Urban Ridesharing with Hybrid Distributed Reinforcement Learning

Benjamin Riviere, Salar Rahili, and Soon-Jo Chung

***Abstract*—Coordinating a fleet of automated taxis to optimally service urban customer requests is an anticipated challenge for Transportation Network Companies (TNC) like Uber and Lyft. In this paper, we derive a learning-based algorithm to assign taxis to customer requests to maximize long-term profit in an unknown and dynamic environment. We propose a novel, hybrid mechanism using distributed online temporal-difference (D-TD) learning with infrequent centralized Bellman updates to solve this problem in a scalable manner with regret bounds. The system is modeled as a Markov Decision Process (MDP) using customer data. The Bellman solution of the MDP is computed at a powerful central node and is passed as an initial policy to the fleet of taxis. Then, using only local information, the D-TD algorithm is computed at each taxi to capture the model errors and the environment changes, such as variations in customer demand, traffic, and fare. This solution will drift from the centralized Bellman solution, so periodically, the agents will pass their customer data to a centralized node to again exactly solve the MDP and obtain an exactly optimal centralized policy. In contrast to conventional agent-based MDP, we formulate a highly-scalable cell-based MDP, and we exploit a game-theoretic task assignment algorithm to coordinate individual taxis, where each agent uses the optimal policy to select its customer from the set of local available requests in a distributed manner. We validate the proposed algorithm in two environments: with a simulated GridWorld and with real customer requests from the Chicago city taxi dataset.**

***Index Terms*—Autonomous Vehicles, Distributed Algorithms, Dynamic Programming**

I. Introduction

URBAN transportation plays a significant role in the development of modern cities. Almost 1.2 million deaths occur on roads each year worldwide, and reports show that 94% of car accidents in the U.S. involve human errors [1]. Autonomous cars are an emergent technology that will quickly become ubiquitous as a safer and more efficient mode of transportation. Transportation Network Companies (TNCs) are planning to deploy coordinated fleets of autonomous ground and air vehicles to improve the urban transportation capabilities [2], and provide autonomous taxi services (see Fig. 1). The deployment of a large number of agent fleets of autonomous vehicles, both ground and air, drives a coupled innovation in highly-scalable distributed algorithms. An example of an intelligent transportation network architecture is a computationally powerful central node and a large fleet of distributed taxis that operate with a small amount of processing power and communication bandwidth. In this paper, we assume this architecture and develop a framework to optimize infinite-horizon reward in a dynamic and uncertain environment.

This work develops a method to exploit the natural advantages of exact and approximate dynamic programming methods to coordinate mobile transportation agents in an urban environment. Exact methods to solve a large state-space Markov Decision Process (MDP) [3], [4] with Bellman value iteration is computationally expensive and thus cannot be computed in real-time by the individual agents. Temporal Difference Q-learning is a tractable, approximate method that bootstraps on a previous estimate and can effectively adjust to dynamic effects, such as change in customer demand or traffic. Our work develops a hybrid learning architecture that is compatible with a TNC architecture by using distributed temporal difference learning and infrequent centralized Bellman solution updates.

Fig. 1. Concept graphic of an intelligent transportation network. Agents, both ground and air vehicles, estimate in real-time the state of the environment and select the customer requests to maximize the transportation company's profit.

Literature Review: Planning such an on-demand transportation system that must adapt to customer needs in real-time has been well-studied in the literature. The problem of providing transportation services for customers can be modeled as a Pick-up and Delivery Problem [5] or its extension Dial-A-Ride Problem [6] in which the transportation of goods is replaced by the transportation of people. Older prior work in the literature is focused on a static routing problem, where all the customers' requests for all time are known before routes are determined. However, due to the fast dynamics of customers requests and unknown future requests, employing these methods for planning real-time transportation in urban areas is not possible.

Recently, research has been conducted on dynamic and stochastic routing, where part or all of their input is unknown and revealed dynamically [7], [8], [9], [10]. The main objective of these studies is to minimize the total distance traveled by

the vehicle while servicing all customers. More sophisticated formulations [11], [12], are often not applicable to large-scale taxi fleets due to their centralized nature. Recent works also propose scalable solutions to dynamic routing problems. In [13], a decentralized solution is presented to minimize traffic congestion using an ant-pheromone-inspired method. In [14], a distributed deep reinforcement-learning method is proposed to learn macro-actions in event-response. Some model-based MPC approaches include [15], where a fleet of autonomous taxis is controlled in a scalable manner to minimize total energy cost and [16], where hierarchical control is used for large scale MPC. A multi-agent reinforcement learning survey paper discusses some additional methods in [17]. All these methods used a conventional agent-based Lagrangian approach, whereas our method uses an abstracted cell-based or Eulerian method that is inherently scalable with a large number of agents, similar to the idea used in probabilistic swarm guidance [18].

Because of this abstraction, our algorithm requires an additional task assignment component to directly interface taxis with customers and to coordinate taxis. The well-known Hungarian method [19], [20], [21], auction-based methods [22], and parallel algorithms [23], and their applications in multi-robot target and task assignment [24], [25], [26] can be employed to solve our problem formulation. However, these algorithms are mainly designed to solve the assignment problem in a centralized manner. Some decentralized methods have been introduced in the literature to tackle this problem [27], [28], [29], [30]. In [27], a distributed auction-based algorithm is introduced, where the task assignment problem is solved in a distributed manner. In [30], the consensus algorithm is employed to find the centralized solution in a distributed manner. However, by using this approach, the size of the problem is not reduced, and only the requirement for having a central node is relaxed. As a result, the algorithm for a large number of customers and agents can become intractable. Our method employs a prescriptive game theoretic approach that is compatible with a distributed framework. Using this method, our computation time per agent does not increase with the number of agents.

In parallel with the advances in operational routing research discussed above, the machine learning community has developed results in convergence of value iteration [31], [32], [33] useful for theoretical analysis of MDP frameworks. Applicable to our problem, there exists work characterizing the value function error due to the dynamics of a changing environment using $(\epsilon, \delta) - \mathrm{MDP}$ theory [32]. Current advances in machine learning address the computational tractability limit associated with the curse of dimensionality for solving MDP's with a large state, control, and temporal space. This effect drives appeal for (i) function approximate methods such as deep neural networks and, in the case of multi-agent systems, (ii) distributed solutions that use only local information. Our method addresses the need for distributed learning by using distributed control theory to estimate the optimal policy.

Main Contributions: The overview of our algorithm is shown in Fig. 2. First, we collect customer taxi data offline, and use it to formulate an MDP and solve using Modified Policy Iteration (MPI) to initialize an optimal Bellman policy. Then, we pass the optimal policy to each agent, where they each perform incremental TD online updates to capture environment dynamics and uncertainty. The agents perform distributed Q-learning using only local information and interactions to update the evolving policy. They assign customer requests cooperatively in a distributed game-theoretic task assignment, which completes the time step. If the policy update condition is met, which essentially dictates how far the agents' computed policy has drifted from the optimal policy, then the agents pass their data to the centralized node, which recomputes the MDP and corresponding MPI solution and globally re-initializes the agent's optimal policy estimates.

We borrow relevant techniques from fields of machine learning, distributed estimation, and prescriptive game theory. Our distributed reinforcement learning framework employs a one-step TD update, and allows us to derive analytical results using dynamic consensus methods developed in [34], [35], [36], and nonlinear stability theory using contraction [37], [38]. Additionally, we use results from prescriptive game theory [39], [40], [41], to design a potential game and iteration law to make agent action sets converge to near-optimal Nash equillibria. As compared to our preliminary work presented in the workshop article [42], this paper includes major revisions in all the sections, including a reformulation of the MDP, an update condition to iterate between exact and approximate solutions, additional mathematically-rigorous proofs of convergence, additional simulation results, and more complete proofs of the theorems.

The main contributions of the paper are as follows:

- We present a novel agent-agnostic MDP formulation for optimal routing. In contrast with conventional agent-based methods, we decompose the program into a cell-based Markov Decision Process and we coordinate individual agents with a game theoretic task assignment.
- We recast temporal difference (TD) Q-learning equations into a conventional distributed estimation theoretic framework, thereby allowing us to apply the distributed Kalman Information Filter to solve for an optimal adaptive learning rate.
- We rigorously bound the error in distributed policy estimation using tracking and synchronization analysis to the TD solution. We also bound the difference between the TD solution and the Bellman solution, allowing us to derive the error of the complete Bellman solution from the distributed estimation. This regret-bound-like result permits the user of the algorithm to explicitly tune the trade-off between computational effort and performance.
- We validate our results in a GridWorld simulation, and using real data from the city of Chicago taxi data set.

The rest of this paper is organized as follows: Section II presents notation and mathematical preliminaries. Section III presents our distributed estimation formulation, convergence results, and the exact solution update condition. Section III-C reviews the game theoretic formulation. Section IV presents the simulation results. Concluding remarks are made in Sec. V.

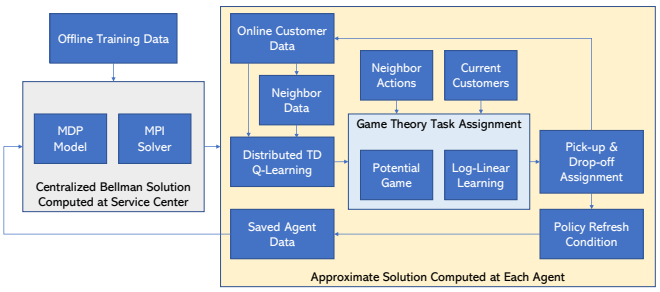

Fig. 2. Schematic of the Distributed Learning-Based Routing algorithm.

TABLE I
HELPER FUNCTIONS

Function	Name	Computation
cta	Customer to action	$(\text{pts}(x_p^k, y_p^k), \text{pts}(x_d^k, y_d^k))$
stp	State to position	Geometric centroid of map cell
tor	Time of request	Customer Data
ttc	Time to complete	Customer Data
tts	Time to start	$\text{tts}(s, c^k) = \|\text{stp}(s) - (x_p^k, y_p^k)\|/C_2$
pts	Position to state	Cell satisfying point-in-polygon condition

II. MATHEMATICAL PRELIMINARIES

A. Notation and Definition

Unless otherwise noted, we adopt the following conventions and notations. We define objects: $\mathbf{1}_n = [1, ..., 1]^\intercal/\sqrt{n}$ and I_n is the identity matrix in $\mathbb{R}^{n\times n}$ where n is an arbitrary dimension. We define an augmented state with $[.]$. For example, $[X] = X \otimes I_n$, where $\otimes$ denotes the Kronecker product. We denote the probability of event s with $\mathbb{P}(s)$. We denote the distribution of normal random variables with $\mathcal{N}(\mu, \Sigma)$, where μ is the random variable mean and Σ is the covariance matrix. Underbar denotes total infinum, $\underline{\alpha} = \inf_{t,i,(s,a)} \alpha_t^i(s,a)$ with $0 < \underline{\alpha} \leq \alpha < 1$. Unless otherwise noted, δ denotes bounds and σ denotes singular values that define contraction or convergence rates. We use the 2-norm. We denote block diagonal matrices with $\text{blkdiag}(.)$. The time index is denoted by t subscript, the customer request index is denoted by a k superscript, the superscript i refers to a quantity particular to an agent.

B. Markov Decision Process (MDP) Formulation

We use a sample-based MDP formulation, where a sample is a customer request, defined as a tuple:

$$c^k = \langle x_p^k, y_p^k, x_d^k, y_d^k, \text{fare}^k, \text{ttc}^k, \text{tor}^k \rangle$$

for $k = 1, .., N_c$, where $(x_p^k, y_p^k), (x_d^k, y_d^k) \in X \subset \mathbb{R}^2$ are the pickup and dropoff coordinates of the request, $\text{fare}^k \in \mathbb{R}$ is the fare of the request, $\text{tor}^k \in \mathbb{R}$ is the time of the request, and N_c is the number of customer requests. We use this tuple of data because this is the format given by the city of Chicago city data [43]. Although the position data here is in $\mathbb{R}^2$, the developed method is applicable to cells distributed in $\mathbb{R}^3$, relevant for future transportation systems including flying taxis as shown in Fig, 1.

In order to define the cell-based MDP tuple $\langle S, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, we need to abstract the sampled data with helper functions. We summarize descriptions for the helper functions in Table I, where the design constant C_2 is the average speed of taxis. We define the MDP as follows:

- S is the set of states, where a state is defined as a cell index of a discretized map. The states are discretized according to their spatial coordinates, $(x, y) \in X \subset \mathbb{R}^2$.
- $\mathcal{A}$ is the set of actions, $a \in \mathcal{A}$, where an action defined as moving to a pick-up state, then moving to drop-off state. We can parameterize the action as $a = (s_p, s_d)$
- $\mathcal{T}$ is the transition probability function. $\mathcal{T} : S \times \mathcal{A} \times S \to \mathbb{P}$, $\mathcal{T}(s, a, s') = \mathbb{P}(s'|s, a)$. For our application, we have a deterministic transition, $\mathcal{T}(s, a, s') \in \{0, 1\}$. Multiplication by the transition probability function extracts the quantity at the next state following the given action.
- $f(s, a)$ is the transition function that maps a state-action pair to the next state $s' = f(s, a)$ so that $f : S \times \mathcal{A} \to S$, where s' corresponds to the drop-off state of a customer request, which can be any state in S. We use this function for the definition of $\mathcal{T}$:

$$\mathcal{T}(s, a, s') = \begin{cases} 1 & f(s, a) = s' \\ 0 & \text{else} \end{cases}$$

- $\mathcal{R}(s, a)$ is the reward function returning the profit of a state-action pair: $\mathcal{R} : S \times \mathcal{A} \to \mathbb{R}$. For a single customer request, we can calculate a sample reward instance:

$$r^k(s, c^k) = \text{fare}^k - C_1 \text{tts}(s, c^k) - C_1 \text{ttc}^k \qquad (1)$$

where the constant design parameter C_1 is the cost per time. For example, in the Chicago city taxi simulation, we set C_1 to be 1 cent per second. We derive the MDP reward, $\mathcal{R}_t$ at time t by batch averaging the data set of previous customer requests reward information, with a geometric penalty to discount outdated information:

$$\mathcal{R}_t(s, a) = \frac{\sum_{k=1}^{N_c} \lambda^{\text{tor}^k - t} \mathcal{I}(\text{cta}(c^k) = a) r^k(s, c^k)}{\sum_{k=1}^{N_c} \lambda^{\text{tor}^k - t} \mathcal{I}(\text{cta}(c^k) = a)}$$

where $\mathcal{I}$ is the indicator function, λ is a design parameter, and we only consider samples taken before time t. We chose the reward to be the TNC's company profit, but this choice is arbitrary and we could use the conventional reward function that minimizes customer waiting time.
- γ is the discount factor and $0 < \gamma < 1$.

Inherently, the sample-based MDP will become the true optimal policy as the number of samples goes to infinity. At a given time, t, we can calculate the exact solution of the MDP as a set of optimal policies, π_t^* that maps from state to the optimal action using a modified policy iteration (MPI). In this paper, we consider the solution of the sampled-MDP as our optimal baseline that we attempt to estimate, because we define our base MDP with sampled information.

Note that we do not adopt a conventional multi-agent MDP framework for a vehicle routing problem with stochastic service requests, as suggested in [44]. Instead, our MDP is a framework is agnostic to the number of agents, and the value function it provides simply captures the value of a cell in the map for a given customer dataset. In effect, the computational complexity scales with the number of cells in the environment, not the number of agents. Our MDP formulation is inspired from an fluid mechanics 'Eulerian' perspective, [18], compared with the conventional agent-based 'Lagrangian' MDP formulation. The action space of the MDP, $\mathcal{A}$, is an abstraction of the taxi's real behavior to pickup and dropoff customers. In Sec. III-C, we complement this approach by hierarchically constructing an agent-based game that directly interfaces customers with taxis.

C. Q-learning Formulation

Reinforcement learning describes a set of methods that can be used to solve a general problem of simultaneously finding near-optimal solutions while exploring an unknown environment. In the urban taxi dynamic assignment problem, the reward model, $\mathcal{R}(s,a)$ is unknown and dynamically affected by the revenue, fare^k, which is driven by customer demand, and by the time to complete task, ttc, which is driven by traffic hours. Both of these values are unknown and time-varying, and we estimate this information by sampling actions and making measurements in the environment. In this paper, we alternate between two Q-learning methods: Bellman iteration and temporal difference (TD). In both cases, the reinforcement learning algorithms are attempting to estimate and maximize the value function:

$$V(s) = \max_a (\mathcal{R}(s,a) + \gamma \sum_{s'} \mathcal{T}(s,a,s')V(s'))$$

First, we present the standard Bellman Q-iteration [45], denoted by Q^*. We capture the net effect of customer demand and traffic through a time-varying optimal Q value, defined by the Bellman optimality equation, and we simplify using the deterministic transition function:

$$Q_t^*(s,a) = \mathcal{R}_t(s,a) + \gamma \max_{a' \in \mathcal{A}} Q_t^*(f(s,a),a')$$

where $Q_t^* : S \times \mathcal{A} \to \mathbb{R}$ and we take the maximum over actions a'. We apply a tabular Q value setting, where elements correspond to state action pairs, and we flatten the matrix into a vector such that $Q_t^* \in \mathbb{R}^{n_q}$ where $n_q = |S||\mathcal{A}| = |S|^3$ and $|S|$ is the number of cells. Additionally, we take the environment to be dynamic, so we add a subscript t to the Q_t^* and $\mathcal{R}_t$ vector. We denote the Q_t^* vector with no argument, and an element of Q_t^* vector with a state-action argument. In vector form, we can write the Bellman Q-iteration as:

$$Q_t^* = \mathcal{R}_t + \gamma \mathcal{T} Q_t^* \tag{2}$$

where $\mathcal{T}$ is the MDP transition function.

Next, we discuss the model-free, temporal difference (TD), one-step Q-learning [45], denoted here Q_t^{**}. We receive information via sampled actions from taking customer requests. Then we can calculate an instance of the reward function, $r_t^k = \mathcal{R}_t + v_t$ where $v_t \sim \mathcal{N}(0, \Sigma_t)$ is the measurement noise, and estimate Q_t^{**} with the following conventional TD method:

$$Q_{t+1}^{**} = (I - \alpha_t^{**} H_t^{**}) Q_t^{**} + \alpha_t^{**} H_t^{**} (\mathcal{R}_t + v_t + \gamma \mathcal{T} Q_t^{**}) \tag{3}$$

where $\alpha_t^{**} \in \mathbb{R}^{n_q \times n_q}$ is the learning rate and H_t^{**} is the measurement model for the available measurements. We define diagonal matrix $H_t^{**} \in \mathbb{R}^{n_q \times n_q}$ to be the average measurement model of all agents, if the information was perfectly centralized:

$$\begin{aligned} H_t^{**} &= [\mathbf{1}]^\intercal [H]_t [\mathbf{1}] \\ [H]_t &= \mathrm{blkdiag}(H_t^1, ..., H_t^{n_i}) \\ H_t^i(j,j) &= \begin{cases} 1 & \text{if } \ M_t(i,j) = 1 \\ 0 & \text{else} \end{cases} \end{aligned} \tag{4}$$

where n_i is the number of agents, $M_t \in \mathbb{R}^{n_i \times n_q}$ is an activation function obeying $M_t(i,j) = 1$ if and only if agent i has a measurement at state-action index j at time step t. Similarly, we define the TD-learning rate as:

$$\begin{aligned} \alpha_t^{**} &= [\mathbf{1}]^\intercal [\alpha]_t [\mathbf{1}] \\ [\alpha]_t &= \mathrm{blkdiag}(\alpha_t^1, ..., \alpha_t^{n_i}) \end{aligned} \tag{5}$$

where $[\mathbf{1}] \in \mathbb{R}^{n_i n_q \times n_q}$, $\alpha_t^i \in \mathbb{R}^{n_q \times n_q}$ and $[\alpha]_t \in \mathbb{R}^{n_i n_q \times n_i n_q}$, and we define the individual agent learning rates, α_t^i in Section III-B.

We have added the measurement model to the conventional Bellman and TD equations and written in vector form. In our framework, each agent estimates the optimal policy, $\hat{Q}_t^i$ by solving (2) with a Modified Policy Iteration (MPI) and initializing to the Bellman value, $\hat{Q}_0^i = Q_0^*$. Then, we account for the changing environment by updating the $\hat{Q}_t^i$ value online with a low-computational cost, incremental temporal difference (3) at each timestep at each agent. The Bellman method computes with a batch process, and it is computationally expensive thus cannot be evaluated at each timestep by each agent. The TD learning technique works naturally with incremental data and is especially effective because we start from an optimal policy, so we are 'bootstrapping' onto an already close-to-optimal policy. In our formulation, we decentralize the TD learning and compute a distributed estimation at each agent.

D. Assumptions

We make the following assumptions about our system:

Assumption 1. *The communication graph of taxis is connected at all times, i.e., $\lambda_2(L) > 0$, where λ_2 denotes the second eigenvalue and L is the communication graph Laplacian matrix (mathematically defined in Sec. III).*

Assumption 2. *The communication graph of taxis is balanced, i.e., $\mathbf{1}^\intercal L = \mathbf{0}$*

Assumption 3. *The dynamics of the environment is bounded such that there exists a constant that bounds the rate of change of the policy:*

$$\exists \delta_{\mathrm{e}} \ \text{ s.t. } \ \delta_{\mathrm{e}} = \sup_t \|\mathcal{R}_t + \gamma \mathcal{T} \hat{Q}_t^i - \hat{Q}_t^i\|$$

Assumption 4. *Every state action pair, (s, a) is sampled by at least one agent every $\boldsymbol{T}$ timesteps.*

Assumption 1 is appropriate for an urban environment because, with a large spatial density of taxis, the communication graph is guaranteed to stay connected. Assumption 2 is trivially satisfied by our use of an undirected communication graph. Assumption 3 is the difference between the measurement and current state, and this assumption is bounding the change in the environment in time, which implies there are no shocks in the reward and transition model, which is a proper assumption as no terms in the reward model can go unbounded. Assumption 4 is reminiscent of an adaptive control Collective Sufficient Richness requirement [46], where the reference signal needs to have enough information to converge on a parameter estimation. This assumption is appropriate for system with a large ratio of the number of agents (and samples) to number of states, and with a large exploration/exploitation trade-off value.

III. Main Results Hybrid Distributed Learning

In the first subsection, we present a solution to this problem where the control/estimation is all processed through a central node, corresponding to the conventional TD result. Then, we present a distributed formulation extension and the distributed performance bounds, with respect to TD and Bellman optimal policy.

A. Centralized TD-Learning

We consider the state of the system as augmented position and Q-values, and perform online estimation techniques on the Q-values, while assuming full knowledge of self state, s. By formulating the Q-learning iteration as a linear dynamical system, we apply the Kalman Information Filter (KIF) or its multi-agent form [34] to optimally estimate within one-step TD iteration. Formulating the Q-learning iterations (2) (3) with conventional control theory, and assuming zero-mean Gaussian noise, we arrive at the following canonical equations for the estimator and measurement:

$$Q^{**}_{t+1} = Q^{**}_t + \alpha^{**}_t(z^{**}_t - H^{**}_t Q^{**}_t) \tag{6}$$

$$z^{**}_t = H^{**}_t(\mathcal{R}_t + \gamma \mathcal{T} Q^{**}_t + v_t) \tag{7}$$

where α^{**}_t is the learning rate for each state-action pair, $\alpha^{**}_t \in \mathbb{R}^{n_q \times n_q}$, z^{**}_t is the pseudo-measurement update vector, as the direct measurements are the set of $\mathcal{R}_t + v_t$ that we augment with the previous estimation. The measurement noise is denoted by $v_t \sim \mathcal{N}(0, \Sigma_t) \in \mathbb{R}^{n_q}$, H^{**}_t is the measurement model, and Σ_t is the measurement noise covariance matrix, upper bounded by ςI. We model the TD equations (6) and (7) directly from (3).

B. Distributed Online Learning: State Sharing

The centralized formulation in the previous section guarantees optimal estimation of Q values for a single TD step. However, in considering computational and hardware limitations of the autonomous fleet, it is important to develop distributed algorithms that only use local information. The governing equations for the distributed system are similar, except we add a superscript i that denotes the equations are solved in a distributed fashion at each agent, $i = 1, ..., n_i$.

The conventional Distributed KIF [34] uses a measurement-sharing scheme where a dynamic consensus algorithm is used to average measurements and covariance matrices after being transformed by the information filter change of basis. We consider another form of the distributed Kalman filter with a state-sharing scheme (i.e., agent i shares $\hat{Q}^i_t$) with the dynamic consensus outside of the innovation step of the Kalman filter.

$$\hat{Q}^i_t = \hat{Q}^i_{t-1} + \alpha^i_t(z^i_t - H^i_t \hat{Q}^i_{t-1}) + \sum_j A_t(i,j)(\hat{Q}^j_{t-1} - \hat{Q}^i_{t-1}) \tag{8}$$

$$z^i_t = H^i_t(\mathcal{R}_t + \gamma \mathcal{T} \hat{Q}^i_t + v_t) \tag{9}$$

where H^i_t is defined as in the centralized case, except activation function $M_t(i,j)$ is evaluated only at the index of the agent performing the update and A_t is the time-varying, doubly stochastic adjacency matrix, defined in [34]. We stack the dynamics as follows for our analysis:

$$[\hat{Q}]_t = (I - [\alpha]_t[H]_t - [L]_t)[\hat{Q}]_{t-1} + [\alpha]_t[z]_t \tag{10}$$

where the dimension of the equation is $\in \mathbb{R}^{n_i n_q \times 1}$ and $[\hat{Q}]_t = [\hat{Q}^1_t; ...; \hat{Q}^{n_i}_t]^\intercal$, $[\alpha]_t = \text{diag}[\alpha^1_t, ..., \alpha^{n_i}_t] \in \mathbb{R}^{n_i n_q \times n_i n_q}$, $[L]_t \in \mathbb{R}^{n_i n_q \times n_i n_q}$. Here, $L_t \in \mathbb{R}^{n_i \times n_i}$ is the time-varying Laplacian matrix of the graph, defined as $L_t = I - A_t$. Additionally we augment this matrix using the kronecker product: $[L]_t = L_t \otimes I_{n_q}$. For the measurement model, $[H]_t$, the stacked model is a diagonal matrix of each of the agents measurements models, i.e., $[H]_t = \text{blkdiag}[H^1_t, ..., H^{n_i}_t]$. Note that these measurement models are switching, heterogeneous, and time-varying, and the flexible nature of contraction analysis allows us to deal with it naturally. Finally, the measurement is stacked as $[z]_t = [z^1_t; ...; z^{n_i}_t]^\intercal$, and note that:

$$[z]_t = [H]_t([\mathbf{1}]\mathcal{R}_t + [\mathbf{1}]v_t + \gamma[\mathbf{1}]\mathcal{T}[\mathbf{1}]^\intercal[\hat{Q}_{t-1}]) \tag{11}$$

Additionally, we can model the effect of the changing environment on the Bellman optimal solution as process noise.

$$Q^*_{t+1} = Q^*_t + w_t$$

where w is a normal random variable representing the process noise that corresponds to the changing environment, $w_t \sim \mathcal{N}(0, \mathcal{W}_t) \in \mathbb{R}^{n_q}$. With knowledge of the changing MDP model, we can tune the process noise covariance, $\mathcal{W}_t$, to match the dynamics of the MDP. For example, in the transition to rush hour times, the MDP-dynamics are fast, and we can increase the process noise such that the agents have a more aggressive learning rate and weigh their measurements more.

In this form, we can employ the Kalman filter or KIF [34] to optimally choose the learning rate, α^i_t to estimate Q^*_t.

$$\alpha^i_t = P^i_{t|t-1}(H^i_t)^\intercal(S^i_t)^{-1} \tag{12}$$

where $P^i_{t|t-1}$ and S^i_t are defined in the Appendix, Sec. B. At each time step, each agent, i, receives a measurement for their customer service, i.e., fare, ttc, which we then transform into a pseudo-measurement update vector for each agent,

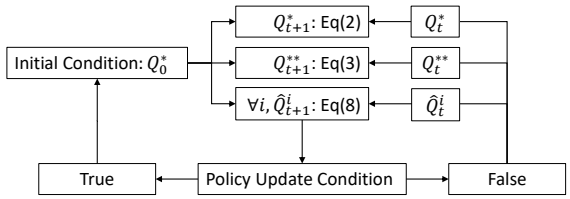

Fig. 3. Summary Block Diagram for different Q policy dynamics

z_t^i, using (9) to evaluate the KIF. We fuse the local agents state outside the measurement update. It is a known result from control theory that the optimal learning rate for this system is the Kalman gain (see the definition of KIF in Appendix, Sec. B) [35], [36]. The Kalman gain depends on the measurement covariance, Σ_t, process covariance, $\mathcal{W}_t$, and each agent's propagated error covariance. By re-casting Q-learning iteration equations into a conventional distributed estimation theoretic framework, we can apply the Kalman Information Filter and solve the Kalman Gain as an optimal adaptive learning rate. In the machine learning community, developing an optimal adaptive learning rate is an active area of research. To summarize the different policy evolutions, we refer to Fig. 3.

The performance bounds are presented here: in Theorems 1, 2, and 3, we compare agent's distributed policy estimation against the TD-optimal policy, Q_t^{**}. In Theorem 4, we compare the two learning methods: TD and Bellman, which lets us present the main result, Theorem 5, comparing the agent's distributed policy estimation to the Bellman solution, Q_t^*. Our analysis builds on the Discrete Gronwall Lemma, Theorem 6, included in Appendix for completeness, and Theorem 1 of [47] for discrete stochastic contraction. We use this result for a robust discrete contraction result, where discrete contraction defines the largest singular value of the Jacobian as the rate of contraction of the system ('C' in the Gronwall lemma) and the Gronwall lemma defines the bound of the sequence.

Theorem 1. *The state $[\hat{Q}]_t$ exponentially converges to the synchronization manifold, $[V_s]^\intercal[\hat{Q}]_t = 0$, at contraction rate σ_s, within a bound, δ_s, implying each agent's estimate converges to the synchronized value within the same error bound.*

$$\sigma_\mathrm{s} \triangleq 1 - \inf_{t'} \lambda_2(L_{t'})$$
$$\delta_\mathrm{s} \triangleq \frac{n_i \delta_\mathrm{e}}{\inf_{t'} \lambda_2(L_{t'})}$$
$$\|\hat{Q}_t^i - [\mathbf{1}]^\intercal \hat{Q}_t\| \leq \delta_\mathrm{s}(1 - \sigma_\mathrm{s}^t)$$

where the synchronization manifold, $[V_s]^\intercal[\hat{Q}]_t = 0$, is defined by augmenting the eigenbasis of the Laplacian matrix, L_t, where $L_t V_t = D_t V_t$, $V = [\mathbf{1}, V_s]$ and D_t is a diagonal matrix of eigenvalues.

Proof: We transform (10) by the eigenbasis of the augmented Laplacian: $[V] : [L][V] = [\Lambda][V]$, $[V] = [[\mathbf{1}], [V_s]]$:

$$\begin{bmatrix} I - [\mathbf{1}]^\intercal[\alpha]_t[H]_t[\mathbf{1}] & -[\mathbf{1}]^\intercal[\alpha]_t[H]_t[V_s] \\ -[V_s]^\intercal[\alpha]_t[H]_t[\mathbf{1}] & [V_s]^\intercal(I - [\alpha]_t[H]_t - [L]_t)[V_s] \end{bmatrix} \cdot \begin{bmatrix} [\mathbf{1}]^\intercal[\hat{Q}]_{t-1} \\ [V_s]^\intercal[\hat{Q}]_{t-1} \end{bmatrix} + \begin{bmatrix} [\mathbf{1}]^\intercal[\alpha]_t[z]_t \\ [V_s]^\intercal[\alpha]_t[z]_t \end{bmatrix} = \begin{bmatrix} [\mathbf{1}]^\intercal[\hat{Q}]_t \\ [V_s]^\intercal[\hat{Q}]_t \end{bmatrix}$$

where $[\mathbf{1}]^\intercal[L]_t = 0$ by the balanced graph assumption (Assumption 2), $[L]_t[\mathbf{1}] = 0$ by definition, and $[\mathbf{1}]^\intercal[V_s] = [V_s]^\intercal[\mathbf{1}] = 0$ by orthogonality of the Laplacian eigenbasis. Consider the state projected in the $[V_s]$ directions:

$$[V_s]^\intercal[\hat{Q}]_t = [V_s]^\intercal(I - [\alpha]_t[H]_t - [L]_t)[V_s][V_s]^\intercal[\hat{Q}]_{t-1} - [V_s]^\intercal[\alpha]_t[H]_t[\mathbf{1}][\mathbf{1}]^\intercal[\hat{Q}]_t + [V_s]^\intercal[\alpha]_t[z]_t$$

We apply the orthogonality identity, $[\mathbf{1}][\mathbf{1}]^\intercal + [V_s][V_s]^\intercal = I$, to reduce to:

$$[V_s]^\intercal[\hat{Q}]_t = [V_s]^\intercal(I - [L]_t)[V_s][V_s]^\intercal[\hat{Q}]_{t-1} + [V_s]^\intercal[\alpha]_t([z]_t - [H]_t[\hat{Q}]_{t-1})$$

We apply partial contraction results and consider virtual system for y:

$$y_t = [V_s]^\intercal(I - [L]_t)[V_s]y_{t-1} + d_t$$
$$d_t = [V_s]^\intercal[\alpha]_t([z]_t - [H]_t[\hat{Q}]_{t-1})$$

The unperturbed system, $d_t = 0$, admits solutions $y_t = [V_s]^\intercal\hat{Q}_t$ and $y = 0$, so a contracting system will imply exact, global exponential convergence to the synchronization manifold, within a bound determined by the disturbance d. The Jacobian is given as: $[V_s]^\intercal(I - [L]_t)[V_s]$. Note that $[V_s]^\intercal[L][V_s] = [D_s]$, where D_s is a positive definite matrix of eigenvalues of L, where the smallest eigenvalue of L is nonzero by Assumption 1.

$$\sigma_1 \leq 1 - \inf_{t'} \lambda_2(L_{t'}) \triangleq \sigma_s$$

where $\lambda_2(L)$ is the Fiedler value of the communication graph. We reduce the disturbance by Assumption 3, apply Discrete Gronwall, Theorem 6, and remove the decaying initial condition as the agents are initialized with the same policy:

$$\|[V_s]^\intercal\hat{Q}_t\| \leq \frac{n_i\delta_\mathrm{e}}{\inf_{t'} \lambda_2(L_{t'})}(1 - \sigma_s^t)$$

Additionally, note that the bound trivially holds for a single agent, and is less tight:

$$\|\hat{Q}_t^i - [\mathbf{1}]^\intercal\hat{Q}_t\| \leq \|\hat{Q}_t - [\mathbf{1}][\mathbf{1}]^\intercal\hat{Q}_t\| = \|[V_s][V_s]^\intercal\hat{Q}_t\| \leq \|[V_s]^\intercal\hat{Q}_t\|$$

■

Theorem 2. *The agent's synchronized value, $[\mathbf{1}]^\intercal\hat{Q}_t$ corresponds exactly to the TD-optimal policy, Q_t^{**} for all time t.*

Proof: Projecting the stacked dynamics (10) into the centroid directions:

$$[\mathbf{1}]^\intercal[\hat{Q}]_t = [\mathbf{1}]^\intercal(I - [\alpha]_t[H]_t)[\mathbf{1}][\mathbf{1}]^\intercal[\hat{Q}]_{t-1} + [\mathbf{1}]^\intercal[\alpha]_t[z]_t$$

where $[L]_t[\mathbf{1}] = 0$ by definition of Laplacian. Subtracting the TD-optimal dynamics (3) and applying definitions of TD-measurement model (4) and TD-learning rate (5):

$$\begin{aligned}
&([\mathbf{1}]^\intercal[\hat{Q}]_t - Q_t^{**}) \\
&\quad = (I - [\mathbf{1}]^\intercal[\alpha]_t[H]_t[\mathbf{1}])[\mathbf{1}]^\intercal[\hat{Q}]_{t-1} + [\mathbf{1}]^\intercal[\alpha]_t[z]_t \\
&\qquad - (I - \alpha_t^{**}H_t^{**})Q_{t-1}^{**} - \alpha_t^{**}z_t^{**} \\
&\quad = (I - [\mathbf{1}]^\intercal[\alpha]_t[H]_t[\mathbf{1}])([\mathbf{1}]^\intercal[\hat{Q}]_{t-1} - Q_{t-1}^{**}) \\
&\qquad + [\mathbf{1}]^\intercal[\alpha]_t([z]_t - [\mathbf{1}]z_t^{**}) \qquad (13)
\end{aligned}$$

Next we apply definitions of stacked updates (11) and TD-update (7), with (4):

$$\begin{aligned}
[z]_t - [\mathbf{1}]z_t^{**} &= [H]_t([\mathbf{1}]\mathcal{R}_t + [\mathbf{1}]v_t + \gamma[\mathbf{1}]\mathcal{T}[\mathbf{1}]^\intercal[\hat{Q}]_{t-1}) \\
&\quad - [H]_t[\mathbf{1}](\mathcal{R}_t + v_t + \gamma\mathcal{T}Q_{t-1}^{**}) \\
&= [H]_t[\mathbf{1}]\gamma\mathcal{T}([\hat{Q}]_{t-1} - Q_{t-1}^{**})
\end{aligned}$$

Plugging back into dynamics equation (13):

$$\begin{aligned}
&([\mathbf{1}]^\intercal[\hat{Q}]_t - Q_t^{**}) \\
&\quad = (I - [\mathbf{1}]^\intercal[\alpha]_t[H]_t[\mathbf{1}](I - \gamma\mathcal{T})([\mathbf{1}]^\intercal[\hat{Q}]_{t-1} - Q_{t-1}^{**})
\end{aligned}$$

which is a semi-contracting system with no disturbance, strictly contracting every $\boldsymbol{T}$ timesteps by Assumption 4. It is a semi-contracting system because there exist unity eigenvalues of the Jacobian until the matrix $[\mathbf{1}]^\intercal[\alpha]_t[H]_t[\mathbf{1}]$ is full rank, which is satisfied every $\boldsymbol{T}$ timesteps, at which point the eigenvalues are strictly less than one and the system is contracting. However, as the trajectories have the same initial condition, we can conclude the synchronized estimation state is exactly the TD-optimal. ∎

Theorem 3. *The estimate of each agent, $\hat{Q}_t^i$, is within a bound, δ_s of the TD-optimal solution, Q_t^{**}:*

$$\|\hat{Q}_t^i - Q_t^{**}\|] \leq \delta_{\mathrm{s}}(1 - \sigma_{\mathrm{s}}^t)$$

Proof: In Theorem 2, we showed the equivalence of Q_t^{**} and $[\mathbf{1}]^\intercal\hat{Q}_t$, and in Theorem 1, we derived the difference between $\hat{Q}_t^i$ and $[\mathbf{1}]^\intercal\hat{Q}_t$. The desired result follows as the same bound. ∎

Next, we present the hybrid reinforcement learning solution, where we compare the TD solution with the Bellman solution.

Theorem 4. *The distance between the TD-optimal solution, Q_t^{**} and the Bellman-optimal solution, Q_t^* evolves as:*

$$\mathrm{E}\|Q_t^{**} - Q_t^*\| \leq \frac{2\sqrt{n_q\varsigma}}{\underline{\alpha}(1-\gamma)} \triangleq \delta_{\mathrm{h}}$$

Proof: Subtracting the definition of the Bellman policy (2) from the the definition of the TD policy (3):

$$\begin{aligned}
Q_{t+1}^{**} - Q_{t+1}^* &= (I - \alpha_t^{**}H_t^{**})(Q_t^{**} - \mathcal{R}_t) \\
&\quad - \gamma\mathcal{T}Q_t^* + \alpha_t^{**}H_t^{**}\gamma\mathcal{T}Q_t^{**} + \alpha_t^{**}H_t^{**}v_t
\end{aligned}$$

Adding and subtracting $\alpha_t H_t^{**}\gamma\mathcal{T}Q_t^*$, and applying (2):

$$\begin{aligned}
Q_{t+1}^{**} - Q_{t+1}^* &= (I - \alpha_t^{**}H_t^{**})(Q_t^{**} - (\mathcal{R}_t + \gamma\mathcal{T}Q_t^*)) \\
&\quad + \alpha_t^{**}H_t^{**}\gamma(Q_t^{**} - Q_t^*) + \alpha_t^{**}H_t^{**}v_t \\
Q_{t+1}^{**} - Q_{t+1}^* &= (I - (1-\gamma)\alpha_t^{**}H_t^{**})(Q_t^{**} - Q_t^*) + \alpha_t^{**}H_t^{**}v_t
\end{aligned}$$

We have a semi-contracting system for $y_t = Q_t^{**} - Q_t^*$ that is strictly contracting every $\boldsymbol{T}$ timesteps from Assumption 4. Applying Theorem 1 of [47] with the identity matrix metric and noting the two trajectories have the same initial condition:

$$\begin{aligned}
\sigma_{\mathrm{h}} &\triangleq 1 + \underline{\alpha}\gamma - \underline{\alpha} \\
C &= n_q\varsigma \geq \mathrm{tr}(H_t^{**}\alpha_t^{**}\alpha_t^{**}H_t^{**}\Sigma_t) \\
\mathrm{E}\|Q_t^{**} - Q_t^*\| &\leq \frac{2\sqrt{C}}{1-\sigma_{\mathrm{h}}} = \frac{2\sqrt{n_q\varsigma}}{\underline{\alpha}(1-\gamma)}
\end{aligned}$$

where α_t^{**} and H_t^{**} are symmetric because they are diagonal and recall that ςI upper bounds the measurement covariance matrix. ∎

Finally, we can combine all the results for the main result of the paper:

Theorem 5. *The distance between the agent's estimate, $\hat{Q}^i$, and the Bellman solution Q_t^* evolves as:*

$$\begin{aligned}
\mathrm{E}\|\hat{Q}_t^i - Q_t^*\| &\leq \delta_{\mathrm{s}}(1 - \sigma_{\mathrm{s}}^t) + \delta_{\mathrm{h}} \\
&= \frac{n_i\delta_{\mathrm{e}}}{\inf_{t'}\lambda_2(L_{t'})}(1 - (1 - \inf_{t'}\lambda_2(L_{t'}))^t) + \frac{2\sqrt{n_q\varsigma}}{\underline{\alpha}(1-\gamma)}
\end{aligned}$$

Proof: This result follows a similar reasoning to Theorem 3. The distance between the distributed agent estimation and the TD solution is bounded by $\delta_{\mathrm{s}}(1 - \sigma_{\mathrm{s}})^t$ and the difference between the TD solution and the Bellman solution is bounded by δ_{h}, implying the desired final bound. ∎

This result leads us to a policy update condition that guarantees a desired policy accuracy: compute $\delta_{\mathrm{error}} = \delta_{\mathrm{s}}(1 - \sigma_{\mathrm{s}})^t + \delta_{\mathrm{h}}$, and when it exceeds the design constant, $\delta_{\mathrm{desired}}$, request a policy update from the central node.

C. Game Theoretic Task Assignment

The purpose of this section is to interface the optimal policy estimation of cell value information with agent-based taxi behavior. In other words, we need to map between the 'Eulerian' MDP to a 'Lagrangian' coordination strategy. We propose a distributed assignment game for taxis to coordinate and choose from an available customer requests. First, we define important elements of the game.

Let $i, j \in \mathcal{I}$ be taxis in the set of all taxis, and x_t^i, y_t^i be the coordinates of the taxi at time t. Let the kth customer request $c^k \in C$ be the tuple: $c^k = \langle x_p^k, y_p^k, x_d^k, y_d^k, \mathrm{fare}^k, \mathrm{ttc}^k, \mathrm{tor}^k\rangle$. At each turn, each taxi, i, has an available customer set, C_t^i, $C_t^i = \{c^k \in C |\ \|(x_p^k, y_p^k) - (x_t^i, y_t^i)\|_2 \leq R_u\}$, where R_u is the maximum radius at which agents can sense customers. At each turn, each taxi has an set of neighbors, $\mathcal{N}_t^i$, $\mathcal{N}_t^i = \{j \in \mathcal{I} |\ \|(x_t^i, y_t^i) - (x_t^j, y_t^j)\|_2 \leq R_c\}$ where R_c is the maximum radius at which taxis can communicate with other taxis. Finally, we define a global action set of all the taxi's current actions: $U_t = \{c_t^{k,i} | \forall i = 1, ..., n_i\}$, where n_i is the number of agents and $c_t^{k,i}$ denotes agent i serving customer request k at time t.

Now we can define the global potential function of our game, which is the company's estimated long-term profit.

$$\Phi_t(U_t) = \sum_{i=1, k \notin \mathcal{K}_t}^{i=n_i} \hat{Q}_t^i(s_t^i, \text{cta}(c_t^{k,i}))$$

where $\mathcal{K}_t$ is the set of already serviced customers, this is included to avoid double counting servicing the same customer. Next, we isolate a single taxi's action to study the marginal benefit of an action. For individual taxi, i, we can define the global null action set $U_t^{i,0}$, $U_t^{i,0} = \{c_t^{k,j} \in U_t | \forall j \neq i\}$, where $c_t^{k,i,0}$ denotes an element in $U_t^{i,0}$. Intuitively, the null action set is the set of all taxi's current actions, omitting the action of the taxi being considered. We can define a function, J_t^i, that captures an action's marginal contribution (or utility):

$$J_t^i(U_t, U_t^{i,0}) = \begin{cases} \hat{Q}_t^i(s_t^i, \text{cta}(c_t^{k,i})) & k \notin \mathcal{K}_t^i \\ 0 & \text{else} \end{cases}$$

By checking over the neighboring customer serviced set, $\mathcal{K}_t^i$ instead of all the agents, the J^i function is local because the elements of the global action set U_t that are non-local (outside neighboring set) are inactive. By setting $R_u = R_c/2$, we guarantee that any set of taxi's that can sense the same customers will also be neighbors.

Next, we show that J_t^i is indeed the marginal contribution on global potential function, Φ_t. For any individual taxi, i, we define the global alternative action set $\tilde{U}_t^i = \{c_t^{k,j} \in U_t | \forall j \neq i\} \bigcup \tilde{c}_t^{k,i}, \tilde{c}_t^{k,i} \in U_t^i$. Intuitively, this is the null action set for taxi i, with the addition of an alternative action $\tilde{u}_t^i$ that is in i's action set. It is easy to see that the change in the potential function for any action is captured by the change in the marginal utility:

$$\Phi_t(\tilde{U}_t^i) - \Phi_t(U_t) = J_t^i(\tilde{U}_t^i, U_t^{i,0}) - J_t^i(U_t, U_t^{i,0})$$

This assignment game is non-cooperative, so taxi's actions will converge to a Nash-equilibria action set, U_t^*, ie: $\forall i, \forall U_t, J_t^i(U_t^*, U_t^{i,0}) \geq J_t^i(U_t, U_t^{i,0})$. At the same time, there might be multiple Nash-equilibria action sets, U_t^* and we want to pick one that maximizes the potential function, Φ_t. We use a game-theoretic reinforcement learning technique, binary log-linear learning, [41] to iterate actions to better Nash Equilibrium. At each time step, t, the action set is randomly initialized, then, while all other agent's actions are held, a single agent, i, chooses between the previously held action, $c_t^{k,i}$, and an alternate action $\tilde{c}_t^{k,i}$. In accordance with the binary log-linear learning formulation, the agent will remain at action $c_t^{k,i}$ with probability $P_t^i(U_t, \tilde{U}_t^i)$, and change action with probability $1 - P_t^i(U_t, \tilde{U}_t^i)$:

$$P_t^i(U_t, \tilde{U}_t^i) = \frac{\exp\left(J_t^i(U_t, U_t^{i,0})/\tau\right)}{\exp\left(J_t^i(U_t, U_t^{i,0})/\tau\right) + \exp\left(J_t^i(\tilde{U}_t^i, U_t^{i,0})/\tau\right)} \tag{14}$$

The coefficient τ, $\tau \in \mathbb{R}_{>0}$, is a design parameter specifying how likely agent i chooses a sub-optimal action, to specify the trade-off between exploration and exploitation. For $\tau \to \infty$ the formulation will choose the action sets U_t^i and $\tilde{U}_t^i$ with equal probability (pure exploration). For $\tau \to 0$, the formulation will choose the action set that has the greatest utility function (pure exploitation). The action set is chosen once the iteration has converged, which completes the game-theoretic task assignment.

IV. Simulation Results

The algorithm overview is presented in Algorithm 1. The following section includes simulation results and discussion.

Algorithm 1 Distributed learning-based routing algorithm.

```
agents.Q = BellmanMPI(Training Data)
while true do
    if δ_error > δ_desired then
        agents.Q = BellmanMPI(agents.Data)
    end if
    for ∀ agent ∈ agents do                ▷ Local Information
        agent.requests = LocalCurrentRequests
        agent.neighbors = LocalAgents
    end for
    for ∀ agent ∈ agents do                ▷ Task Assignment
        agent.action = Game(agent.neighbors), Eq. (14)
    end for
    for ∀ agent ∈ agents do                ▷ Learning Update
        agent.update = ...
        UpdateEquation(agent.action), Eq. (9)
        agent.Q = ...
        D-TD(agent.update, agent.neighbors), Eq. (8)
    end for
    moveAgents(agents)
end while
```

A. GridWorld Simulations

The GridWorld simulations are used to testing our algorithm in a clean environment prior to testing on real datasets. First, the state-space representation for illustration is shown in Fig. 4, where the dotted blue circle represents the radius of tasks available to each agent.

Next, we present estimation results, shown in Fig. 5, comparing the pure distributed, distributed with update, and centralized estimations for the Q-values for each state-action pair. We compute the error with respect to the optimal MPI solution computed at each time step. As designed, our algorithm better estimates the MPI Q-values solution compared with a pure distributed solution. For the distributed algorithms, we can also see the how the estimates do not exceed a certain bound from the average value, and this is a visualization of the synchronization bound result from Theorem 1.

Next, we discuss the performance of the algorithms in Figs. 6 and 7, where each data point is a simulation. On the y-axis, the performance ratio, J/J_{MPI} is plotted, where J is the total reward over the simulation, $J = \sum_t \sum_i r_t^i$ and J_{MPI} is a normalization that corresponds to the pure MPI algorithm, which should outperform in all cases. Note that this J is different from the marginal utility function defined in Sec. III-C.

In Fig. 6, on the x-axis, the number of time steps in the simulation is shown to indicate that the optimal policy drift of

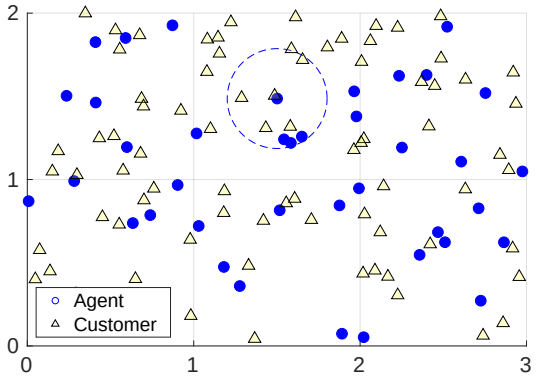

Fig. 4. GridWorld: state space example.

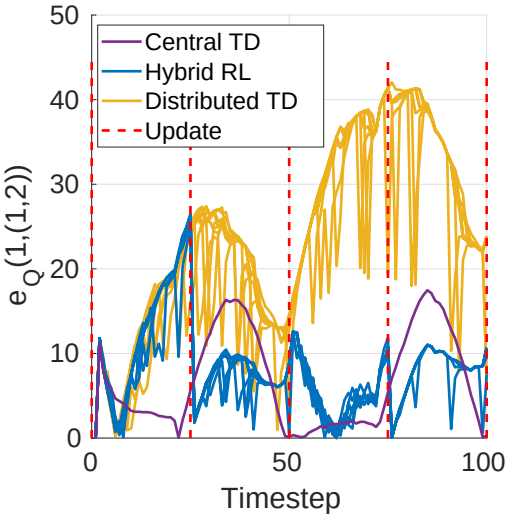

Fig. 5. GridWorld Q-value error trace with respect to Bellman solution for a single state-action pair.

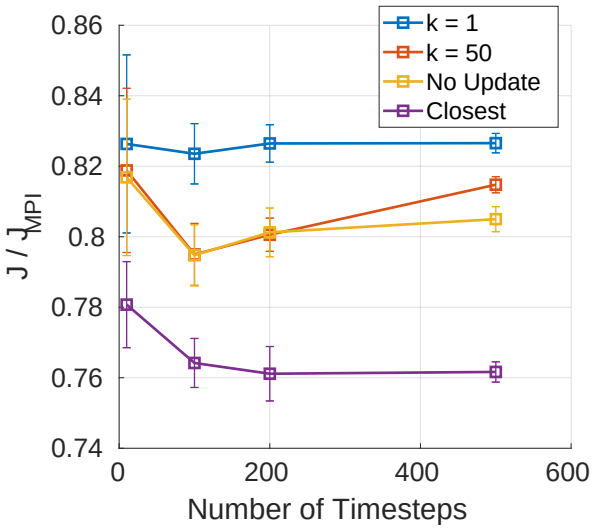

Fig. 6. GridWorld: performance vs. number of time steps for 80 Agents.

the Q-learning algorithms eventually causes the agents to make bad decisions resulting in low reward. We plot for different values of k, where k is the number of iterations between MPI iterations. The $k = 1$ case represents a completely MPI solution, and as k approaches the number of time steps, the algorithm becomes completely distributed temporal difference Q-learning. We also plot a baseline 'Closest' algorithm, where the agents do not do any distributed estimation, they simply pick the closest customer request to themselves. We identify the trends we expect: our hybrid algorithm outperforms the 'No update' case, (which corresponds to the conventional distributed estimator, Distributed Kalman Filtering), and the performance increases inversely with k, corresponding to a better estimate of the optimal policy. Note that the $k = 1$ case is not unity, because we are still normalizing with respect to a centralized MPI solution where each agent has access to all actions, whereas the distributed solution uses only local available actions. Note that in the 'No update' case, the optimal policy drift causes the solution to degrade with length of simulation.

In Fig. 7, the performance of the algorithms for different numbers of agents is shown. We see that as the number of agents increases, the performance ratio of the different algorithms approaches unity. Note that we are able to simulate planning problems with a large number of agents because we formulated a cell-based MDP, whereas solving an agent-based MDP would be computationally intractable. We find that the proposed algorithm runtime per agent per time-step is approximately constant while the number of agents increases, see Table III, where we attribute the slight negative correlation to the computational overhead in each simulation. We ran the simulations with an Intel i7-8700 processor and 15G of RAM. We use the design parameters specified in Table II.

TABLE II
DESIGN PARAMETERS

Variable Name	GridWorld Value	Chicago Sim Value
C_1	1	0.01 USD/s
C_2	1	0.0001 deg/s
δ_{desired}	N/A	10870
τ	1	1
λ	0.8	0.8
$\amalg_t$	1	1
ς_t	1	1
γ	0.8	0.8
$\|S\|$	6	20
R_c	0.4	3500m

B. Chicago City Taxi Simulations

In this section, we explore the practical applicability of our algorithm by running our simulation using the Chicago city taxi dataset [43] for customer requests. The state space of the Chicago map is shown in Fig 8. As a benchmark for our algorithm, we use the same 'closest' baseline as in the GridWorld simulations. We compute an offline solution using the final hour of April 31, 2017, and our online testing set is over the 24 hours of May 1, 2017 for a total of 17,165

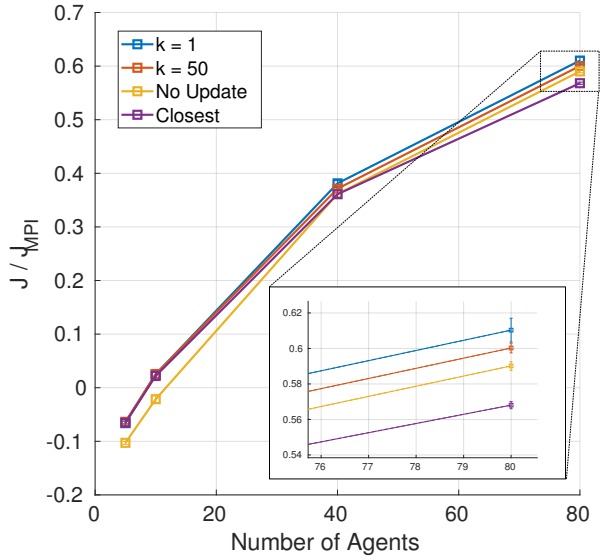

Fig. 7. GridWorld: performance vs. number of agents for 500 time steps.

TABLE III
RUNTIME PER AGENT (SEC)

	$n_i = 5$	$n_i = 10$	$n_i = 40$	$n_i = 80$
$k = 1$	2.9152	2.5735	2.2096	2.0015
$k = 50$	1.1110	0.9230	0.8868	0.8708
No Update	1.0952	0.9449	0.8640	0.8361

customer requests. We assign a fleet of 100 taxis, and a update condition of $\delta_{\text{desired}} = 10870$, which corresponds to roughly 10 percent error per Q-value. We choose the design constants, C_1 to be 1 cent per second and C_2 to be approximately 10 meters per second, which corresponds to 0.0001 degrees latitude/longitude per second. The plot of cumulative rewards is shown in Fig. 10, and it can be seen that our algorithm outperforms the closest and distributed with no update baselines. In total, our distributed algorithm outperformed the 'closest' solution by 17.2 percent, which in this case corresponds to 58 USD per taxi per day. It is interesting to note that the slope of the rewards: slow in the early hours of the morning (0 AM–5 AM), then picks up strongly in midday (10 AM–3 PM) and stabilizes to approximately linear for the rest of the day.

It is important to note the spatial distribution of requests of this dataset, presented in Fig. 9. We can see the cluster of customer requests in the spatial area surrounding near north-side Chicago. The other trend of note is the isolated cluster of customer requests in the far north-east, that we attribute to the Chicago O'Hare airport traffic. When taxi's service there, the bi-modal spatial distribution of requests causes the taxi's graph to be disconnected, instantly triggering a Bellman update request by the result from Theorem 5.

V. CONCLUSION

In this paper, we presented a hybrid distributed reinforcement learning mechanism to solve the urban ride-sharing problem in uncertain and dynamic environments. By formulating

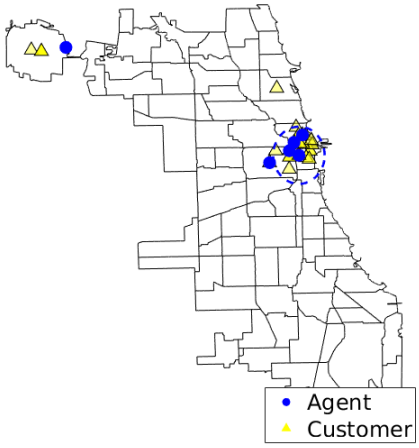

Fig. 8. Chicago city state space. Note the bimodal distribution of customer requests at the O-Hare International Airport (northwest), and in the downtown (East).

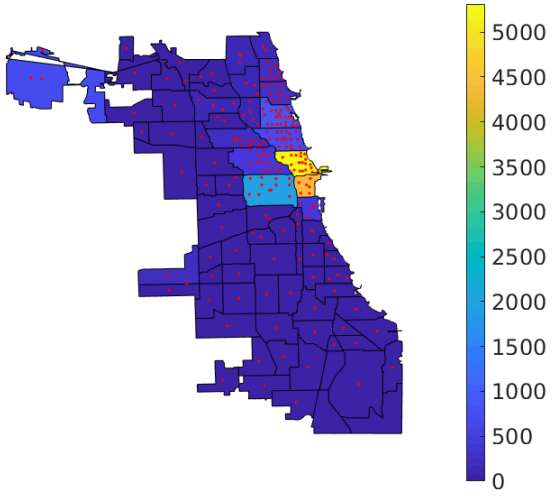

Fig. 9. Chicago city customer request heatmap where the red dots represent customer requests.

an 'Eulerian' MDP and distributed reinforcement learning update law, we are able to estimate optimal policy online in a framework that is highly scalable with the number of agents. By exploiting the theoretical intersection of distributed TD Q-learning, Bellman iteration, and distributed estimation analysis from control theory, we are able to provide guarantees on the performance of this method. This allows us to introduce a design constant that characterizes the trade-off of computational expense and estimation accuracy by switching between heavy computations at a central node and light distributed computations at each taxi. We validate our method in a GridWorld environment and using taxi data from the city of Chicago. In the GridWorld environment, we confirmed our theory and expectations by comparing Q-learning and MPI solutions, in centralized and distributed settings. On the real

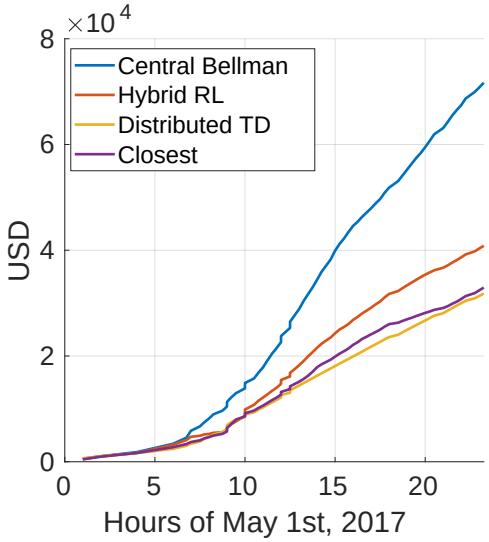

Fig. 10. Chicago simulation cumulative rewards for May 1, 2017 for 100 taxis.

dataset from Chicago city taxis, we found that the proposed algorithm outperforms a closest algorithm by about 17 percent. Although our solution is inherently scalable with the number of agents, this formulation is limited with the number of cells in the discretized state-space. Thus, we propose future work in exploring map segmentation, and deep learning methods to reduce the dimensionality. Another future direction could be to explore the conventional reward to minimize customer waitng time. We chose to formulate a TNC profit reward function for simplicity and from limitations in the available dataset, but our theoretical framework is compatible with arbitrary reward functions. Finally, we compared our distributed algorithm to a locally greedy algorithm that maximizes reward at each time step. We found that, in some cases, this greedy version would outperform the proposed algorithm, and understanding the performance difference is an ongoing investigation.

APPENDIX

A. Contraction Theorems for Discrete Systems

We present two existing results supporting the proofs of the main theorems.

Theorem 6. *(Discrete Gronwall Lemma [48]) Let a positive sequence* $\{Z_n\}$ *satisfy:*

$$Z_{n+1} \leq CZ_n + D \quad \forall n$$

for some constants C, D*, with* $C > 0$*. Then:*

$$Z_n \leq \frac{D}{1-C}(1-C^n) + Z_0 C^n \quad \forall n$$

when $C \neq 1$.

B. Kalman Information Filter

We include the Kalman Information Filter (KIF) form for completeness [49]. Consider the following dynamical systems:

$$x_t = F_t x_{t-1} + B_t u_t + w_t$$
$$z_t = H_t x_t + v_t$$

where x_t is the hidden state, z_t is the measurement, F_t is the dynamical model, H is the measurement model, B_t is the input matrix, u_t is the input, w_t is the process noise, $w_t \sim N(0, \mathcal{W}_t)$ and v_t is the measurement noise, $v_t \sim N(0, R_t)$. Given these equations, the optimal estimator, $\hat{x}_t$ is given by a prediction and measurement step, that we will write in the notation of the paper. The prediction step is given by:

$$\hat{Q}^i_{t|t-1} = \hat{Q}^i_{t-1|t-1}$$
$$P^i_{t|t-1} = P^i_{t-1|t-1} + \mathcal{W}_t$$

The measurement step of the information filter is given by defining an information vector, $\hat{y}^i_t$ and information matrix, Y^i_t:

$$Y^i_{t|t} = (P^i_{t|t})^{-1}$$
$$\hat{y}^i_{t|t} = (P^i_{t|t})^{-1}\hat{Q}^i_{t|t}$$

where measurements z^i_t taken at time step t can be transformed as:

$$Y^i_{t|t} = Y^i_{t|t-1} + (H^i_t)^\intercal R_t^{-1} H^i_t$$
$$\hat{y}^i_{t|t} = \hat{y}^i_{t|t-1} + (H^i_t)^\intercal R_t^{-1} z^i_t$$

Finally, the Kalman gain, α^i_t is calculated as follows:

$$S^i_t = H^i_t P^i_{t|t-1}(H^i_t)^\intercal + R_t$$
$$\alpha^i_t = P^i_{t|t-1}(H^i_t)^\intercal (S^i_t)^{-1}$$

ACKNOWLEDGMENT

The authors thank the feedback from colleagues in the Data-driven Intelligent Transportation workshop (DIT 2018, held in conjunction with IEEE ICDM). The work is funded in part by the Raytheon Company. The authors would also like to thank Wolfgang Hoenig and Anthony Fragoso for their helpful feedback and comments.

REFERENCES

[1] Google-Waymo. (2016, December) Google self-driving car. Https://waymo.com/.
[2] J. Holden and N. Goel. (2016, October) Uber elevate: Fast-forwarding to a future of on-demand urban air transportation. Https://www.uber.com/elevate.pdf.
[3] R. Bellman, "A Markovian decision process," *J. Mathematics and Mechanics*, pp. 679–684, 1957.
[4] Puterman and Shin, "Modified policy iteration algorithms for discounted markov decision problems," *Management Science*, vol. 24, no. 11, 1978.
[5] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte, "Static pickup and delivery problems: a classification scheme and survey," *TOP*, vol. 15, no. 1, pp. 1–31, 2007.
[6] J.-F. Cordeau and G. Laporte, "The dial-a-ride problem (DARP): Variants, modeling issues and algorithms," *Quarterly J. Belgian, French and Italian Operations Research Societies*, vol. 1, no. 2, pp. 89–101, 2003.
[7] L. M. Hvattum, A. Løkketangen, and G. Laporte, "Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic," *Transportation Science*, vol. 40, no. 4, pp. 421–438, 2006.

[8] K. Treleaven, M. Pavone, and E. Frazzoli, "Asymptotically optimal algorithms for one-to-one pickup and delivery problems with applications to transportation systems," *IEEE Trans. Autom. Control*, vol. 58, no. 9, pp. 2261–2276, 2013.

[9] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European J. Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.

[10] G. Kim, Y. S. Ong, and T. Cheong, "Solving the dynamic vehicle routing problem under traffic congestion," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2367–2380, 2016.

[11] Q. Lu and M. Dessouky, "An exact algorithm for the multiple vehicle pickup and delivery problem," *Transportation Science*, vol. 38, no. 4, pp. 503–514, 2004.

[12] Seongmoon Kim, M. E. Lewis, and C. C. White, "Optimal vehicle routing with real-time traffic information," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 2, pp. 178–188, 2005.

[13] R. Claes, T. Holvoet, and D. Weyns, "A decentralized approach for anticipatory vehicle routing using delegate multiagent systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 364–373, 2011.

[14] K. Menda, Y.-C. Chen, and J. Grana, "Deep reinforcement learning for event-driven multi-agent decision processes," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, pp. 1–10, 2017.

[15] R. Luo, T. J. van den Boom, and B. De Schutter, "Multi-agent dynamic routing of a fleet of cybercars," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 5, pp. 1340–1352, 2018.

[16] Z. Zhou, B. D. Schutter, S. Lin, and Y. Xi, "Two-level hierarchical model-based predictive control for large-scale urban traffic networks," *IEEE Trans. Contr. Sys. Techn.*, vol. 25, no. 2, pp. 496–508, 2017.

[17] L. Busoniu, R. Babuska, and B. D. Schutter, "Multi-agent reinforcement learning: A survey," in *Ninth International Conference on Control, Automation, Robotics and Vision, ICARCV 2006, Singapore, 5-8 December 2006, Proceedings*, 2006, pp. 1–6.

[18] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and distributed control of a large-scale swarm of autonomous agents," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1103–1123, 2017.

[19] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[20] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal of the ACM*, vol. 19, no. 2, pp. 248–264, 1972.

[21] J. Yu, S.-J. Chung, and P. G. Voulgaris, "Target assignment in robotic networks: Distance optimality guarantees and hierarchical strategies," *IEEE Transactions on Automatic Control*, vol. 60, no. 2, pp. 327–341, 2014.

[22] D. P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," *Annals of Operations Research*, vol. 14, no. 1, pp. 105–123, 1988.

[23] D. P. Bertsekas and D. A. Castañon, "Parallel synchronous and asynchronous implementations of the auction algorithm," *Parallel Comput.*, vol. 17, no. 6-7, pp. 707–732, 1991.

[24] C. Schumacher, P. R. Chandler, and S. R. Rasmussen, "Task allocation for wide area search munitions," in *Proc. the 2002 American Control Conf.*, vol. 3, 2002, pp. 1917–1922.

[25] Y. Jin, A. A. Minai, and M. M. Polycarpou, "Cooperative real-time search and task allocation in UAV teams," in *42nd IEEE Int. Conf. Decision Control*, vol. 1, 2003, pp. 7–12.

[26] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-task allocation and path planning for cooperating UAVs," in *Cooperative Control: Models, Applications and Algorithms*, S. Butenko, R. Murphey, and P. M. Pardalos, Eds. Boston, MA: Springer US, 2003, pp. 23–41.

[27] D. Morgan, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, "Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming," *Int. J. Robotics Research*, vol. 35, no. 10, pp. 1261–1285, 2016.

[28] D. Dionne and C. A. Rabbath, "Multi-UAV decentralized task allocation with intermittent communications: the DTC algorithm," in *2007 American Control Conf.*, 2007, pp. 5406–5411.

[29] P. B. Sujit and R. Beard, "Distributed sequential auctions for multiple UAV task allocation," in *2007 American Control Conf.*, 2007, pp. 3955–3960.

[30] H. L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, 2009.

[31] M. L. Littman and C. Szepesvári, "A generalized reinforcement-learning model: Convergence and applications," in *International Conference on Machine Learning (ICML)*, 1996.

[32] B. C. Csáji and L. Monostori, "Value function based reinforcement learning in changing markovian environments," *J. Mach. Learn. Res.*, vol. 9, pp. 1679–1709, 2008.

[33] C. Szepesvri and M. L. Littman, "A unified analysis of value-function-based reinforcement-learning algorithms," *Neural Computation*, vol. 11, no. 8, pp. 2017–2060, 1999.

[34] S. Bandyopadhyay and S.-J. Chung, "Distributed bayesian filtering using logarithmic opinion pool for dynamic sensor networks," *Automatica*, vol. 97, pp. 7–17, 2018.

[35] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[36] O. Saber, "Kalman-consensus filter : Optimality, stability, and performance," in *IEEE Conference on Decision and Control*, vol. 48, 2009.

[37] W. Lohmiller and J.-J. E. Slotine, "On contraction analysis for nonlinear systems," *Automatica*, vol. 34, pp. 683–696, 1998.

[38] W. Wang and J.-J. E. Slotine, "On partial contraction analysis for coupled nonlinear oscillators," *Biological Cybernetics*, vol. 92, no. 1, pp. 38–53, 2005.

[39] D. Fudenberg and D. K. Levine, *The Theory of Learning in Games*, ser. MIT Press Books. The MIT Press, 1998, vol. 1.

[40] W. H. Sandholm and H. P. Young, "Strategic learning and its limits," *Games and Economic Behavior*, vol. 63, no. 1, pp. 417–420, 2008.

[41] J. R. Marden and J. S. Shamma, "Revisiting log-linear learning: Asynchrony, completeness and payoff-based implementation," *Games and Economic Behavior*, vol. 75, no. 2, pp. 788 – 808, 2012.

[42] S. Rahili, B. Riviere, S. Oliver, and S.-J. Chung, "Optimal routing for autonomous taxis: Distributed reinforcement learning approach," in *Proc. 1st Workshop Data-driven Intell. Transp., IEEE ICDM*, 2018.

[43] "Chicago data portal," https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew.

[44] M. W. Ulmer, D. C. Mattfeld, and F. Köster, "Budgeting time for dynamic vehicle routing with stochastic customer requests," *Transportation Science*, vol. 52, no. 1, pp. 20–37, 2018.

[45] R. S. Sutton and A. G. Barto, *Reinforcement learning - an introduction*, ser. Adaptive computation and machine learning. MIT Press, 1998. [Online]. Available: http://www.worldcat.org/oclc/37293240

[46] P. M. Wensing and J. E. Slotine, "Cooperative adaptive control for cloud-based robotics," in *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, 2018, pp. 6401–6408.

[47] Q. Pham, "Analysis of discrete and hybrid stochastic systems by nonlinear contraction theory," in *10th International Conference on Control, Automation, Robotics and Vision, ICARCV 2008, Hanoi, Vietnam, 17-20 December 2008, Proceedings*, 2008, pp. 1054–1059.

[48] A. M. Stuart and T. Humphries, *Dynamical Systems and Numerical Analysis*, 1996.

[49] G. Chen, "Introduction to random signals and applied kalman filtering," *International Journal of Adaptive Control and Signal Processing*, vol. 6, pp. 173–175, 1992.

Benjamin Riviere is a PhD student at California Institute of Technology (Caltech). He received the B.S. in Mechanical Engineering from Stanford University in 2017 and the M.S. in Aeronautics from Caltech in 2018. His research interests include combining machine learning and network control theory with applications in transportation and space systems.

Salar Rahili received the B.S. and the M.Sc. degrees in electrical engineering from the Isfahan University of Technology, Isfahan, Iran, in 2009 and 2012, respectively. He received his Ph.D. degree in electrical engineering at the University of California, Riverside, CA, USA, in 2016 and joined the Aerospace Department of the California Institute of Technology, USA, as a postdoctoral researcher afterwards. His research focuses on distributed control of multi-agent systems, reinforcement learning, distributed optimization, and guidance and control of autonomous systems.

Soon-Jo Chung received the B.S. degree (summa cum laude) from Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 1998; the S.M. degree in aeronautics and astronautics; and the Sc.D. degree in estimation and control from Massachusetts Institute of Technology, Cambridge, MA, USA, in 2002 and 2007, respectively. He is the Bren Professor of Aerospace and Jet Propulsion Laboratory Research Scientist in the California Institute of Technology. Dr. Chung was on the faculty of the University of Illinois at Urbana-Champaign (UIUC) during 2009-2016. His research focuses on spacecraft and aerial swarms and autonomous aerospace systems, and in particular, on the theory and application of complex nonlinear dynamics, control, estimation, guidance, and navigation of autonomous space and air vehicles. Dr. Chung is received the UIUC Engineering Deans Award for Excellence in Research, the Beckman Faculty Fellowship of the UIUC Center for Advanced Study, the U.S. Air Force Office of Scientific Research Young Investigator Award, the National Science Foundation Faculty Early Career Development Award, and three Best Conference Paper Awards from the IEEE, and the American Institute of Aeronautics and Astronautics. He is an Associate Editor of IEEE Transactions on Robotics, IEEE Transactions on Automatic Control, and Journal of Guidance, Control, and Dynamics.