Optimal Routing for Autonomous Taxis using Distributed Reinforcement Learning

Salar Rahili, Benjamin Riviere, Suzanne Oliver, Soon-Jo Chung

***Abstract*—In this paper, a learning-based optimal transportation algorithm for autonomous taxis and ridesharing vehicles is introduced. The goal is to design a mechanism to solve the routing problem for a fleet of autonomous vehicles in real-time in order to maximize the transportation company's profit. To solve this problem, the system is modeled as a Markov Decision Process (MDP) using past customers data. By solving the defined MDP, a centralized high-level planning recommendation is obtained, where this offline solution is used as an initial value for the real-time learning. Then, a distributed SARSA reinforcement learning algorithm is proposed to capture the model errors and the environment changes, such as variations in customer distributions in each area, traffic, and fares, thereby providing an accurate model and optimal policies in real-time. Agents are using only their local information and interaction, such as current passenger requests and estimates of neighbors' tasks and their optimal actions, to obtain the optimal policies in a distributed fashion. The agents use the estimated values of each action, provided by distributed SARSA reinforcement learning, in a distributed game-theory based task assignment to select their conflict-free customers. Finally, the customers data provided by the city of Chicago is used to validate the proposed algorithms.**

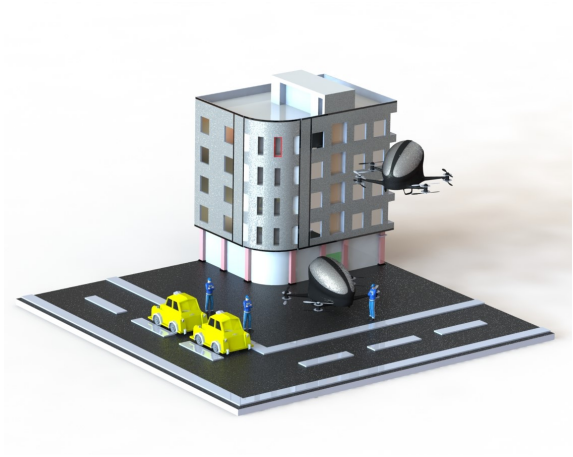

Fig. 1. Concept graphic of an intelligent transportation network. Agents estimate in real-time the state of the environment and select optimal customers to maximize the transportation company's profit.

I. Introduction

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 employ coordinated fleets of autonomous ground and air vehicles to improve the urban transportation capabilities [1], see Fig. I. The deployment of fleets of autonomous vehicles, both ground and air, drives a coupled innovation in algorithm development.

Developing optimal control algorithms for fleets of vehicles is an active topic in the literature. The problem of providing transportation services for customers can be modeled as a Pick-up and Delivery Problem (PDP) or its extension Dial-A-Ride Problem (DARP) in which the transportation of goods is replaced by the transportation of people. Most 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 [2], [3]. Especially in urban areas, the environment for an optimal MDP problem should not be considered known because of fast dynamic effects such as traffic and consumer demand, and this necessitates an dynamic routing solution. Recently, some new research has been conducted on dynamic and stochastic routing (PDPs), where part or all of their input is unknown and revealed dynamically [4]–[6], but they are not necessarily multi-agent or decentralized. The disadvantage of a centralized solution is that the computational expense does not scale favorably with number of agents and they are often intractable for a large fleet of agents. In [7], a multi-vehicle algorithm is studied, where the routing problem is solved only for 5 vehicles and $17-25$ customers due to computation complexity. Existing mechanisms of optimal transportation in the literature do not satisfy all requirements.

In this paper, we present a distributed learning-based optimal traffic planning and decision making algorithm that integrates planning with a local decision making algorithm. The proposed scheme performs in a distributed fashion based on local information. Such local information includes other vehicles' tasks, their estimate of the optimal actions, and current passenger requests. To this goal, each vehicle selects the best customer among current requests in order to maximize the company's profit in the long run. The past customers data is used to predict the probability of having customers, and their trips in each area. Solving the MDP, a high-level planning recommendations is provided for agents including a list of desire transitions and their values. This is shown in left half of Fig. 2. However, the static solution built only based on past data is not accurate, and it is not able to capture any changes in the environment. Reinforcement learning can be used as a decision making scheme when an accurate model of the environment is not known. In Sec. III-A, a State-Action-Reward-State-Action (SARSA) reinforcement learning algorithm is introduced which allows the system to learn its model (i.e., transition probabilities and rewards) and update the optimal policies while the optimal policies obtained in Sec. II-B are used as an initial value. The conventional reinforce-

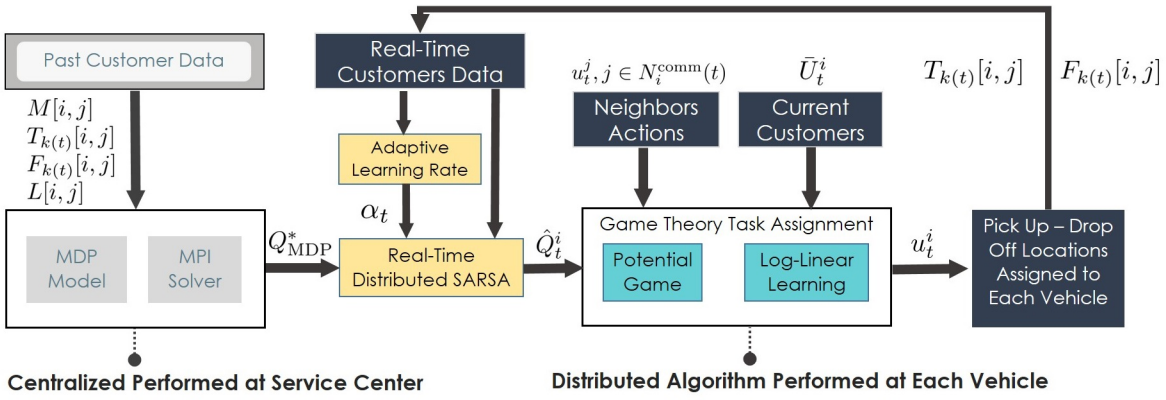

Fig. 2. Schematic of the proposed optimal routing algorithm.

ment learning usually is not well-suited for a non-stationary environment, and the commonly-used proofs of convergence hold only for stationary environments. In our problem setting, the environment is non-stationary. In particular, the number of the customers in each area, traffic and fares are changing over time. In Sec. III-B, we propose an optimal adaptive learning-rate tuning, and modify the SARSA reinforcement learning in order to track the environment changes in non-stationary environments.

Moreover, the SARSA algorithm is a centralized algorithm, where all information is required to be sent to a central node to be fused. However, in our framework with many vehicles and trips in each step, it is not feasible to pass all information to and from a command center. Hence, a fully distributed SARSA reinforcement learning is proposed in Sec. III-C, where agents are only using their own local information and local interactions to update the optimal policies of the system.

The proposed modified distributed SARSA provides the value of each action in the environment. Each vehicle uses these values to evaluate each customer. In a single agent scenario, the agent simply selects the customer with the largest value. However, in a multi-vehicle scenario, agents are required to reach an agreement on selected customers in order to avoid any conflict. To solve this problem, agents need to agree on how to distribute the customers among themselves. In Sec. IV, a real-time task assignment algorithm based on game theory is proposed to enable the agents to select their non-conflicting tasks/customers in a distributed fashion. Our formulation is completely distributed, providing scalability with the number of agents and a property of robustness to loss or gain of agents.

II. Preliminaries and Definitions

A. Notation

The following notations are adopted throughout this paper. $\mathbb{R}^n$ denotes the vector of size n. Let $\mathbf{1}_n$ and $\mathbf{0}_n$ denote the column vectors of n ones and zeros, respectively. The Cartesian product of sets, $S_i, \forall i$ is denoted by $\prod_{i=1}^{N} S_i$. The empty set is denoted by $\varnothing$. The cardinality of a set S is denoted by $|S|$. $\mathbb{E}[\cdot]$ is the expectation operator. The arithmetic mean of a series of numbers, a_i, for $i = 1, \cdots, n$, is denoted by $\text{Avg}(a_i) = \frac{1}{n}\sum_{i=1}^{i=n} a_i$. The exponential function is written as $\exp(\cdot)$. A time-varying digraph $\mathcal{G}_t \triangleq (V, E_t)$ is used to characterize the interaction topology among the agents, where $V \triangleq \{1, \ldots, N\}$ is the node set and $E_t \subseteq V \times V$ is the edge set. An edge $(i,j) \in E_t$ means that node i can obtain information from node j at time t. The *adjacency matrix* $\mathcal{A}_t \triangleq [a_{ij}(t)] \in \mathbb{R}^{N\times N}$ of the graph $\mathcal{G}_t$ is defined such that the edge weight $a_{ij} \neq 0$ if $(j,i) \in E_t$ and $a_{ij} = 0$ otherwise. The compact two-dimensional mission space (i.e., city map) is partitioned into $n_q \in \mathbb{N}$ disjoint partitions. The size of the cells is selected by the designer based on the desired spatial resolution and the computation expenses. In later sections with distributed algorithms, the superscript is by default referring to agent index, and the subscript is a time index.

B. Markov Decision Process Formulation

MDP is a mathematical framework introduced to help make decisions in a stochastic environments (see [8], [9] and references therein), and the solution of MDP is a policy providing all optimal actions in each state of our environment. Our MDP is formulated with a tuple, $< S, A, \mathcal{P}, \mathcal{R} >$ as follows:

- State variables S: The finite set of zones or cells in the city, denoted by $S = \{i \mid \forall i \in 1, 2, \cdots, n_q\}$.
- Actions A: The set of possible actions at cell state l is $A(l) = \{a_j^l\}$, where a_j^l is the action of moving into cell j from cell l.
- Reward model $\mathcal{R}$: The reward model is $\mathcal{R}_{a_j^l}(l,j) = \frac{F[l,j]M[l,j]}{T[l,j]}$, $\forall j : a_j^l \in A(l)$, where $F[l,j]$ corresponds to the fare to go from cell l to cell j , $M[l,j]$ corresponds to any motion constraints to go from cell l to cell j, and $T[l,j]$ corresponds to time to go from cell l to cell j
- Transition probabilities $\mathcal{P}$:

$$\mathcal{P}_{a_i^l}(l,i) = \begin{cases} L[i,j] & i \neq j \\ 1 + L[i,i] - \sum_j L[i,j] & i = j \end{cases}$$

where $L[i,j]$ be the probability of having a customer to pick up from cell i and deliver to cell j.

We formulate the maximum reward problem with a Q-value Q, a value function V, and an optimal policy π^* , defined with terms from the MDP tuple. This is a dynamic programming

problem, where $\mathcal{R}(i,j)$ is the immediate reward to go from cell$i$ to cell j, $V(j)$ is all the future reward if actions are chosen optimally from cell j, and γ is the discount factor that penalizes future rewards exponentially.

$$Q(i,\pi[i]) = \sum_{j\in n_q} \mathcal{P}_{\pi[i]}(i,j)\big(\mathcal{R}_{\pi[i]}(i,j) + \gamma V(j)\big), \quad (1)$$

$$V(j) = \max_{\pi[j]\in A(j)} Q(j,\pi[j]), \quad (2)$$

$$\pi^*[i] = \arg\max_{a_j^i \in A(i)} Q(i,a_j^i), \quad (3)$$

A solution to (3) is an optimal policy, denoted by $\pi^*[i]$, defined by the Bellman equation. Knowledge of the long-term value, $Q(.)$, for each state-action pair is an equivalent solution. To find the solution for (3), we use a Modified Policy Iteration (MPI) algorithm to estimate (2) through several steps of successive approximation. The optimal solution of the MDP problem is aggregated as a vector denoted by Q^*_{MDP}.

III. Distributed SARSA Reinforcement Learning with Adaptive Learning Rate Tuning

To account for the changing environment, we begin with the near-optimal Q^*_{MDP} and update the optimal policy online at each time step using new data from customers and agents' trips. State-Action-Reward Reinforcement Learning (SARSA RL) is used because it can obtain an optimal policy when the system's model (i.e., $\mathcal{P}$ and $\mathcal{R}$) is not known in advance. First, we present a centralized SARSA RL algorithm in Sec. III-A. Then, we present an optimal adaptive learning rate in Sec. III-B. Finally, we present a novel decentralized SARSA RL algorithm with a proof of convergence in Sec. III-B.

A. Centralized SARSA RL for Stationary Environment

First, we present a standard, centralized model-free reinforcement learning as a contextual comparison for our main contribution of deriving distributed algorithms. The Q-values with respect to state-action pairs are updated in a SARSA RL framework as:

$$Q_{t+1}(i,\pi[i]) = (1-\alpha_t)Q_t(i,\pi[i]) + \alpha_t\big(\mathcal{R}_{\pi[i]}(i,j) + \gamma Q_t(j,\pi[j])\big), \quad (4)$$

where α_t is a learning rate satisfying Remark 1, and $Q(\cdot)$ is updated under policy $\pi[i]$ to transition from cell i to cell j. This formulation means that the action $\pi[j]$ at the successor state j is not necessarily optimal, while in Q-learning, the successor action is chosen to be optimal.

Remark 1: In conventional reinforcement learning, the sequence of otherwise arbitrary α_t satisfies: $\sum_{T=0}^{\infty}\alpha_t = \infty$, and $\sum_{T=0}^{\infty}\alpha_t^2 < \infty$. The Q-values eventually converge to a constant as the update term goes to zero.

However, in a non-stationary environment, we want the adaptive learning rate to not converge to zero, such that Q can continue being updated in (4). In the next section, we will introduce a method to estimate the optimal adaptive learning rate dynamic signal.

B. Adaptive Learning-Rate for Non-stationary Environment

In this subsection, a new algorithm (shown in yellow color in Fig. 2) is derived to estimate the optimal learning rate, $\alpha_t(i,\pi[i])$, at each time t and for each action-state pair, to track the time-varying optimal Q-value, where the dynamics of Q-value captures the environment non-stationarity. Our mathematical framework is as follows.

Estimating the optimal policy is equivalent to converging the estimated Q-value to the Q-value at the next time-step. For each new sample data from a new customer with pair $(i,\pi[i])$ at time t, the Q-update and its stochastic information can be written as:

$$\begin{aligned} v_t(i,\pi[i]) &= \mathcal{R}_{\pi[i]}(i,j) + \gamma Q_t(j,\pi[j]),\\ \mathbb{E}(v_t) &= Q_t^*(i,\pi[i]),\\ \mathbb{V}\text{ar}(v_t) &= \sigma_t^2(i,\pi[i]) \end{aligned}$$

Thus, we define the loss function and the expected value of the loss function as follows:

$$\mathcal{L}(Q_t(i,\pi[i])) = \frac{1}{2}\bigg(Q_t(i,\pi[i]) - v_t(i,\pi[i])\bigg)^2, \quad (5)$$

$$\mathbb{E}\big[\mathcal{L}(Q_t(i,\pi[i]))\big] = \frac{1}{2}\bigg\{\bigg(Q_t(i,\pi[i]) - Q_t^*(i,\pi[i])\bigg)^2 + \sigma_t^2(i,\pi[i])\bigg\},$$

where $\mathbb{V}\text{ar}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$ is used in the last equality. The loss function is a distance between the estimated Q-value and the Q-value at the next time-step. According to conventional stochastic stability formulation, the Lyapunov function of the system is the expected value of the loss function.

Adopting the stochastic stability iteration framework from [10], our optimization problem is to choose $\alpha_t^*(i,\pi[i])$ to minimize the expectation of the Lyapunov function, conditioned on the value at the previous state:

$$\alpha_t^*(i,\pi[i]) = \arg\min_{\alpha_t(i,\pi[i])} \mathbb{E}\bigg[\mathbb{E}\big[\mathcal{L}\big(Q_{t+1}(i,\pi[i])\big)\big]|Q_t(i,\pi[i])\bigg] \quad (6)$$

Theorem 1: For SARSA RL (4), the optimal estimate of α_t minimizing the cost function (5) for each state-action pair is computed as a ratio of exponential moving averages as

$$\alpha_t^*(i,\pi[i]) = \frac{f_t^*(i,\pi[i])^2}{g_t^*(i,\pi[i])}, \quad (7)$$

$$\begin{aligned} f_{t+1}^*(i,\pi[i]) &= f_t^*(i,\pi[i]) + \zeta\varepsilon\Big(\frac{\partial\mathcal{L}}{\partial Q_t(i,\pi[i])} - f_t^*(i,\pi[i])\Big)\\ g_{t+1}^*(i,\pi[i]) &= g_t^*(i,\pi[i]) + \zeta\varepsilon\Big(\frac{\partial\mathcal{L}}{\partial Q_t(i,\pi[i])}^2 - g_t^*(i,\pi[i])\Big), \end{aligned} \quad (8)$$

where

$$\frac{\partial\mathcal{L}}{\partial Q_t(i,\pi[i])} = Q_t(i,\pi[i]) - \mathcal{R}_{\pi[i]}(i,j) - \gamma Q_t(j,\pi[j]) \quad (9)$$

with $\varepsilon = 1$, if there is a new update for action $\pi[i]$ in state i at time t, else $\varepsilon = 0$. ζ is a design parameter used for exponential

convergence, where $0 < \zeta < 1$. With this formulation, we can compute alpha with only Q-values and reward update information. We recover (9) from the definition of v_t and taking the gradient of (5).

Proof: The derivation of α_t^* is not included for space limit considerations. The outline of the proof is as follows: perform Stochastic Gradient Descent on (5), rewrite (6), perform an optimization with derivative method, numerically calculate expected value using with an exponential moving average of samples. ■

C. Distributed SARSA RL for Non-stationary Environment

In Sec. III-A and III-B, we introduced an adaptive SARSA algorithm for non-stationary environments. Here, we present a dynamic average tracking algorithm to estimate the time-varying Q-values in a distributed manner that is, by nature, scalable to a large number of agents, where, in the context of optimal routing, an agent is defined as a taxi. The mathematical overview is as follows. First, we present the proposed update rules with each agent i's structure. Second, we make assumptions on the system to present upper limit bounds on estimate errors. Third, we show convergence of a stochastic difference equation to prove that the estimated Q-values converge to the true values with bounded errors.

The update rule for agent i and the observation pair (j, a_j^l) is proposed as follows:

$$\begin{aligned}
&\hat{\mathbf{Q}}_{t+1}^i = \hat{\mathbf{Q}}_t^i + \sum_{k=1}^{k=N} a_{ik}\big(\hat{\mathbf{Q}}_t^k - \hat{\mathbf{Q}}_t^i\big) + \mathbf{r}_t^i,\\
&\mathbf{r}_t^i = N\mathbf{I}_t^i \alpha_t^i(j, a_j^l) r_t^i,\\
&r_t^i = \mathcal{R}_{a_j^l}(j, l) + \gamma \hat{Q}_t^i(l, \pi[l]) - \hat{Q}_t^i(j, a_j^l), \quad r_0^i = 0, \forall i,
\end{aligned} \tag{10}$$

where N is the number of agents, $[a_{ik}] \in \mathbb{R}^{N\times N}$ is the adjacency matrix of communication among agents defined in Sec. II and Assumption 2, and l is the successor state after conducting action a_j^l at state j. $\mathbf{I}_t^i$ is a vector with one non-zero entry corresponding to the state-action pair (j, a_j^l), unless agent i does not select an action, and $\mathbf{I}_t^i \in \mathbb{R}^{\sum_{k=1}^{n_q} |A(k)|}$.

The agent's structure is as follows. Each agent i maintains its estimate of Q-values for state action pairs at time, t, in vector $\hat{\mathbf{Q}}_t^i \in \mathbb{R}^{\sum_{k=1}^{n_q} |A(k)|}$. The agent i's estimate of the optimal learning-rate vector is obtained as:

$$\begin{aligned}
&\alpha_t^i(j, a_j^l) = \frac{\hat{f}_t^i(j, a_j^l)^2}{\hat{g}_t^i(j, a_j^l)}, \ \hat{\boldsymbol{f}}_t^i = \hat{\boldsymbol{f}}_{t-1}^i + \zeta(\boldsymbol{\omega}_{t-1} - \hat{\boldsymbol{f}}_{t-1}^i), \\
&\hat{\boldsymbol{g}}_t^i = \hat{\boldsymbol{g}}_{t-1}^i + \zeta(\boldsymbol{\omega}_{t-1}^2 - \hat{\boldsymbol{g}}_{t-1}^i),\\
&\boldsymbol{\omega}_t^i = \boldsymbol{\omega}_{t-1}^i + \sum_{k=1}^{k=N} a_{ik}(\boldsymbol{\omega}_{t-1}^i - \boldsymbol{\omega}_{t-1}^k) + N(\mathbf{I}_t^i r_t^i - \mathbf{I}_{t-1}^i r_{t-1}^i),
\end{aligned} \tag{11}$$

where $\hat{f}_t^i$, and $\hat{g}_t^i$ are the estimates of f_t^*, and g_t^*, defined in (8), respectively, and their aggregated vector forms are written as $\hat{\boldsymbol{f}}_t^i$ and $\hat{\boldsymbol{g}}_t^i$. Also, $\boldsymbol{\omega}_t^i$ is the estimate of the gradient of the loss function, $\frac{\partial \mathcal{L}}{\partial Q_t(j, a_j^l)}$, written in a vector form. The initial values are chosen as $\boldsymbol{\omega}_0^i = \mathbf{0}$, and $\hat{\boldsymbol{f}}_0^i = \hat{\boldsymbol{g}}_0^i = \mathbf{1}, \forall i$. The information updates available to agents are local customer requests data: state transitions of departure and arrival cells, fare, and travel time. The agents share their Q-value estimates with their neighbors. The algorithms in (10) and (11) are the distributed forms of (4) and (7), respectively. Using (10) and (11), each agent only requires local information and local interactions to update its values.

We make the following assumptions of our system:

Assumption 1: There exists a time-invariant constant $\Delta r_{\max}$, such that for all agents and all time, $\left\|r_{t+1}^i - r_t^i\right\| \leq \Delta r_{\max}$, where r_t^i is the Q-value update for agent i at time t.

Assumption 2: The digraph $\mathcal{G}_t \triangleq (V, E_t)$ with adjacency matrix $\mathcal{A}(t)$ satisfy the following conditions

(I) Periodic Strong Connectivity: There exists a positive integer $\mathbf{b}$, such that the digraph $\mathcal{G}_t \triangleq (V, E_t \cup E_{t+1} \cup \cdots \cup E_{t+\mathbf{b}-1})$ is strongly connected for all t.

(II) Non-degeneracy: There exists a constant $\gamma \in (0, 1)$ such that $a_{ij}(t) \in \{0\} \cup [\gamma, 1]$.

(III) Balanced Communication: The matrix $\mathcal{A}(t)$ is doubly stochastic for all t, i.e., $\mathbf{1}^T\mathcal{A}(t) = \mathbf{1}^T$, and $\mathcal{A}(t)\mathbf{1} = \mathbf{1}$.

Assumption 3: The observations' covariance of each state-action pair is time-invariant, i.e. $\sigma_{t+1}(i, \pi[i]) = \sigma_t(i, \pi[i]), \forall t$, and it will be written as $\sigma(i, \pi[i])$.

Theorem 2: Suppose that we have a stochastic difference equation

$$\begin{aligned}
Q_{t+1}^i(i, \pi[i]) = Q_t^i(i, \pi[i]) + [\alpha^*(t) + \epsilon_\alpha^k] \\
\times \big(\mathcal{R}_{\pi[i]}(i, j) + \gamma Q_t^k(j, \pi[j]) - Q_t^k(i, \pi[i])\big) + \epsilon^i,
\end{aligned} \tag{12}$$

where $|\epsilon_\alpha^k| \leq \delta_\alpha$ and $|\epsilon^i| \leq \delta_\omega$ are unknown but bounded disturbances. Then, under aforementioned assumptions, we have $\lim_{t\to\infty} \mathbb{E}[\left\|Q_t^i(i, \pi[i]) - Q_t^*(i, \pi[i])\right\|] \leq \Delta$, where $Q_t^*(\cdot)$ is the aggregated vector of the optimal Q-values at time t.

Proof: The outline of the proof is as follows: define upper limit of estimation error and learning rate estimation error, show the control law in (11) is equivalent to the stochastic difference equation used in theorem above, show difference equation satisfies conditions for convergence to bounded error in infinite time. The proof of this convergence is omitted for paper length considerations. ■

IV. Distributed Local Task Assignment

Once the agents have a Q-value table of optimal policies, the agents must coordinate to assign tasks uniquely in order to maximize the profit of a company. We propose a distributed method using a potential game and binary log-linear learning, as shown in green in the right side of Fig. 2. We use a distributed framework game-based method that are compatible the distributed SARSA RL estimation of optimal policies presented in the previous section. There are various algorithms in the literature to solve the task assignment problem. The well-known Hungarian method [11], [12], auction based methods [13], and parallel algorithms [14], and their applications in multi-robot target and task assignment [15]–[17] can be employed to solve our problem formulation. However, these algorithms are mainly designed to solve the assignment problem in a centralized manner. In our framework, m_c and N can be large numbers; hence, it might not be feasible to pass all information at each time step to a command center that could process the information. Furthermore, the complexity

of the overall system makes the problem of constructing a centralized optimal policy computationally heavy or even intractable. Some decentralized methods have been introduced in literature to tackle this problem [18]–[20]. In [20], a distributed auction-based algorithm is introduced, where the task assignment problem is solved in a distributed manner. In that algorithm, 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 large number of customers and agents becomes intractable.

A. Game Design

In this section, we present a game-based local interaction among agents to select their customers in a distributed manner. In particular, we consider a problem with m_c customer requests and N agents available. The agents can only see requests and other agents, if they are in a range.

Definition 1: The pick-up and delivery task is denoted by $T(c_p^j, c_d^l)$. The task is completed when the agent picks up the customer from the pick-up point $c_p^j \in j$ and delivers them to destination $c_d^l \in l$. Note that the terms action and task are used interchangeably in this section.

Assumption 4: Each agent is aware of any pick-up requests within radius of r_c from its current position. In other words, the tasks available for agent i are denoted by the set $U_t^i = \{T(c_p^j, c_d^l) \mid \left\|p_t^i - c_p^j\right\| < r_c\}$.

Assumption 5: Each agent is able to communicate with its neighbors to exchange information. The set of neighbors of agent i is given by $N_i^{\text{comm}}(t) := \{j \mid \left\|p_t^i - p_t^j\right\|_2 \leq R_i^{\text{comm}}\}$, where R_i^{comm} is the communication range of agent i.

We require R_i^{comm} to be larger than or equal to $2r_c$. That is, when the agents have an action set intersection, they can communicate with each other. The agent's action at time step t is denoted by u_t^i, where $u_t^i \in U_t^i$ and U_t^i is the available action set for agent i defined in Assumption 4. The action profile of all agents is denoted by $u_t = (u_t^1, ..., u_t^N) \in U_t := \prod_{i=1}^N U_t^i$.

Now, we propose a non-cooperative game for our agents to solve the task assignment problem in a distributed fashion. First, we design a potential game. To formulate our task assignment problem as a game, we design a utility function, J^i, that aims to capture an action's marginal contribution on the company's profit, for each agent i.

$$J^i(u_t) = H(u_t^i, u_t^{-i}) - H(u_0^i, u_t^{-i}) \tag{13}$$

$$H(u_t) = \sum_{i=1}^{N} h(u_t^i), \tag{14}$$

$$h(u_t^i) = Q(j, a_l) + \mathcal{R}_l(j, l) - C\left\|p_t^i - c_p^j\right\|, \tag{15}$$

for $u_t^i = T(c_p^j, c_d^l)$, where $c_p^j \in j$ and $c_d^l \in l$. u_0^i is the null action of agent i, and u_t^{-i} denotes the actions of all agents other than agent i. $C\left\|p_t^i - c_p^j\right\|$ is the cost for agent i moving from its current position to the pick-up location c_p^j, where C is a design constant. Note that the utility function J^i is local. Our potential game is defined by:

Lemma 1: The assignment game $\Upsilon := \langle N, U, J\rangle$, where $J = \{J^i, i = 1, ..., N\}$ with J^i given by (13), is a potential game with the potential function

$$\Phi(u_t) = H(u_t). \tag{16}$$

Proof: A potential game has to satisfy

$$\Phi\big(u'^i_t, u_t^{-i}\big) - \Phi\big(u_t\big) = J^i\big(u'^i_t, u_t^{-i}\big) - J^i\big(u_t\big) \tag{17}$$

for any agent $i = 1, ..., N$ and action $u'^i_t \in U_t^i$. According to (13) and (16), it is easy to see that (17) holds. ■

B. Game Theory Extension: Ridepooling Utility

Here, we design a new utility function for our task assignment game, where ridepooling is also considered. It is assumed that the vehicle can only service two customers at the same time. This assumption holds for both UberPool and LyftLine, where ridepooling option is offered to customers. Instead of having Assumption 4, the available tasks for each agent is defined as Assumption 6.

Assumption 6: The available tasks for agent i are denoted by set $\bar{U}_t^i = \{[(T(c_p^j, c_d^l), T(c_p^{j'}, c_d^{l'})] \mid \forall T(c_p^j, c_d^l), T(c_p^{j'}, c_d^{l'}) \in U_t^i\}$.

The set of available tasks in Assumption 6, contains all possible coupled customers tasks, including having only one customer. Now, we define the utility function, $h(u_t^i), \forall u_t^i = [(T(c_p^j, c_d^l), T(c_p^{j'}, c_d^{l'})] \in \bar{U}_t^i$, for ridepooling as

$$\begin{aligned}
h(u_t^i) &= \exp^{-C'\beta}\Big[Q(k, a_{\mathsf{d}_2}) + \mathcal{R}_l(j, l) \\
&\qquad + \mathcal{R}_{l'}(j', l') - C\left\|p_t^i - c_p^{\mathsf{p}_1}\right\|\Big], \\
\mathsf{p}_1 &= \arg\min_o \left\|p_t^i - c_p^o\right\|,\ o \in \{j, j'\}, \mathsf{p}_2 = \{j, j'\}\backslash\{\mathsf{p}_1\}, \\
\mathsf{d}_1 &= \arg\min_o \left\|c_p^{\mathsf{p}_2} - c_d^o\right\|,\ o \in \{l, l'\}, \mathsf{d}_2 = \{l, l'\}\backslash\{\mathsf{d}_1\}, \\
\text{Path}_{\min} &= \left\|p_t^i - c_p^{\mathsf{p}_1}\right\| + \left\|c_p^{\mathsf{p}_1} - c_p^{\mathsf{p}_2}\right\| \\
&\qquad + \left\|c_p^{\mathsf{p}_2} - c_d^{\mathsf{d}_1}\right\| + \left\|c_d^{\mathsf{d}_1} - c_d^{\mathsf{d}_2}\right\|, \\
\beta &= \frac{\text{Path}_{\min}}{\min\{\left\|c_p^j - c_d^l\right\|, \left\|c_p^{j'} - c_d^{l'}\right\|\}} - 1,
\end{aligned} \tag{18}$$

where p_1, and p_2 are the first and second pick-up location indices, respectively. The first and second drop off indexes are denoted by d_1 and d_2, respectively. $\text{Path}_{\min}$ denotes the shortest path that agent must travel to accomplish its task, picking up and dropping off both customers, computed from the agent is current position. The cost of sharing the ride for two customers $T(c_p^j, c_d^l)$ and $T(c_p^{j'}, c_d^{l'})$ is denoted by $\beta \geq 0$, where a smaller β indicates a better coupling. $C' > 0$ is a positive constant to be selected, and it is assumed that $p_t^i \in k$. Now, replacing (15) with (18), a new utility function for ridepooling can be calculated. It is easy to see that the game remains a potential game.

C. Nash Equilibrium Convergence Using Binary Log-Linear Learning

We need a distributed adaptation rule to converge to a Nash equilibrium defined in Sec. IV-A. The goal is that each agent

can maximize its own utility function using these rules. Game theoretic reinforcement learning provides iterative algorithms to reach a Nash equilibrium [21], [22].

Binary log-linear learning is a modified version of the log-linear learning for potential games, where only a single player updates its action at each iteration. The agents are allowed to explore and can select non-optimal actions but with relatively low probabilities. This plays an important role for agents to escape the suboptimal actions, and as a result the probability of finding a better Nash equilibrium is increased. Binary log-linear learning can be used for varying available action sets. In [23], it is shown that a potential game will converge to stochastically stable actions, where these actions are the set of potential maximizers if the feasibility and reversibility assumptions are satisfied on the agents available sets. Binary log-linear learning is defined in our system as follows: At each time t, one agent i is randomly selected and allowed to alter its current action, u_t^i, while all other agents repeat their actions, i.e., $u_t^{-i} = u_{t-1}^{-i}$. The selected agent i chooses a trial action $u_t'^i$ uniformly randomly from the available action set U_t^i. The player calculates, $J^i(u_t'^i, u_{t-1}^{-i})$, the utility function for this trial action. Then agent i changes its action according to the following distribution:

$$P_i^{(u_{t-1}^i, u_{t-1}^{-i})}(t) = \frac{\exp(\frac{1}{\tau}J^i(u_{t-1}))}{\exp(\frac{1}{\tau}J^i(u_{t-1})) + \exp(\frac{1}{\tau}J(u_t'^i, u_{t-1}^{-i}))},$$

$$P_i^{(u_t'^i, u_{t-1}^{-i})}(t) = \frac{\exp(\frac{1}{\tau}J^i(u_t'^i, u_{t-1}^{-i}))}{\exp(\frac{1}{\tau}J^i(u_{t-1})) + \exp(\frac{1}{\tau}J^i(u_t'^i, u_{t-1}^{-i}))},$$

$$P_i^{(u''^i_t, u_{t-1}^{-i})}(t) = 0, \quad \forall u''^i_t \neq u_t'^i, u_{t-1}^i. \tag{19}$$

where $P_i^{(u^i, u_{t-1}^{-i})}(t)$ denotes the probability of choosing action u^i at time t while other agents are repeating their action u_{t-1}^{-i}. The coefficient $\tau > 0$ is a design parameter specifying how likely agent i chooses a suboptimal action, to specify the trade-off between exploration and exploitation.

V. Simulation and Discussion

To validate the proposed algorithms, we prepare simulations using taxi data provided by the city of Chicago, [24]. The city is partitioned into 77 cells (as shown in Fig. 3). In provided data, each entry contains the pick-up and drop-off cells of the trip, time-stamp, duration and fare. We use data from May 2017, which gives us approximately one million trips to analyze. In order to numerically compare the centralized and decentralized SARSA RL algorithms, we use the respective learning update with the same game theory task assignment. First, we illustrate the algorithm at different timesteps, then we show convergence of our distributed SARSA RL algorithm to the centralized solution with two metrics: tracking estimated Q values, and comparing total revenue of each algorithm, and finally we show the economic advantage of a learning environment. Also, we provide an animation of the task assignment, provided in https://youtu.be/j1zuadmi7OQ.

We illustrate the algorithm in practice in Fig. 4. In (a)-(c), agents are iteratively running the algorithm to select their customers, where the utility function J^i for each selected customer is shown. In (d), agents are picking up and dropping off their selected customers with maximum utility function. In (e), agents are at their destinations after accomplishing their tasks and observing local new customers. Then, they execute the game theory task assignment again.

The first validation of proposed distributed SARSA RL method is to compare distributed and centralized Q-value tracking of $Q(8, a_8^8)$, as shown in Fig. 5. As expected, we see the distributed estimate approach the centralized algorithm's

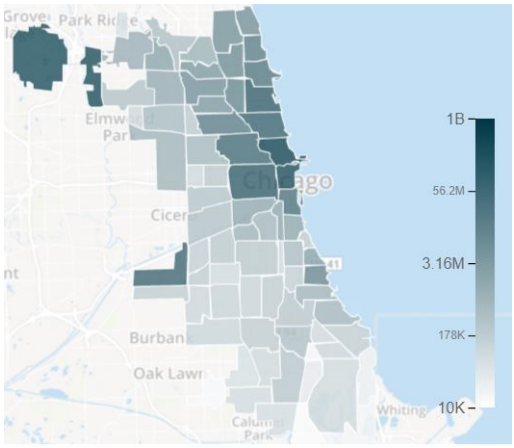

Fig. 3. Grid map of city of Chicago with 77 cells . Colormap corresponds to number of taxi trips.

Algorithm 1 Simulation Formulation

Input: Trip Data: [start cell, end cell, fare, time]
Output: Customer Assignments for each agent

Initialize agents with $\boldsymbol{Q}_t^i = \boldsymbol{Q}_{MDP}, \boldsymbol{f}_t^i, \boldsymbol{g}_t^i, \boldsymbol{\omega}_t^i, \alpha_t^i$
while true **do**
  Information Seen by Agents
  for $\forall$ agent $\in$ agents **do**
    Find tasks, $T(c_p^j, c_d^l)$, within r_{tasks} of agent
    Find agents within R_{comm} of agent
  end for
  Task Assignment
  while $\neg\forall u_t^i$ converged **do**
    Randomly select agent, i, with current action u_t^i
    Find J, marginal contribution for current action using agents estimate, (Q_t^i) (13) (15)
    Find J_p, marginal contribution for randomly proposed action using agents estimate, (Q_t^i) (13) (15)
    Change action with probability: $P_i^{(u_t'^i, u_{t-1}^{-i})}$, (19)
  end while
  Learning Update
  for $\forall$ agent $\in$ agents **do**
    Use agent i's current customer reward information to update r_t^i and $\boldsymbol{r_t^i}$ using update law (10)
    Update $\boldsymbol{\omega}_t^i, \hat{\boldsymbol{f}}_t^i, \boldsymbol{g}_t^i, \alpha_t^i$ using learning rate rule (11) and neighbor consensus information.
    Update $\boldsymbol{Q}_{t+1}^i$ using update law (10).
  end for
  Update agents information
end while

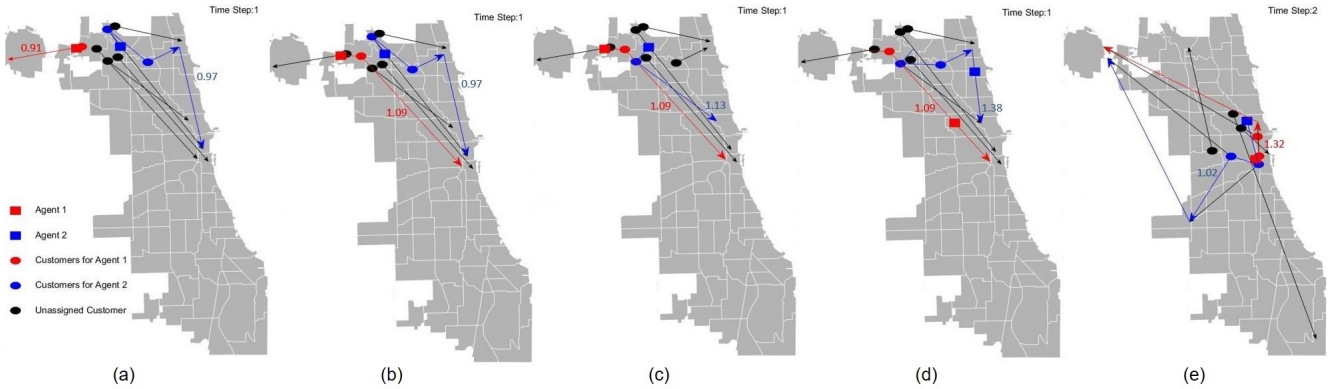

Fig. 4. The task assignment algorithm proposed in Sec. IV is illustrated for two agents

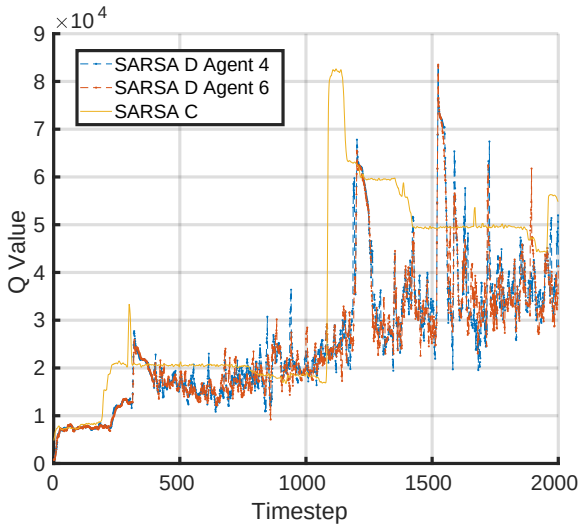

Fig. 5. The Q-values for a state-action pair tracking for centralized and distributed algorithms. Only two of many agents Q-estimates are shown for plot readability. The distributed algorithm is capable of tracking a dynamic system.

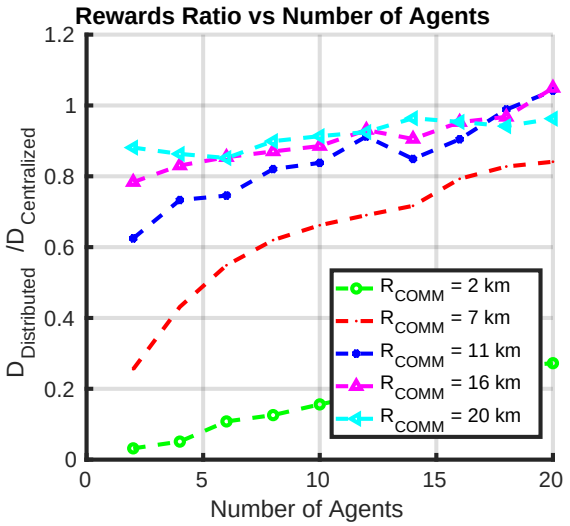

Fig. 6. Total Revenue vs number of agents with varying communication radius. Each data point is a simulation.

Q-value and we see that the distributed estimation is capable of dynamic tracking. The non-zero error at large time-value is captured in our model by the error bound, Δ. This simulation is run with ten agents, and both results are presented through a moving average filter.

The second validation of the proposed distributed SARSA RL method is to compare the total revenue generated by distributed and centralized policies, D_D and D_C, respectively. The ratio of the generated revenue, $\frac{D_D}{D_C}$ is plotted against number of agents in Fig. 6 with varying radius of communication between agents.

Figure 6 reveals a few important effects. First, as the radius of communication increases, the revenue ratio approaches one. This corresponds to the effect that if every agent can communicate with every other agent (a complete graph), the distributed solution for every agent will converge to the centralized solution and we will recover a revenue ratio of one. The second effect we observe is that as the number of agents increases, the ratio approaches one. This illustrates the advantage of distributed systems with a large number of agents. All held equal, number of agents increase would increase the estimation error from consensus. However in our case, the rewards increase because the connectivity of the graph is getting better. So there are two competing effects determining the performance relative to number of agents. These trends are expected and validate our algorithm in a numerical simulation.

Figure 7 demonstrates the economic utility of our proposed algorithm. This simulation is for 20 agents with a radius of communication of 5.5km. This is the cumulative reward of each algorithm, normalized against the cumulative reward of the centralized SARSA RL. We define a 'greedy' algorithm where agents value each trip from the immediate reward. We also define a 'lazy' algorithm where agents value each trip from the how close the request is to the agent's current location. Both these algorithms have no forecasting ability, the value function is identically zero. At early time-steps, the distributed SARSA RL algorithm performs similarly to the greedy and lazy algorithms, but outperforms these algorithms

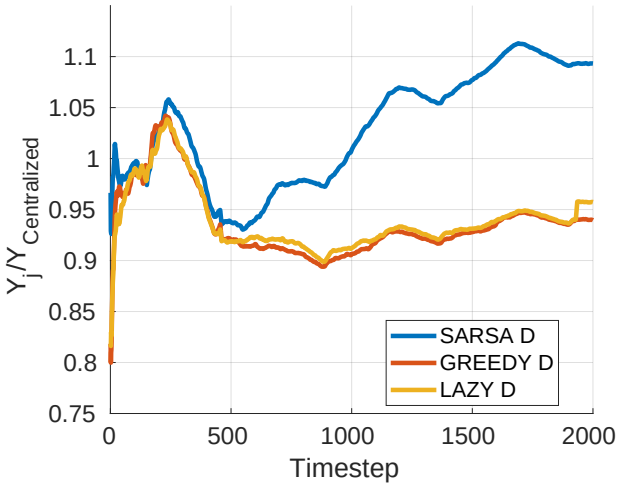

Fig. 7. This is the cumulative revenue at each timestep for each algorithm, Y, normalized against the cumulative revenue of the centralized SARSA RL, Y_C. We see that the distributed algorithm outperforms the greedy and lazy algorithms.

overtime because the agents are collectively updating information on a changing environment by estimating future values using the Q-value formulation. Over equivalent to two weeks, the average return of each trip for the greedy algorithm is 10.52 USD and the average return for our distributed SARSA RL algorithm is 12.03 USD.

VI. Conclusion

In this paper, two learning based algorithms were proposed to solve the optimal transportation planning problem of autonomous vehicles for dynamic route services. First, we propose an adaptive learning rate for SARSA RL learning in non-stationary environments. Second, we propose a distributed version of SARSA RL and show its convergence. A game-theory-based task assignment algorithm is proposed, where each agent used the high-level recommendations, provided by distributed SARSA RL, to select the optimal customer from the set of local available requests in a distributed manner. Finally, the customers data provided by the city of Chicago was used to validate the proposed algorithms and it is shown that the distributed estimation of Q-values converges to the centralized solution. Future directions include further simulation results with different data-sets, using heterogeneous agents for information gathering and task allocation, or expanding a ride-share utility for a single agent to task multiple requests.

References

[1] J. Holden and N. Goel, "Uber elevate: Fast-forwarding to a future of on-demand urban air transportation," *https://www.uber.com/elevate.pdf*, 2016.

[2] 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, Jul 2007.

[3] J.-F. Cordeau and G. Laporte, "The dial-a-ride problem (darp): Variants, modeling issues and algorithms," *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, vol. 1, no. 2, pp. 89–101, Jun 2003.

[4] 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, Nov. 2006.

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

[6] V. Pillac, M. Gendreau, C. GuÃl'ret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1 – 11, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221712006388

[7] 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.

[8] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[9] R. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics*, pp. 679–684, 1957.

[10] T.-J. Tarn and Y. Rasis, "Observers for nonlinear stochastic systems," *IEEE Transactions on Automatic Control*, vol. 21, no. 4, pp. 441–448, 1976.

[11] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955. [Online]. Available: http://dx.doi.org/10.1002/nav.3800020109

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

[13] 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, Dec 1988.

[14] 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, Sep. 1991.

[15] C. Schumacher, P. R. Chandler, and S. R. Rasmussen, "Task allocation for wide area search munitions," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, vol. 3, May 2002, pp. 1917–1922 vol.3.

[16] Y. Jin, A. A. Minai, and M. M. Polycarpou, "Cooperative real-time search and task allocation in uav teams," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, vol. 1, Dec 2003, pp. 7–12 Vol.1.

[17] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, *Multi-Task Allocation and Path Planning for Cooperating UAVs*. Boston, MA: Springer US, 2003, pp. 23–41.

[18] D. Dionne and C. A. Rabbath, "Multi-uav decentralized task allocation with intermittent communications: the dtc algorithm," in *2007 American Control Conference*, July 2007, pp. 5406–5411.

[19] P. B. Sujit and R. Beard, "Distributed sequential auctions for multiple uav task allocation," in *2007 American Control Conference*, July 2007, pp. 3955–3960.

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

[21] D. Fudenberg and D. K. Levine, *The Theory of Learning in Games*, ser. MIT Press Books. The MIT Press, January 1998, vol. 1, no. 0262061945.

[22] W. H. Sandholm, "H. peyton young, strategic learning and its limits , oxford univ. press (2004) 165 pages," *Games and Economic Behavior*, vol. 63, no. 1, pp. 417–420, May 2008.

[23] 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.

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