An Integrated Decomposition and Approximate Dynamic Programming Approach for On-demand Ride Pooling

Xian Yu* and Siqian Shen*

Abstract—**Through smartphone apps, drivers and passengers can dynamically enter and leave ride-hailing platforms. As a result, ride-pooling is challenging due to complex system dynamics and different objectives of multiple stakeholders. In this paper, we study ride-pooling with no more than two passenger groups who can share rides in the same vehicle. We dynamically match available drivers to randomly arriving passengers and also decide pick-up and drop-off routes. The goal is to minimize a weighted sum of passengers' waiting time and trip delay time. A spatial-and-temporal decomposition heuristic is applied and each subproblem is solved using Approximate Dynamic Programming (ADP), for which we show properties of the approximated value function at each stage. Our model is benchmarked with the one that optimizes vehicle dispatch without ride-pooling and the one that matches current drivers and passengers without demand forecasting. Using test instances generated based on the New York City taxi data during one peak hour, we conduct computational studies and sensitivity analysis to show (i) empirical convergence of ADP, (ii) benefit of ride-pooling, and (iii) value of future supply-demand information.**

Index Terms—**Mobility on Demand (MoD), supply-demand uncertainty, ride-pooling, spatial-temporal decomposition, approximate dynamic programming**

I. Introduction

Increasing population and environmental issues have led to various shared-mobility forms, including carsharing and ride-hailing services whose demand drastically increased in the past decade (see [1]). Ref. [2] thoroughly reviewed the literature related to ride-hailing including vehicle dispatching, scheduling, routing, and solution methods mainly based on optimization and heuristics. In this paper, we study an on-demand ride-pooling problem over a finite time horizon, and the decisions include matching drivers with passengers, as well as finding optimal routes for drivers to pick up and drop off matched passengers. We focus on the case where no more than two groups of passengers may share rides at the same time in one vehicle, and prove value function properties of the dynamic problem characterized by an ADP approach.

To show the benefit and feasibility of pooling rides in practice, [3] considered the minimum fleet problem and conducted large-scale simulation to show that by pooling rides, the taxi fleet size in the Manhattan area of the New York City can be reduced by 40% without delaying existing trips. Ref. [4] studied ride-pooling effects under different pricing schemes and demonstrated the importance of pooling rides for revenue maximization. Reported in [5], UberPOOL has saved \$4.5 million worth in fuel costs in India since launch. Large-scale operations of ride-pooling were considered in [6]–[10] with the deployment of simulation, parallel computing, machine learning, stochastic control, and combinatorial optimization techniques. Their studies demonstrated the possibility and effectiveness of implementing on-demand ride-sharing at scale.

In this paper, we model the ride pooling problem as a multistage stochastic program and incorporate different objectives of individual passenger groups and drivers, measured based on waiting time and trip delay. If directly using dynamic programming, the number of states will grow exponentially as the number of drivers and passengers grows, resulting in the "curse of dimensionality" issue. We employ ADP [11] for the tractability of modeling system dynamics. While dynamic programming calculates value functions exactly, ADP exploits an approximation of the value function at each stage [12]. Furthermore, by examining spatial-temporal structures of supply-demand data, we propose a heuristic decomposition scheme and only apply ADP to each subproblem to achieve better computational performance. Our goal is to show the convergence of ADP, benefit of pooling rides, and value of supply-demand information via testing real-world instances using different parameters.

A. Literature Review

Ride-hailing operations are closely related to several fundamental mathematical problems including matching, vehicle routing, service scheduling, and queueing networks. Different from the traditional static settings in these problems where demands and supplies are pre-given, the drivers (supply) and passengers (demand) randomly arrive in on-demand ride-hailing systems, urging the use of stochastic and dynamic methods to ensure solution quality. Ref. [13] deployed a rolling horizon approach and matched drivers with random passenger arrivals in each horizon without optimizing detailed routes. Refs. [14] and [15] considered algorithms for online matching and the latter specifically focused on the ride-sharing application. Ref. [16] utilized queueing networks to develop a continuous linear program that accounted for the time-varying property of arrival rates of passengers and drivers in different locations. In addition to matching and routing, pricing and vehicle rebalancing are two other important issues that can significantly affect operational efficiency and ride-hailing

*Xian Yu and Siqian Shen are with Department of Industrial and Operations Engineering at the University of Michigan in Ann Arbor, USA. yuxian@umich.edu, siqian@umich.edu

revenue, and they were studied in [16]–[22] and in [23]–[28], respectively.

Incorporating ride-pooling into ride-sharing can increase modeling and computational complexity, but its benefit has been justified in the previously mentioned work, such as [3]–[5]. The study of pooling rides among deterministic sets of drivers and passengers dates back to [29], who used spatial-temporal networks for pooling taxi rides. Ref. [30] applied heuristics to develop routing strategies in taxi-pooling. Ref. [31] assumed that all the trips are known in advance, and introduced a "shareability" network and a graph-based model to quantify the benefits of ride pooling.

In practice, mobility-on-demand (MoD) or autonomous mobility-on-demand (AMoD) systems ubiquitously involve information uncertainty. Refs. [8], [10], [23], [26], [28] employed queueing theories, machine learning, and/or stochastic control to predictively position vehicles under random spatial-temporal distributions of demand and supply. Ref. [32] proposed a dynamic request-vehicle assignment heuristic and a rebalancing policy for AMoD systems, and evaluated the value of demand information through simulation. On the other hand, the design and operations of MoD/AMoD systems need to account for diverse objectives of drivers, passengers and system operators, whose benefits may be conflicting. Ref. [31] optimized ride-pooling by considering the maximum number of trips that can be shared and the maximum delay customers can tolerate. Ref. [33] characterized quality of service in ride-sharing through potential trip delay probability. They proposed a predictive positioning method to improve quality of service. Under demand uncertainties, [27], [34] proposed data-driven distributionally robust optimization schemes to ensure service fairness.

The difficulty of involving uncertainty in MoD- or AMoD-related system optimization including ride-pooling optimization, is to design scalable solution approaches. When considering vehicle routing in ride-pooling, the problem is closely related to the Vehicle Routing Problem (VRP) [35], Dial-a-Ride [36] and Pickup and Delivery problems [37], [38], which are NP-hard in general. To implement on-demand ride-pooling strategies, [6], [7], [9], [10], [39] avoided explicitly modeling road networks and routing decisions, and deployed simulation, local search, reinforcement learning, parallel computing to handle real-world large-scale data. Specifically, [40] aimed to dynamically assign passenger requests and exploited hybrid simulated annealing to obtain quick solutions. Ref. [7] presented a heuristic approach for real-time high-capacity ride-hailing that dynamically produced routes given randomly arriving demand requests and uncertain vehicle distribution. Ref. [6] proposed a real-time online simulation and parallelization framework to study taxi ride-pooling at scale. Refs. [41] and [42] developed a taxi-sharing system to handle passengers' real-time requests.

B. *Purposes of the Paper*

This paper models dynamic ride-pooling as a multistage stochastic program and uses ADP ([11], [43]) for solving the problem through a heuristic decomposition scheme. Refs. [44]–[46] are representative literature of applying ADP for solving stochastic VRP and other transportation problems. We incorporate multiple objectives including minimizing the total passengers' waiting time, trips' delay time, and unsatisfied trip requests, to improve quality of service. Note that although the drivers' profit is not specified in the objective function, the passenger-oriented objective can potentially increase drivers' profit when more passengers' requests are satisfied. In the numerical studies, a reward formula is built to calculate and reflect drivers' benefits based on the results of passengers' reward. Moreover, this paper only focuses on how to dynamically match and route drivers and passengers while assuming given fixed prices.

The main objective is to study the structure of value functions in ADP. Specifically, we explore the linkage between value functions in different states and prove the monotonicity result. We also heuristically decompose time horizon and service region into sub-periods and sub-regions to allow more efficient implementation. We conduct numerical studies on instances generated from real-world data, and consider two benchmark approaches: (i) myopic, which solves a deterministic linear program in each stage and implements the solutions in a rolling horizon way and (ii) NotPool, which does not allow ride pooling.

C. *Structure of the Paper*

The remaining of this paper is organized as follows. In Section II, problem settings and a deterministic formulation are introduced. In Section III, we develop the ADP algorithm for on-demand ride-pooling with no more than two passenger groups in any shared ride. In Section IV, a spatial-temporal decomposition heuristic is described for implementing ADP. In Section V, we test a diverse set of instances and present computational results. We conclude the paper and state future research directions in Section VI.

II. Problem Description and Formulations

A. *Assumptions and Notation*

The paper has the following assumptions: (i) the capacity of each vehicle is fixed and pre-given (which we set as 4 seats excluding the driver seat in our later computational studies); (ii) every vehicle can be shared by at most two passenger groups; (iii) at the end of each time stage, passengers with no matches quit the system. Assumptions (i) and (iii) can be justified by practical implementation and customer behavior, respectively. Assumption (ii) is a limitation and it is mainly made for modeling simplicity. In practice, UberPool and Lyft Line Apps can allow more than two passenger groups to share rides at the same time. When the number of ride-sharing groups increases from 2 to 3, combinations of pickup and drop-off sequences increase drastically, while the trip delay time and customer waiting time can be longer. On the other hand, pooling more rides can potentially increase revenue and we will explore the more general case without Assumption (ii) in our future research.

In our problem, for every driver, an attribute vector a corresponds to a state such that:

$$a = (o_a, \mathcal{L}_a, N_a),$$

where o_a denotes the current location of the driver, $\mathcal{L}_a$ is an ordered list containing all the places that the driver will visit, and N_a is the number of passengers that are currently assigned to the driver. Note that $|\mathcal{L}_a|$ is 0 at minimum (when a vehicle is empty) and 3 at maximum (when a vehicle is occupied by a passenger group and needs to pick up another passenger group). Let $\mathcal{A}_t^i$ be the set of all possible driver attribute vectors a, whose $|\mathcal{L}_a| = i$ in stage t, for $i = 0, 1, 2, 3$. We denote $\mathcal{A}_t = \mathcal{A}_t^0 \cup \mathcal{A}_t^1 \cup \mathcal{A}_t^2 \cup \mathcal{A}_t^3$ for each stage $t \in \{1, \ldots, T\}$.

For every passenger request, an attribute vector b also corresponds to a state following:

$$b = (\mathbf{o}_b, \mathbf{d}_b, N_b),$$

where $\mathbf{o}_b$ denotes the origin of passenger(s), $\mathbf{d}_b$ denotes the destination of passenger(s), and N_b denotes the number of passengers in the request. Let $\mathcal{B}_t$ be the set of all possible passenger attribute vectors b in stage t, R_{ta} be the number of drivers with attribute vector a in stage t, and $R_t = (R_{ta})_{a \in \mathcal{A}_t}$ be the resource state vector in stage t. Similarly, let D_{tb} be the number of passengers with attribute b in stage t, and $D_t = (D_{tb})_{b \in \mathcal{B}_t}$ be the demand state vector in stage t.

Define variable x_{tab} as the number of drivers with attribute a assigned to passengers with attribute b in stage t, and variable $x_{ta\emptyset}$ as the number of drivers with attribute a not assigned to any passengers in stage t. Let $d \in \mathcal{D}$ represent a decision index referring to either assigning a driver to pick up a particular group of passengers or instructing that driver to wait at the current position, i.e., $\mathcal{D} = \mathcal{B}_t \cup \{\emptyset\}$. Note that there may be other actions such as vehicle rebalancing and empty-car rerouting, which are not considered in this paper. Variable $\mathbf{x}_t = (x_{tad})_{a \in \mathcal{A}_t, d \in \mathcal{D}}$ denotes the overall decision vector in stage t.

B. Objective Function

Our objective function accounts for passengers' waiting time and trip delay time, described in detail as follows.

1) Passengers' Waiting time: Passengers' waiting time is measured using the distance that the assigned driver moves from his/her current location to the passengers' origin. For every pair of driver with attribute $a = (o_a, \mathcal{L}_a, N_a)$ and passenger group with attribute $b = (\mathbf{o}_b, \mathbf{d}_b, N_b)$, the waiting time is:

$$w_{ab} = \| o_a - \mathbf{o}_b \|, \tag{1}$$

where $o_a, \mathbf{o}_b$ denote the driver's and passenger's current locations, respectively. The norm $\| \cdot \|$ measures the travel distance (or time) from o_a to $\mathbf{o}_b$.

2) Passengers' Delay time: Passengers' delay time would only occur when they start to share rides with other passengers. The distance of detour is used to measure the delay time (i.e., the difference between the original individual travel time and the total travel time if they share rides). For each pair of passenger groups to be dropped off by the same driver, we denote the driver's current location by o_1, the first group of passengers' destination by d_1, the second group of passengers' origin and destination by o_2 and d_2, respectively. Which passenger group to drop off first is decided based on respective shortest paths. Specifically, when $\| d_2 - o_2 \| \leq \| d_1 - o_2 \|$, the driver will first drop off the second group of passengers. In this case, only the first group of passengers is delayed, and the delay time is

$$d_{ab} = \| o_2 - o_1 \| + \| d_2 - o_2 \| + \| d_1 - d_2 \| - \| d_1 - o_1 \| . \tag{2}$$

When $\| d_2 - o_2 \| > \| d_1 - o_2 \|$, the driver will first drop off the first group of passengers. In this case, the total delay time of both groups of passengers is

$$\begin{aligned} d_{ab} &= \| o_2 - o_1 \| + \| d_1 - o_2 \| - \| d_1 - o_1 \| \\ &+ \| d_1 - o_2 \| + \| d_1 - d_2 \| - \| d_2 - o_2 \| . \end{aligned} \tag{3}$$

The above prepossessing procedures are detailed in Algorithm 1.

Algorithm 1 Preprocessing

```
1: For each pair of driver $a = (o_a, \mathcal{L}_a, N_a)$ and passenger $b = (\mathbf{o}_b, \mathbf{d}_b, N_b)$
2: Use Google map API to calculate travel time and waiting time $w_{ab}$ following Eq. (1).
3: if $\| d_2 - o_2 \| \leq \| d_1 - o_2 \|$ then
4:     Calculate delay time $d_{ab}$ following Eq. (2);
5: else
6:     Calculate delay time $d_{ab}$ following Eq. (3).
7: end if
```

C. Current-stage-based Deterministic Formulation

In each stage t, using current drivers and passengers, one can solve the following linear program to myopically match drivers with passenger groups and determine pick-up and drop-off routes, where set $S = \{(a, b) \in \mathcal{A}_t \times \mathcal{B}_t \mid N_a + N_b \leq c_1\}$ is defined as the set of all possible driver-passenger assignments constrained by vehicle capacities.

$$\begin{aligned} \min \quad & \lambda_1 \sum_{\substack{a \in \mathcal{A}_t^0 \cup \mathcal{A}_t^1 \\ (a,b) \in S}} x_{tab} w_{ab} + \lambda_2 \sum_{\substack{a \in \mathcal{A}_t^1 \\ (a,b) \in S}} x_{tab} d_{ab} + M \sum_{\substack{a \notin \mathcal{A}_t^0 \cup \mathcal{A}_t^1 \\ b \in \mathcal{B}_t}} x_{tab} \\ & + N \sum_{\substack{a \in \mathcal{A}_t^0 \cup \mathcal{A}_t^1 \\ (a,b) \notin S}} x_{tab} + P \sum_{b \in \mathcal{B}_t} \left(D_{tb} - \sum_{a \in \mathcal{A}_t^0 \cup \mathcal{A}_t^1} x_{tab} \right) \qquad (4) \\ \text{s.t.} \quad & \sum_{d \in \mathcal{B}_t} x_{tab} + x_{ta\emptyset} = R_{ta}, \ \forall a \in \mathcal{A}_t \qquad (5) \\ & \sum_{a \in \mathcal{A}_t} x_{tab} \leq D_{tb}, \ \forall b \in \mathcal{B}_t \qquad (6) \\ & x_{tad} \geq 0, \ \forall a \in \mathcal{A}_t, \ d \in \mathcal{D}. \qquad (7) \end{aligned}$$

In the objective function (4), the first two terms are a weighted sum of passengers' total waiting time and trip delay time where $\lambda_1 + \lambda_2 = 1, \ \lambda_1, \ \lambda_2 \geq 0$, and the last three terms denote the penalty costs associated with assigning unavailable drivers, exceeding vehicle capacities, and unsatisfied passengers' demand, respectively. Constraint (5) is flow conservation for drivers, i.e., the total number of drivers is capacitated. Constraint (6) is flow conservation for passengers, i.e., the number of drivers assigned to passengers with certain attribute is no more than the total number of passengers with that attribute. Since the constraint matrix composed by (5) and (6) is totally unimodular, integral solutions can be attained at all extreme points of the above linear program (4)–(7) (see [47]). Therefore, constraint (7) only requires that all the decision variables are non-negative, yielding a linear program to solve for each stage t, given updated R_{ta} and D_{tb} at the beginning of the stage.

III. Approximate Dynamic Programming

The traditional dynamic programming algorithm approximates value functions around pre-decision states, and therefore it requires the calculation of expectation over future information. We propose an ADP variant (see [12]) to approximate value functions around post-decision states.

A. Dynamic Process

Denote the state of the system in stage t as $S_t = (R_t, D_t)$, called the pre-decision state, meaning that S_t is measured before making a decision in stage t. The exogenous information is denoted as $W_t = \hat{D}_{tb}$, where $\hat{D}_{tb}$ is the number of new passengers with attribute b known between stages $t-1$ and t. The dynamic process in which the system evolves is

$$(S_0, \mathbf{x}_0, S_0^x, W_1, S_1, \mathbf{x}_1, S_1^x, W_2, S_2, \ldots, S_t, \mathbf{x}_t, S_t^x, \ldots, S_T).$$

Here S_t^x represents the state after taking decision $\mathbf{x}_t$, known as the post-decision state (see [12]). Denote $S_t^x = (R_t^x, D_t^x)$ where $R_t^x = (R_{ta}^x)_{a\in\mathcal{A}_{t,x}}$ and $D_t^x = (D_{tb}^x)_{b\in\mathcal{B}_t}$, which represent the "post-decision" values of R_t and D_t, respectively. Next, we describe how to construct post-decision states of resource R_t^x and demand D_t^x.

1) Post-decision State: For each pre-decision state $S_t = (R_t, D_t)$, define a transition function a^M that acts decision $\mathbf{x}_t$ on the attribute vector $a \in \mathcal{A}_t$ and let $a' = a^M(a, \mathbf{x}_t),\ \forall a \in \mathcal{A}_t$. Define set $\mathcal{A}_{t,x}$ with $a' = (o'_a, \mathcal{L}'_a, N'_a) \in \mathcal{A}_{t,x}$.

In each stage t, consider all pairs $(a, b) \in \mathcal{A}_t \times \mathcal{B}_t$ that have $x_{tab} > 0$. Recall that $b = (\mathbf{o}_b, \mathbf{d}_b, N_b)$, and set $o'_a = o_a,\ \mathcal{L}'_a = \mathcal{L}_a \cup \{\mathbf{o}_b, \mathbf{d}_b\},\ N'_a = N_a + N_b$. If $|\mathcal{L}'_a| = 3$, the route is determined using the shortest-path strategy in Section II-B2, and the driver is not available until the current trip is completed. To better represent post-decision states, consider an indicator function:

$$\delta_{\tilde{a}}(a, \mathbf{x}_t) = \begin{cases} 1, & \text{if } a^M(a, \mathbf{x}_t) = \tilde{a} \\ 0, & \text{otherwise.} \end{cases}$$

Then, the post-decision resource state $R_t^x = (R_{ta}^x)_{a\in\mathcal{A}_{t,x}}$ is updated using

$$R_{ta'}^x = \sum_{a\in\mathcal{A}_t} \sum_{d\in\mathcal{D}} \delta_{a'}(a, \mathbf{x}_t) x_{tad},\ \forall a' \in \mathcal{A}_{t,x}. \tag{8}$$

The post-decision demand state $D_t^x = (D_{tb}^x)_{b\in\mathcal{B}_{t,x}}$ is updated using

$$D_{tb}^x = D_{tb} - \sum_{a\in\mathcal{A}_t} x_{tab},\ \forall b \in \mathcal{B}_t. \tag{9}$$

The state transition function $S^{M,x}$ follows $S^{M,x}(S_t, \mathbf{x}_t) = S_t^x = (R_t^x, D_t^x)$ where the post-decision resource and demand state R_t^x, D_t^x follow (8) and (9), respectively.

2) Pre-decision State: For a post-decision state $S_t^x = (R_t^x, D_t^x)$, we gather the exogenous information W_{t+1} from stages t to $t+1$. To determine the pre-decision state $S_{t+1} = (R_{t+1}, D_{t+1})$, define a transition function $a^{M,W}$ that captures the physical movement of vehicles with attribute $a = (o_a, \mathcal{L}_a, N_a)$ from t to $t+1$. That is, $a' = a^{M,W}(a),\ \forall a \in \mathcal{A}_{t,x}$, where $a' = (o'_a, \mathcal{L}'_a, N'_a) \in \mathcal{A}_{t+1}$.

When $\mathcal{L}_a$ is nonempty, let l_1 be the first element in $\mathcal{L}_a$ (i.e., the first location that driver is going to visit). After finding the shortest path from o_a to l_1, let o'_a be where the driver arrives if he/she moves one stage from o_a to l_1 following the shortest path. Let $\mathcal{L}'_a = \mathcal{L}_a - \{l_1\}$ if $o'_a = l_1$. Also, let $N'_a = N_a - N_b$ if the driver drops off a passenger group with attribute $b = (\mathbf{o}_b, \mathbf{d}_b, N_b)$. Similarly, consider an indicator function:

$$\delta_{\tilde{a}}(a) = \begin{cases} 1, & \text{if } a^{M,W}(a) = \tilde{a} \\ 0, & \text{otherwise.} \end{cases}$$

Then the pre-decision resource state $R_{t+1} = (R_{t+1,a})_{a\in\mathcal{A}_{t+1}}$ is updated using

$$R_{t+1,a'} = \sum_{a\in\mathcal{A}_{t,x}} \delta_{a'}(a) R_{ta}^x,\ \forall a' \in \mathcal{A}_{t+1}. \tag{10}$$

Because unsatisfied demand is immediately lost in every stage (i.e., Assumption (iii)), the pre-decision demand state $D_{t+1} = (D_{t+1,b})_{b\in\mathcal{B}_{t+1}}$ can be updated as

$$D_{t+1,b} = W_{t+1},\ \forall b \in \mathcal{B}_{t+1}. \tag{11}$$

The state transition function $S^{M,W}$ is $S^{M,W}(S_t^x, W_{t+1}) = S_{t+1} = (R_{t+1}, D_{t+1})$, where the pre-decision resource and demand state R_{t+1}, D_{t+1} follow (10) and (11), respectively.

B. Dynamic Programming Equation

The objective function in problem (4)–(7) is denoted by $C_t(S_t, \mathbf{x}_t)$. Then the Bellman equation is conventionally given by:

$$V_t(S_t) \quad = \quad \min_{\mathbf{x}_t\in\mathbf{X}_t} \{C_t(S_t, \mathbf{x}_t) + \gamma \mathbb{E}[V_{t+1}(S_{t+1})|S_t]\} \tag{12}$$

where γ is the discount factor, $\mathbb{E}[V_{t+1}(S_{t+1})|S_t]$ denotes the conditional expectation of objective value at time $t+1$ given the current state S_t, and the feasible region $\mathbf{X}_t$ consists of constraints (5)–(7).

Using the post-decision state, Eq. (12) can be decomposed into two steps:

$$V_t(S_t) \quad = \quad \min_{\mathbf{x}_t\in\mathbf{X}_t} \{C_t(S_t, \mathbf{x}_t) + \gamma V_t^x(S_t^x)\}, \tag{13}$$

$$V_t^x(S_t^x) \quad = \quad \mathbb{E}[V_{t+1}(S_{t+1})|S_t^x]. \tag{14}$$

Next, we configure an approximation function $\bar{V}_t(S_t^x)$ for value function $V_t^x(S_t^x)$ around post-decision state S_t^x, which yields the following optimization problem:

$$F_t(S_t) = \min_{\mathbf{x}_t\in\mathbf{X}_t} \left\{C_t(S_t, \mathbf{x}_t) + \gamma \bar{V}_t(S_t^x)\right\}. \tag{15}$$

The basic algorithmic strategy is as follows: At iteration n, a sample path ω^n is randomly chosen, and then the following optimization problem for every time stage $t = 0, 1, \ldots, T$ is solved under the current approximation:

$$F_t(S_t^n) = \min_{\mathbf{x}_t \in \mathbf{X}_t} \left\{ C_t(S_t^n, \mathbf{x}_t) + \gamma \bar{V}_t^{n-1}(S^{M,x}(S_t^n, \mathbf{x}_t)) \right\}. \tag{16}$$

After obtaining the optimal solution $\mathbf{x}_t^n$, compute the post-decision state $S_t^{x,n} = S^{M,x}(S_t^n, \mathbf{x}_t)$ and the next pre-decision state $S_{t+1}^n = S^{M,W}(S_t^{x,n}, W_t(\omega^n))$. Then, solve the optimization problem (16) again to continue the process from stage t to stage $t+1$ until reaching stage T. After that, advance to the next iteration $n+1$. The algorithm is terminated if it reaches a given maximum number of iterations (denoted by N) and also the objective function becomes stable.

To efficiently computing (16), consider an approximation function that is linear in R_{ta}, given by:

$$\begin{aligned} \bar{V}_t^{n-1}(S_t^x) &= \bar{V}_t^{n-1}(R_t^x) = \sum_{a'} \bar{v}_{ta'} R_{ta'}^x \\ &= \sum_{a'} \bar{v}_{ta'} \sum_a \sum_d \delta_{a'}(a, \mathbf{x}_t) x_{tad} \\ &= \sum_a \sum_d \bar{v}_{t,a^M(a,\mathbf{x}_t)}^{n-1} x_{tad}. \end{aligned}$$

Then the optimization problem (16) becomes

$$F_t(S_t^n) = \min_{\mathbf{x}_t \in \mathbf{X}_t} \left\{ C_t(S_t^n, \mathbf{x}_t) + \gamma \sum_a \sum_d \bar{v}_{t,a^M(a,\mathbf{x}_t)}^{n-1} x_{tad} \right\}. \tag{17}$$

What is left now is how to update the coefficient $\bar{v}_{ta}^{n-1}$ in the linear approximation function. Note that $\bar{v}_{ta}^{n-1}$ is the slope of $F_t(S_t^n)$ with respect to R_{ta}^x, which can be computed using:

$$\hat{v}_{t-1,a}^n = \frac{\partial F(S_t)}{\partial R_{t-1,a}^x} = \sum_{a' \in \mathcal{A}_t} \frac{\partial F(S_t)}{\partial R_{ta'}} \frac{\partial R_{ta'}}{\partial R_{t-1,a}^x},$$

where $\partial F(S_t)/\partial R_{ta'}$ are the dual variables associated with constraints (5) in model (17), denoted by $\nu_{ta'}^n$. We have

$$\frac{\partial R_{ta'}}{\partial R_{t-1,a}^x} = \begin{cases} 1, & \text{if } a' = a^{M,W}(a_{t-1}^x) \\ 0, & \text{otherwise.} \end{cases}$$

This means that if from stage $t-1$ to stage t, attribute a_{t-1}^x evolves to $a' = a^{M,W}(a_{t-1}^x)$, then we only need to optimize over variable $\nu_{ta'}^n$, and update $\hat{v}_{t-1,a'}^n = \nu_{ta'}^n$. After obtaining $\hat{v}_{t-1,a'}^n$, the coefficient can be updated using

$$\bar{v}_{t-1,a_{t-1}}^n = (1 - \alpha_{n-1}) \bar{v}_{t-1,a_{t-1}}^{n-1} + \alpha_{n-1} \hat{v}_{t-1,a'}^n, \tag{18}$$

where α_{n-1} is a stepsize in iteration n. The above steps of ADP are summarized in Algorithm 2.

Algorithm 2 ADP

```
 1: Initialize value functions $\bar{V}_t^0$, $t = 0, 1, \ldots, T$ and state $S_0^1$. Set $n = 1$.
 2: while $n < N$ do
 3:     Randomly pick a sample path $\omega^n$.
 4:     for $t = 0, 1, \ldots, T$ do
 5:         Gather all drivers' state $\mathcal{A}_t$ and passengers' requests $\mathcal{B}_t$. For each pair
            of $a \in \mathcal{A}_t$ and $b \in \mathcal{B}_t$, implement Algorithm 1 to obtain parameters
            $w_{ab}$, $d_{ab}$.
 6:         Solve the optimization problem (16). Let $\mathbf{x}_t^n$ be the optimal solution, and
            $\nu_{ta'}$ be the dual associated with the resource constraint.
 7:         Update the value function using (18).
 8:         Update the state:
                $S_t^{x,n} = S^{M,x}(S_t^n, \mathbf{x}_t)$, $S_{t+1}^n = S^{M,W}(S_t^{x,n}, W_t(\omega^n))$.
 9:     end for
10:     $n := n + 1$
11: end while
```

C. Value Function Properties

Suppose that the state space is equipped with a partial order $\preceq$, then a value function is monotone if it satisfies

$$S_t \preceq S_t' \ \Rightarrow \ V_t(S_t) \leq V_t(S_t'), \ \forall t \leq T. \tag{19}$$

A common example of $\preceq$ is the generalized component-wise inequality. In the ADP approach, our state can be decomposed into $S_t = (R_t, D_t)$. For any two states $S_t = (R_t, D_t), S_t' = (R_t', D_t')$, we have

$$S_t \preceq S_t' \iff R_t \leq R_t', \ D_t = D_t'. \tag{20}$$

The monotonicity of the value function indicates that we have more resource in stage $t+1$ if starting with more resource in stage t, regardless of the outcome of the random information W_{t+1}.

Theorem 1. *Let $\preceq$ be the generalized component-wise inequality over all dimensions of the state space. The optimal value function is monotone based on* (19) *and* (20).

The detailed proof of Theorem 1 is provided in the online e-companion.

We continue exploring quantitative properties of ADP value functions. The linear program in each time t is:

$$\begin{aligned} \min \quad & \sum_{\substack{a \in \mathcal{A}_t \\ d \in \mathcal{D}}} (c_{tad} + \gamma v_{t,a^M(a,d)}^n) x_{tad} \\ \text{s.t.} \quad & (5)\text{–}(7), \end{aligned}$$

where c_{tad} is the cost related to each action x_{tad}, and $v_{t,a^M(a,d)}^n$ is the approximate value function. (We abbreviate $v_{t,a^M(a,d)}^n$ as v_{ad}^n in our later discussion.) Using specific cost terms in the objective function (4), the dual of the above linear program is

$$\begin{aligned} \max \quad & \sum_{a \in \mathcal{A}_t} \nu_a R_{ta} + \sum_{b \in \mathcal{B}_t} \mu_b D_{tb} & (21) \\ \text{s.t.} \quad & \nu_a + \mu_b \leq \lambda_1 w_{ab} - P + \gamma v_{ab}^n, \ \forall a \in \mathcal{A}_t^0, \ (a,b) \in S & (22) \\ & \nu_a + \mu_b \leq \lambda_1 w_{ab} + \lambda_2 d_{ab} - P + \gamma v_{ab}^n, \ \forall a \in \mathcal{A}_t^1, \ (a,b) \in S & (23) \\ & \nu_a + \mu_b \leq N + \gamma v_{ab}^n, \ \forall a \in \mathcal{A}_t^0 \cup \mathcal{A}_t^1, \ (a,b) \notin S & (24) \\ & \nu_a + \mu_b \leq M + \gamma v_{ab}^n, \ \forall a \notin \mathcal{A}_t^0 \cup \mathcal{A}_t^1, \ b \in \mathcal{B}_t & (25) \\ & \nu_a \leq \gamma v_{a\emptyset}^n, \ \forall a \in \mathcal{A}_t & (26) \\ & \mu_b \leq 0, \ \forall b \in \mathcal{B}_t, & (27) \end{aligned}$$

where ν_a is the Lagrangian multiplier (dual variable) associated with constraint (5), and μ_b is the dual variable associated with constraint (6).

Based on strong duality and complementary slackness, the following result reveals the relationship between dual variables and value functions.

Lemma 1.
1) If $a \notin \mathcal{A}_t^0 \cup \mathcal{A}_t^1$ or $a \in \mathcal{A}_t^0 \cup \mathcal{A}_t^1$, $(a,b) \notin S$, $\forall b \in \mathcal{B}_t$, then $\nu_a = \gamma v_{a\emptyset}^n$.
2) If $a \in \mathcal{A}_t^0$, then
$\nu_a \geq \min_{b:(a,b) \in S} \{\lambda_1 w_{ab} - P + \gamma v_{ab}^n\}$;

3) if $a \in \mathcal{A}_t^1$, then
$\nu_a \geq \min_{b:(a,b)\in S} \{\lambda_1 w_{ab} + \lambda_2 d_{ab} - P + \gamma v_{ab}^n\}$.

Please refer to the online e-companion for the detailed proof of Lemma 1.

IV. Temporal and Spatial Decomposition

Two decomposition schemes based on time and location are proposed to improve the solution time of the ADP approach.

A. *Temporal Decomposition*

When making driver-passenger matching and dispatching decisions, it is often unnecessary to incorporate the uncertainty over the entire operational time horizon. On the other hand, as most trips end within shorter time frame, it becomes important to determine the best time duration to plan ahead. To implement the ADP algorithm, we propose a temporal decomposition scheme (see Figure 1) to train data.

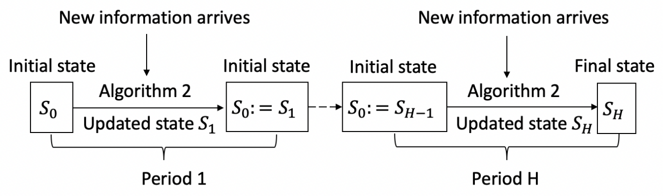


Fig. 1. Procedure of Temporal Decomposition

Algorithm 3 describes the algorithmic details for temporal decomposition. The overall time horizon is divided into H periods with each period having T stages, and then the ADP approach is implemented in a rolling-horizon manner. That is, for each period, we perform Algorithm 2 on the corresponding T stages to find an optimal solution and implement it in the current period to see where the drivers are after actions are taken and new information arrives. Then, we move one period forward, update the system state, and implement Algorithm 2 again. Note that for each period $h = 1, \dots, H$, the procedures are divided into "Training" and "On-demand Solution" (ODS) phases, of which the former can be done offline using historical data and the latter accounts for the time needed for on-demand response. In Section V, we compare the CPU time taken by "training" versus "ODS" for different-sized problems.

Algorithm 3 Temporal Decomposition

```
1: Divide the whole operational time into $H$ periods.
2: for $h = 1, \dots, H$ do
3:     Training: Implement Algorithm 2 on period $h$ with initial state $S_0$.
4:     ODS: Update drivers' status according to the optimal solution and corresponding passengers' information at the end of period $h$.
5:     Set the end state of the current period as the initial state for next period: $S_0 := S_T$.
6: end for
```

B. *Spatial Decomposition*

We further divide a given service region into L sub-regions along with a subset of drivers in each sub-region given their initial locations. Drivers belong to each region only serve for trips within that region. For trips across regions, we create a $(L+1)$-th sub-region, namely, the Cross region and its own subset of drivers whose travel trajectories are "mostly" (specified later) across sub-regions based on historical data. Take New York City for example. It has five boroughs clearly defined by its daily taxi data, labeled from 1 to 5 in Figure 2. Moreover, we consider a sixth Cross region enclosed by the red dashed box.

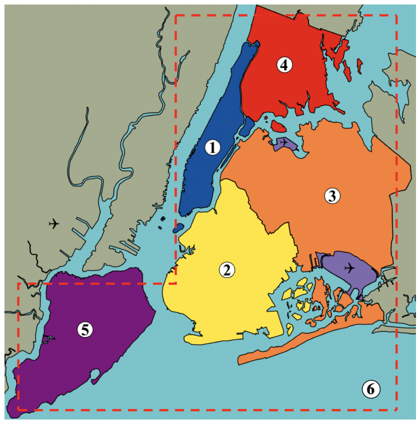


Fig. 2. Map and Spatial Decomposition of New York City

Specifically, we determine the subset of drivers that serve the Cross region using the percentage of cross-region trips at the initial state. Given initial state $S_0 = (R_0, D_0)$, for all $l = 1, \dots, L$, denote the number of drivers in region l by r_l, the number of trips starting from region l by d_l, and the number of trips starting from region l but ending in other regions by e_l. Then for drivers in region l, we randomly choose $r_l \frac{e_l}{d_l}$ drivers to serve region $L+1$ and keep the rest $r_l(1 - \frac{e_l}{d_l})$ drivers to serve region l, for all $l = 1, \dots, L$. Also, denote the corresponding initial state in region l by $S_{0l} = (R_{0l}, D_{0l})$ where $|R_{0l}| = r_l(1 - \frac{e_l}{d_l})$, $|D_{0l}| = d_l - e_l$, for all $l = 1, \dots, L$, and $|R_{0,L+1}| = \sum_{l=1}^L r_l \frac{e_l}{d_l}$, $|D_{0,L+1}| = \sum_{l=1}^L e_l$.

We illustrate the drivers' assignment in Table I, and provide the details of spatial decomposition heuristic in Algorithm 4.

TABLE I
Illustration of Drivers' Assignment

Region	In-region drivers	Out-region drivers
$l = 1, \dots, L$	$r_l(1 - \frac{e_l}{d_l})$	$r_l \frac{e_l}{d_l}$
$L + 1$	$\sum_{l=1}^L r_l \frac{e_l}{d_l}$	N.A.

Algorithm 4 Spatial Decomposition

```
1: Given initial state $S_0$, choose $r_l(1 - \frac{e_l}{d_l})$ drivers in region $l$ to form the initial state $S_{0l}$ for $l = 1, \dots, L$. Also, form the initial state $S_{0,L+1}$ using the rest $\sum_{l=1}^L r_l \frac{e_l}{d_l}$ drivers.
2: for $l = 1, \dots, L+1$ do
3:     Implement Algorithm 3 on region $l$ with initial state $S_{0l} = (R_{0l}, D_{0l})$.
4: end for
```

Note that Step 3 in Algorithm 4 can be performed in parallel given the spatial-independence of all $L+1$ sub-regions.

V. Numerical Results

We first test randomly generated small instances to configure parameter for implementing ADP (see the online e-companion), and then use instances generated based on real-world data collected from New York City taxi daily operations to perform numerical tests. We use the New York City data for one peak-hour operations (from 8am to 9am). Two benchmark policies are described below.

- **Benchmark 1 (B1):** Solve the linear program (4)–(7) for the current stage given information of existing drivers and passengers. Repeat the process for each stage in a rolling horizon way.
- **Benchmark 2 (B2):** Apply decomposition-based ADP without ride-pooling.

A. Experimental Setup

According to the parameter configuration results, we set $M = 1000,\ N = 1000,\ P = 500$, weight $\lambda = (0.2, 0.8)$, and stepsize $\alpha_n = 1/n$ in each iteration n of the ADP algorithm. Our test instances are based on data from the New York City Taxi and Limousine Commission (TLC) (see [48]). The taxi trip records include pick-up and drop-off dates/times, locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts. There are 265 different locations among all five boroughs (i.e., Manhattan, Bronx, Brooklyn, Queens and Staten). In Table II, we summarize the total number of trips in one month within each borough (see Row 'In'), going out of each borough (see Row 'Out'), and their approximate ratios. Note that the number of trips within Cross region is the sum of out-region trips in all the five boroughs. Staten has much fewer trips compared to other boroughs, and thus we eliminate this borough in our tests. The ratio of trips among all remaining five sub-regions is roughly 1:0.1:1:1:0.5, which is used to generate their associated drivers in the initial state.

TABLE II
Trips distribution among boroughs

	Manhattan	Bronx	Brooklyn	Queens	Staten	Cross
In	301618	30833	296324	271349	123	162155
Out	35639	9747	84577	32158	34	N.A.
Ratio	1	0.1	1	1	N.A.	0.5

We first test a finite horizon with each stage length being 5 minutes. The data from 1/1/17 to 6/30/17 (181 days) during the rush hour 8am-9am, and is divided into 12 stages. For each stage and each location, we fit the historical data into the negative binomial distribution, and then generate test instances. The average trip duration of all the data is 14.7 minutes with a standard deviation 1 minute and 42 seconds, and the average number of trips per hour is 1523. Drivers' profits are calculated using a base fee \$2.5 plus \$2 per mile similar to New York City Taxi fares. Let $|\mathcal{A}|$ and $|\mathcal{B}|$ be the total number of drivers and passengers in all five regions, respectively. We vary their values to change instance sizes in our later analysis.

B. In-sample Tests and Results of Smaller Instances

First, we focus on time period 8:00am-8:25am and Manhattan borough with 69 nodes. Let $T = 5,\ |\mathcal{A}| = 10, 20$. In Table III, we display waiting time (WT) and delay time (DT) per passenger per stage, profits per driver per stage and proportion of unsatisfied demand (UD) per stage.

TABLE III
Comparison of the results of ADP, B1, and B2

$\|\mathcal{A}\|$	$\|\mathcal{B}\|$	Metrics	ADP	B1	B2
		WT (min.)	4.33	4.11	4.29
		DT (min.)	1.54	1.28	0.00
10	54	Profits (\$)	9.72	9.67	4.34
		UD (%)	18.77%	19.82%	33.25%
		WT (min.)	4.72	4.26	4.55
		DT (min.)	2.01	1.56	0.00
10	72	Profits (\$)	10.43	9.83	4.31
		UD (%)	18.14%	19.46%	31.57%
		WT (min.)	3.62	3.29	4.8
		DT (min.)	0.21	0.17	0.00
20	56	Profits (\$)	5.73	5.90	4.06
		UD (%)	0.68%	1.01%	6.62%
		WT (min.)	4.22	3.73	5.08
		DT (min.)	0.49	0.25	0.00
20	73	Profits (\$)	6.27	6.08	4.03
		UD (%)	0.81%	1.27%	7.76%

In Table III, when the number of drivers increases from 10 to 20, the proportion of unsatisfied demand for ADP drops from above 18% to below 1%, showing much better quality of service. Comparing ADP, B1 and B2, we observe that ADP always yields the lowest unsatisfied demand rate, while B2 always has the highest rate. However, ADP may result in longer waiting time than B1 because the drivers pick up more passengers. Although B2 always has zero delay time (since it does not allow for pooling), it performs badly in waiting time, drivers' profits and unsatisfied demand. When we increase $|\mathcal{B}|$ while keeping $|\mathcal{A}|$ unchanged, almost all the above results in Table III increase slightly.

In summary, having more drivers can lead to better performance in waiting time, delay time and proportion of unsatisfied demand, although profits per driver could become slightly less.

C. Results of Larger Instances

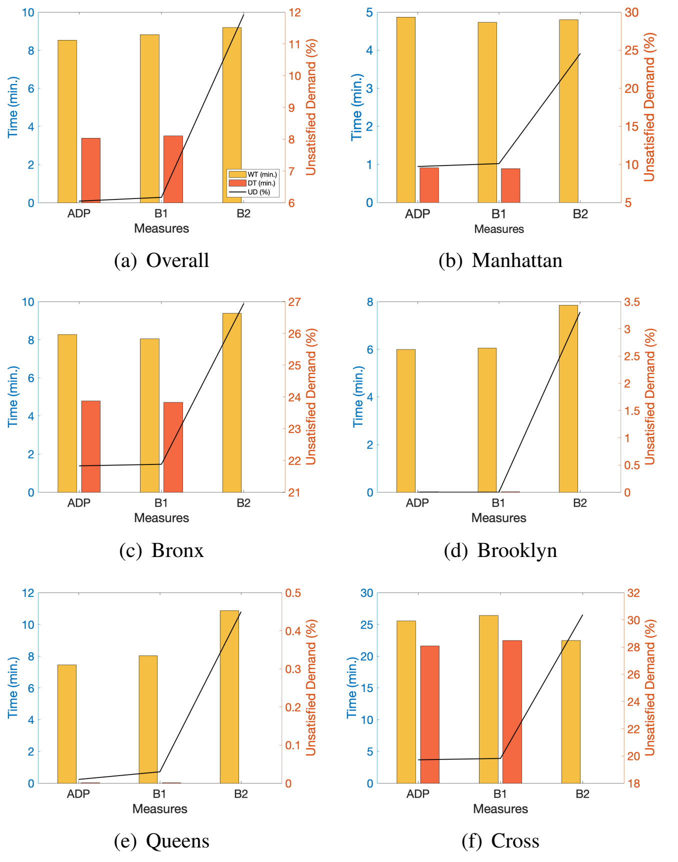
We then test the decomposition-based ADP for all five sub-regions of New York City for 1-hour ride-pooling operations. Let $T = 4,\ H = 3$ (and thus 12 stages in total with each stage being 5 minutes). In Table IV, the results in each region are aggregated by a weighted sum where the weight is the ratio of drivers in each region.

We focus on the case where $|\mathcal{A}| = 108,\ |\mathcal{B}| = 393$ and use bar charts to illustrate the overall performance in Table IV as well as the performance in each region separately. In Figure 3, x-axis denotes different approaches. From Table IV, when the driver-passenger ratio is approximately 1:4, the proportion of unsatisfied demand is around 6%, produced by ADP, whereas it drops to below 0.3% when the driver-passenger ratio becomes 1:2. This result can provide guidelines for ridesharing operators to control driver-passenger balance

TABLE IV
RESULTS OF ADP, B1, B2 WHEN $T = 4,\ H = 3$ FOR ONE-HOUR TAXI OPERATIONS IN NEW YORK CITY

$\|\mathcal{A}\|$	$\|\mathcal{B}\|$	Metrics	ADP	B1	B2
108	286	WT (min.)	6.31	6.33	9.26
		DT (min.)	0.31	0.29	0.00
		Profits (\$)	5.57	5.51	4.06
		UD (%)	0.28%	0.34%	2.40%
108	393	WT (min.)	8.53	8.81	9.19
		DT (min.)	3.39	3.51	0.00
		Profits (\$)	7.32	7.26	4.21
		UD (%)	6.05%	6.17%	11.92%
180	283	WT (min.)	4.09	4.20	5.16
		DT (min.)	0.00	0.00	0.00
		Profits (\$)	3.92	3.92	3.50
		UD (%)	0.00%	0.00%	0.02%
180	389	WT (min.)	4.06	4.17	5.66
		DT (min.)	0.11	0.11	0.00
		Profits (\$)	4.31	4.33	3.69
		UD (%)	0.15%	0.16%	0.40%

with desired quality-of-service levels. From Figure 3, in all five regions, ADP yields the lowest unsatisfied demand rate, where B2 always gains the longest waiting time and the highest unsatisfied demand. On the other hand, across different regions, Brooklyn has the best results with the lowest unsatisfied demand rate and shortest delay time, while Cross region yields the longest waiting and delay time because the related trips usually have longer travel distances.



(a) Overall (b) Manhattan (c) Bronx (d) Brooklyn (e) Queens (f) Cross

Fig. 3. Performance overall and in different regions

D. Value of Uncertainty

Now consider Manhattan borough and uncertain demand over twelve time stages with each stage being 5-minute long (i.e., 60 minutes). We first solve the problem containing the first six stages (i.e., 30 minutes), update the state of the system, and solve the problem for the later six stages. Alternatively, we repeatedly solve the problem for every four/three/two stages (i.e., 20/15/10 minutes) to further reduce the problem dimension and the number of stages we "look ahead." We use $|\mathcal{A}| = 30,\ |\mathcal{B}| = 120$ and present the results of $T = 6,\ T = 4,\ T = 3,\ T = 2$ in Table V. The cases with $T = 4$ and $T = 3$ perform relatively better overall. Both cases with $T = 6$ and $T = 2$ have worse unsatisfied demand, and the latter is much worse than the other three in all three approaches. However, $T = 2$ has slightly better waiting time, trip delay time, and profit. These results agree on that it may not be necessary to take into account the information 20 minutes from now since the average trip duration is 14.7 minutes with standard deviation being roughly 1 minute. However, if the dispatcher only forecasts one or two stages' future demand, the unsatisfied demand rate is intolerably high, indicating the importance of looking-ahead, stochastic policies.

We further test the model with each stage being one minute to examine result sensitivity dependent on the granularity of data. We consider demand during 8am to 8:30am (i.e., 30 stages) and look ahead 5-minute demand uncertainty each time, resulting in parameter setting $T = 6,\ H = 5$. We compare the solutions with the ones in the previous case having each stage being five minutes and use parameter setting $T = 2,\ H = 3$ so that we also look ahead five minutes each time. The corresponding results are displayed in Table VI. We observe that cases with 1-minute stage length have shorter waiting and delay time, but they have much higher and unacceptable unsatisfied demand rates due to Assumption (i).

TABLE V
VALUE OF UNCERTAINTY RESULTS BY VARYING T AND H

T	H	Metrics	ADP	B1	B2
6	2	WT (min.)	3.94	3.54	4.47
		DT (min.)	0.77	0.62	0.00
		Profits (\$)	7.49	7.34	4.10
		UD (%)	2.90%	2.71%	15.67%
4	3	WT (min.)	3.77	3.70	4.97
		DT (min.)	0.14	0.09	0.00
		Profits (\$)	6.21	6.17	4.14
		UD (%)	0.54%	0.74%	6.11%
3	4	WT (min.)	5.00	5.02	7.81
		DT (min.)	0.12	0.15	0.00
		Profits (\$)	5.67	5.72	3.88
		UD (%)	0.09%	0.19%	7.50%
2	6	WT (min.)	2.33	2.34	2.57
		DT (min.)	0.17	0.18	0.00
		Profits (\$)	8.36	8.40	4.50
		UD (%)	29.71%	29.85%	43.58%

TABLE VI
VALUE OF UNCERTAINTY RESULTS BY VARYING TIME UNIT

T	H	Metrics	ADP	B1	B2
6	5	WT (min.)	0.75	0.81	0.66
		DT (min.)	0.28	0.48	0.00
		Profits ($)	3.17	3.16	2.68
		UD (%)	81.3%	81.36%	83.97%
2	3	WT (min.)	2.86	2.89	2.93
		DT (min.)	0.18	0.19	0.00
		Profits ($)	9.24	9.1	4.51
		UD (%)	28.32%	28.62%	45.29%

E. Computational Time

We end this section by showing the computational time for different instances. First, in Table VII we report the CPU time for solving instances having each stage being five minutes with $T = 4,\ H = 3$. As the decomposed ADP can be implemented for each region in parallel, we record the maximum time used by each of the five regions. Moreover, the training time for each period is recorded separately in Columns "$h = 1$", "$h = 2$" and "$h = 3$", and the on-demand solution time is presented in Column "ODS". All the training steps can be performed *offline* while the on-demand implementation just extracts the updated value functions to solve a linear program in each stage, which only takes no more than 90 seconds for the largest instance and it is almost linearly dependent on the number of drivers and passengers. All times that exceed 3600-second CPU time limit are labeled by N.A.

TABLE VII
TRAINING TIME AND ON-DEMAND SOLUTION (ODS) TIME (IN SECONDS)

$\|\mathcal{A}\|$	$\|\mathcal{B}\|$	$h = 1$	$h = 2$	$h = 3$	ODS
108	285	83.62	83.85	82.03	12.26
	390	149.51	149.63	148.85	11.58
	498	354.58	351.82	350.93	24.56
180	285	145.47	143.7	143.2	10.09
	390	402.18	400.9	403.48	22.14
	498	861.76	825.43	817.29	45.08
360	285	517.86	519.14	518.78	82.21
	390	1354.97	1291.5	1290.5	90.01
	498	N.A.	N.A.	N.A.	N.A.

Second, we compare the CPU time taken by the two settings used in Section V-D. Specifically, when letting each stage being five minutes (with $T = 2$ and $H = 3$), on average the instances take 20.34 seconds, 18.28 seconds, and 22.37 seconds for training data in periods $h = 1, 2, 3$, respectively, and the on-demand computation only requires 2.01 seconds. However, when each stage is one minute (with $T = 6$, $H = 5$), based on the same 30-minute data, the ADP approach takes around 2400 seconds for training in each period, and the ODS phase requires 275.5 seconds. This also justifies our choice of letting each stage being five minutes.

VI. CONCLUSIONS

We considered ride-pooling problem with no more than two passenger groups sharing rides at the same time. We employed the ADP approach to solve the problem dynamically and exploited properties of value functions. A decomposition heuristic was developed to divide the whole space and operation time into sub-regions and several periods. Numerical results showed quick convergence and result stability of using ADP. We compared ADP with two benchmarks, and demonstrated that it can serve the most passengers among all, showing the importance of including future demand uncertainty into ride-pooling decision making. Also, ADP led to shorter waiting time per passenger as compared to the benchmark with no ride-pooling, showing the importance of pooling rides in ride-hailing systems.

For future research, we plan to investigate ride pooling problems allowing more than two passenger groups in the same vehicle. One can also incorporate pricing and vehicle relocation into the current operational model, to investigate how they affect ride pooling strategies.

ACKNOWLEDGMENT

We thank the Associate Editor and four anonymous reviewers for their detailed comments and constructive feedback. The authors are grateful for the support of the NSF grant CMMI-1727618.

REFERENCES

[1] S. Wallsten, "The competitive effects of the sharing economy: How is Uber changing taxis," *Technology Policy Institute*, vol. 22, 2015.

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

[4] J. Jacob and R. Roet-Green, "Ride solo or pool: The impact of sharing on optimal pricing of ride-sharing services," *Available at SSRN: https://ssrn.com/abstract=3008136 or http://dx.doi.org/10.2139/ssrn.3008136*, 2017.

[5] S. Agarwal, "Uberpool has saved $4.5 million worth in fuel import costs in india since launch," https://economictimes.indiatimes.com/small-biz/startups/newsbuzz/uberpool-has-saved-4-5-million-worth-in-fuel-import-costs-in-india-since-launch/articleshow/64455325.cms, 2018.

[6] M. Ota, H. Vo, C. Silva, and J. Freire, "STaRS: Simulating taxi ride sharing at scale," *IEEE Transactions on Big Data*, vol. 3, no. 3, pp. 349–361, 2017.

[7] 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.

[8] J. Alonso-Mora, A. Wallar, and D. Rus, "Predictive routing for autonomous mobility-on-demand systems with ride-sharing," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3583–3590.

[9] M. Zhu, X.-Y. Liu, and X. Wang, "An online ride-sharing path-planning strategy for public vehicle systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 616–627, 2018.

[10] Z. Xu, Z. Li, Q. Guan, D. Zhang, Q. Li, J. Nan, C. Liu, W. Bian, and J. Ye, "Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach," in *The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 905–913.

[11] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[12] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, 2007, vol. 703.

[13] N. Agatz, A. L. Erera, M. W. Savelsbergh, and X. Wang, "Dynamic ride-sharing: A simulation study in metro Atlanta," *Procedia-Social and Behavioral Sciences*, vol. 17, pp. 532–550, 2011.

[14] M. Lowalekar, P. Varakantham, and P. Jaillet, "Online spatio-temporal matching in stochastic and dynamic domains," *Artificial Intelligence*, vol. 261, pp. 71–112, 2018.

[15] C. Dutta and C. Sholley, "Online matching in a ride-sharing platform," *arXiv preprint arXiv:1806.10327*, 2018.

[16] E. Ozkan and A. R. Ward, "Dynamic matching for real-time ridesharing," *Available at SSRN: https://ssrn.com/abstract=2844451*, 2016.

[17] S. Banerjee, D. Freund, and T. Lykouris, "Pricing and optimization in shared vehicle systems: An approximation framework," *arXiv preprint arXiv:1608.06819*, 2016.

[18] L. Zha, Y. Yin, and Z. Xu, "Geometric matching and spatial pricing in ride-sourcing markets," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 58–75, 2018.

[19] A. Fiat, Y. Mansour, and L. Shultz, "Flow equilibria via online surge pricing," *arXiv preprint arXiv:1804.09672*, 2018.

[20] M. Chen, W. Shen, P. Tang, and S. Zuo, "Optimal vehicle dispatching for ride-sharing platforms via dynamic pricing," in *Companion of the The Web Conference 2018*. International World Wide Web Conferences Steering Committee, 2018, pp. 51–52.

[21] V. Kamble, "On optimal pricing of services in on-demand labor platforms," *arXiv preprint arXiv:1803.06797*, 2018.

[22] A. Gupta, B. Saha, and P. Banerjee, "Pricing decisions of car aggregation platforms in sharing economy: A developing economy perspective," *Journal of Revenue and Pricing Management*, pp. 1–15, 2018.

[23] R. Iglesias, F. Rossi, K. Wang, D. Hallac, J. Leskovec, and M. Pavone, "Data-driven model predictive control of autonomous mobility-on-demand systems," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–7.

[24] K. Spieser, S. Samaranayake, W. Gruel, and E. Frazzoli, "Shared-vehicle mobility-on-demand systems: A fleet operator's guide to rebalancing empty vehicles," in *Transportation Research Board 95th Annual Meeting*, no. 16-5987. Transportation Research Board, 2016.

[25] H. R. Sayarshad and J. Y. Chow, "Non-myopic relocation of idle mobility-on-demand vehicles as a dynamic location-allocation-queueing problem," *Transportation Research Part E: Logistics and Transportation Review*, vol. 106, pp. 60–77, 2017.

[26] A. Braverman, J. G. Dai, X. Liu, and L. Ying, "Empty-car routing in ridesharing systems," *arXiv preprint arXiv:1609.07219*, 2016.

[27] F. Miao, S. Han, A. M. Hendawi, M. E. Khalefa, J. A. Stankovic, and G. J. Pappas, "Data-driven distributionally robust vehicle balancing using dynamic region partitions," in *The 8th International Conference on Cyber-Physical Systems*. ACM, 2017, pp. 261–271.

[28] R. Zhang and M. Pavone, "Control of robotic mobility-on-demand systems: A queueing-theoretical perspective," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 186–203, 2016.

[29] S. Yan, C.-Y. Chen, and C.-C. Wu, "Solution methods for the taxi pooling problem," *Transportation*, vol. 39, no. 3, pp. 723–748, 2012.

[30] X. Wei, "Routing for taxi-pooling problem based on ant colony optimization algorithm," *Revista de la Facultad de Ingeniería*, vol. 31, no. 7, 2016.

[31] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti, "Quantifying the benefits of vehicle pooling with shareability networks," *Proceedings of the National Academy of Sciences*, vol. 111, no. 37, pp. 13 290–13 294, 2014.

[32] J. Wen, N. Nassir, and J. Zhao, "Value of demand information in autonomous mobility-on-demand systems," *Transportation Research Part A: Policy and Practice*, vol. 121, pp. 346–359, 2019.

[33] J. Miller and J. P. How, "Predictive positioning and quality of service ridesharing for campus mobility on demand systems," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1402–1408.

[34] F. Miao, S. Han, S. Lin, Q. Wang, J. A. Stankovic, A. Hendawi, D. Zhang, T. He, and G. J. Pappas, "Data-driven robust taxi dispatch under demand uncertainties," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 1, pp. 175–191, 2017.

[35] G. Laporte, "What you should know about the vehicle routing problem," *Naval Research Logistics*, vol. 54, no. 8, pp. 811–819, 2007.

[36] J.-F. Cordeau and G. Laporte, "The dial-a-ride problem: Models and algorithms," *Annals of Operations Research*, vol. 153, no. 1, p. 29, 2007.

[37] M. W. Savelsbergh and M. Sol, "The general pickup and delivery problem," *Transportation Science*, vol. 29, no. 1, pp. 17–29, 1995.

[38] G. Berbeglia, J.-F. Cordeau, and G. Laporte, "Dynamic pickup and delivery problems," *European Journal of Operational Research*, vol. 202, no. 1, pp. 8–15, 2010.

[39] 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.

[40] J. Jung, R. Jayakrishnan, and J. Y. Park, "Dynamic shared-taxi dispatch algorithm with hybrid-simulated annealing," *Computer-Aided Civil and Infrastructure Engineering*, vol. 31, no. 4, pp. 275–291, 2016.

[41] S. Ma, Y. Zheng, O. Wolfson *et al.*, "Real-time city-scale taxi ridesharing." *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 7, pp. 1782–1795, 2015.

[42] S. Ma, Y. Zheng, and O. Wolfson, "T-share: A large-scale dynamic taxi ridesharing service," in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 2013, pp. 410–421.

[43] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming: An overview," in *The 34th IEEE Conference on Decision and Control*, vol. 1. IEEE, 1995, pp. 560–564.

[44] C. Novoa and R. Storer, "An approximate dynamic programming approach for the vehicle routing problem with stochastic demands," *European Journal of Operational Research*, vol. 196, no. 2, pp. 509–515, 2009.

[45] V. Schmid, "Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming," *European Journal of Operational Research*, vol. 219, no. 3, pp. 611–621, 2012.

[46] A. J. Kleywegt, V. S. Nori, and M. W. Savelsbergh, "Dynamic programming approximations for a stochastic inventory routing problem," *Transportation Science*, vol. 38, no. 1, pp. 42–70, 2004.

[47] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. New York, NY: Wiley-Interscience, 1999.

[48] N. Taxi and L. Commission, "New York City's taxi trip data," http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml/, 2018, [Online; accessed 19-March-2018].

Xian Yu Xian Yu received the B.S. degree from the Honors Science Program in Mathematics in Xi'an Jiaotong University, Xi'an, China, in 2017. She is currently a PhD candidate in the Department of Industrial and Operations Engineering at the University of Michigan, Ann Arbor. Her research interests include stochastic integer programming with applications in transportation and logistics.

Siqian Shen Siqian Shen received the B.S. degree in Industrial Engineering from Tsinghua University, China, in 2007, and the M.S. and Ph.D. degrees in Industrial and Systems Engineering from the University of Florida, USA, in 2009 and 2011, respectively. She is an Associate Professor in the Department of Industrial and Operations Engineering, University of Michigan at Ann Arbor, and also an Associate Director for the Michigan Institute for Computational Discovery & Engineering (MICDE). Her research interests include stochastic programming, network optimization, and integer programming. Applications of her work include transportation and energy.