim, "Temporal Degree-Degree and Closeness-Closeness: A New Centrality Metrics for Social Network Analysis," *Mathematics*, vol. 9, no. 22, p. 2850, Jan. 2021.
- [37] F. Paparella, T. Hofman, and M. Salazar, "Electric autonomous mobility-on-demand: Jointly optimal vehicle design and fleet operation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 11, pp. 17 054–17 065, 2024.
- [38] D. Bertsimas, P. Jaillet, and S. Martin, "Online vehicle routing: The edge of optimization in large-scale applications," *Operations Research*, vol. 67, no. 1, pp. 143–162, 2019.
- [39] W. H. Cunningham, "A network simplex method," *Mathematical Programming*, vol. 11, no. 1, pp. 105–116, 1976.
- [40] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.
- [41] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.
- [42] P. Scokaert, D. Mayne, and J. Rawlings, "Suboptimal model predictive control (feasibility implies stability)," *IEEE Transactions on Automatic Control*, vol. 44, no. 3, pp. 648–654, 1999.
- [43] N. Taxi and L. Commission. (2022) NYC Taxi and Limousine Commission Trip Record Data. [Online]. Available: <https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>
- [44] C. Schmidt, D. Gammelli, F. Camara Pereira, and F. Rodrigues, "Learning to control autonomous fleets from observation via offline reinforcement learning," in *European Control Conference*, 2024.
- [45] A. Singhal, D. Gammelli, J. Luke, K. Gopalakrishnan, D. Helmreich, and M. Pavone, "Real-time control of electric autonomous mobility-on-demand systems via graph reinforcement learning," in *European Control Conference (ECC)*, 2024, pp. 1407–1414.
- [46] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, Aug. 2004.



**Ruiling Wang** is a Postdoctoral Fellow at KTH Royal Institute of Technology. She received her Ph.D. degree in Systems Engineering from the University of California, Berkeley in 2025, and got her B.S. degree in Building Energy Engineering from Tsinghua University in 2020. Her research combines ideas from optimization, control, economics, and learning to develop algorithms for socio-technical decision-making problems in future energy-mobility systems, from urban energy infrastructure, to intelligent transportation networks, to EV markets.



**Jiaman Wu** is an Assistant Professor in Data Science at City University of Hong Kong. She got her Ph.D. degree in Systems Engineering in the Department of Civil and Environmental Engineering, UC Berkeley. Before that, she received her bachelor's degree in geographical monitoring and census from the School of Remote Sensing and Information Engineering, Wuhan University, and her master's degree in computer science and technology from the Institute for Interdisciplinary Information Sciences, Tsinghua University. Her research interests include operation and planning for the power system and transportation system.



**Fabio Paparella** obtained his Ph.D. from the Control Systems Technology (CST) section at Eindhoven University of Technology, the Netherlands. He studied mechanical engineering at Politecnico di Milano, Italy, where he received his Bachelor's degree in 2017 and his Master's cum laude in 2020 with a thesis in collaboration with NASA Jet Propulsion Laboratory, California, USA. His research interests include mobility-on-demand, smart mobility, and optimization. In 2024 he was a visiting scholar at University of California, Berkeley.



**Scott J. Moura** is a Full Professor in Civil & Environmental Engineering and Director of the Energy, Controls, & Applications Lab (eCAL) at the University of California, Berkeley. He received a B.S. degree from the University of California, Berkeley, CA, USA, and the M.S. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 2006, 2008, and 2011, respectively, all in mechanical engineering. From 2011 to 2013, he was a Post-Doctoral Fellow at the Cymer Center for Control Systems and Dynamics, University of California,

San Diego. In 2013, he was a Visiting Researcher at the Centre Automatique et Systèmes, MINES ParisTech, Paris, France. His research interests include control, optimization, and machine learning for batteries, electrified vehicles, and distributed energy resources.

Dr. Moura is a recipient of the National Science Foundation (NSF) CAREER Award, Carol D. Soc Distinguished Graduate Student Mentor Award, the Hellman Fellowship, the O. Hugo Shuck Best Paper Award, the ACC Best Student Paper Award (as advisor), the ACC and ASME Dynamic Systems and Control Conference Best Student Paper Finalist (as student and advisor), the UC Presidential Postdoctoral Fellowship, the NSF Graduate Research Fellowship, the University of Michigan Distinguished ProQuest Dissertation Honorable Mention, the University of Michigan Rackham Merit Fellowship, and the College of Engineering Distinguished Leadership Award.



**Marta C. Gonzalez** is a Full Professor of City and Regional Planning, Civil and Environmental Engineering at the University of California, Berkeley, and a Physics Research faculty in the Energy Technology Area (ETA) at the Lawrence Berkeley National Laboratory (Berkeley Lab). Prior to joining Berkeley, Dr. Gonzalez worked as an Associate Professor of Civil and Environmental Engineering at MIT, a member of the Operations Research Center and the Center for Advanced Urbanism. She is a member of the scientific council of technology companies such as Gran Data, PTV and the Pecan Street Project consortium.

Dr. Gonzalez received the Licenciatura degree in physics from the Universidad Simon Bolivar, in 1999, the Magister Sc. degree in physics from Central University of Venezuela, in 2001, and the PhD (Dr. rer. nat) degree in physics from Stuttgart University at in 2006. She has more than 60 publications, and her work has been published in the *Nature*, the *Science*, the *Nature Physics*, the *Physics A*, the *Journal of the Royal Society Interface*, the *Physical Review Letters*, the *Scientific Reports*, the *Transportation Research Part C: Emerging Technologies*, the *Data Mining and Knowledge Discovery*, among many others.