2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Multiagent Reinforcement Learning for Autonomous Routing and Pickup Problem with Adaptation to Variable Demand

Daniel Garces[1], Sushmita Bhattacharya[1], Stephanie Gil[1], Dimitri Bertsekas[2]

***Abstract*— We derive a learning framework to generate routing/pickup policies for a fleet of autonomous vehicles tasked with servicing stochastically appearing requests on a city map. We focus on policies that 1) give rise to coordination amongst the vehicles, thereby reducing wait times for servicing requests, 2) are non-myopic, and consider *a-priori* potential future requests, 3) can adapt to changes in the underlying demand distribution. Specifically, we are interested in policies that are adaptive to fluctuations of actual demand conditions in urban environments, such as on-peak vs. off-peak hours. We achieve this through a combination of (i) an online play algorithm that improves the performance of an offline-trained policy, and (ii) an offline approximation scheme that allows for adapting to changes in the underlying demand model. In particular, we achieve adaptivity of our learned policy to different demand distributions by quantifying a region of validity using the q-valid radius of a Wasserstein Ambiguity Set. We propose a mechanism for switching the originally trained offline approximation when the current demand is outside the original validity region. In this case, we propose to use an offline architecture, trained on a historical demand model that is closer to the current demand in terms of Wasserstein distance. We learn routing and pickup policies over real taxicab requests in San Francisco with high variability between on-peak and off-peak hours, demonstrating the ability of our method to adapt to real fluctuation in demand distributions. Our numerical results demonstrate that our method outperforms alternative rollout-based reinforcement learning schemes, as well as other classical methods from operations research.**

I. Introduction

We consider the problem of dynamic multiagent autonomous taxicab routing for rider pickups, a special case of the Dynamic Vehicle Routing (DVR) problem [1]. This problem has relevance to several robotics tasks, including coordinated package delivery [2] [3] [4], warehouse robot path planning [5], autonomous transportation, and on-demand mobility systems [6] [7] [8] [9] [10]. We express the demand as riders waiting to be transported, and assume that taxicabs can transport one rider at a time. We are interested in finding a strategic, cooperative pickup plan for a fleet of autonomous taxicabs to minimize the total wait time of all riders where the number and location of future requests are not known *a-priori*. Obtaining an optimal solution for this problem is intractable since it requires considering multiple scenarios of potential future requests and all relevant taxicab actions at each decision point. This condition results in an extremely large state space and a control space that grows exponentially with the number of agents. Hence, finding a competitive sub-optimal solution is crucial.

Several sub-optimal solutions to the taxicab pickup problem explore instantaneous assignment [11] [12] [13] [14], and other routing heuristics, including 2-opt [15], local search [16], and genetic algorithms [17]. These methods tend to generate myopic policies due to the lack of consideration for future demand. Sampling-based stochastic optimization methods that consider potential future requests [18] address this limitation, but at the expense of long computation times due to multistep planning in a large state space. Several learning-based approaches aim to achieve faster computation times. Some authors consider offline trained approximations, such as approximate value iteration [19], Deep Q-learning [20], Deep Q Networks [21], and transformer-based architectures [22]. Offline trained architectures may fail to generalize to unknown scenarios, not fully represented in the training data. This condition makes them infeasible for deployment in real urban environments with fluctuating demand. Other authors consider online policy evaluation with finite lookahead, including Monte Carlo Tree Search (MCTS) [23], DESPOT [24], and a Multiple Scenario Approach (MSA) [25]. These methods tend to be computationally expensive if no approximations are used.

In this paper we aim to address the lack of generalization of offline trained architectures, and the high policy evaluation time of online methods, by proposing a hybrid planning method with online optimization on top of an offline trained policy approximation. Our method obtains a competitive suboptimal solution to the taxicab routing problem by leveraging (i) *online play*, a lookahead optimization scheme that improves on the results of offline training [26] [27], and (ii) an offline approximation switching scheme that endows the system with adaptivity in the face of significant changes in the underlying demand model. We use an offline trained approximation as the base policy for online play for faster computation times compared to simulation-based rollout algorithms. Our offline approximation is implemented using Graph Neural Networks (GNNs) [28] that exploit the topological characteristics of a city environment. We achieve adaptivity for the offline approximations by formulating Wasserstein Ambiguity Sets [29] centered around representative historical demand models. These sets represent regions of validity, or probability spaces in which the offline approximations correctly approximate the rollout based Re-

[1]Daniel Garces, Sushmita Bhattacharya, and Stephanie Gil are with the REACT lab, Harvard University, Boston, MA, USA (e-mail: [dgarces]@g.harvard.edu, sushmita_bhattacharya@g.harvard.edu, sgil@g.harvard.edu).

[2]Dimitri Bertsekas is with the Department of Electrical Engineering and Computer Science , Arizona State University, AZ, USA (e-mail: dimitrib@mit.edu).

This work was supported by ONR YIP (grant # N00014-21-1-2714), and Amazon Research Award.

2023 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-2365-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRA48891.2023.10161067

inforcement Learning (RL) method [26] under a particular demand model.

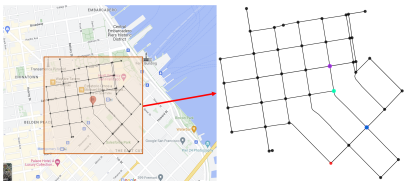

Fig. 1. Example of a topological map and corresponding graph preserving one-way streets and traffic direction constraints.

The main contributions of this work are as follows: 1) We formulate the autonomous taxicab routing problem as a stochastic Dynamic Program (DP) in such a way that an offline trained GNN successfully approximates a rollout-based RL method. This then allows us to apply online play as an approximate Newton step [27] to further improve the performance of the learned policy. 2) Our method adapts to fluctuating demand conditions by switching between offline approximations to maintain the performance improvement of online play. We replace the offline approximation once the current demand is no longer inside the approximation's region of validity. 3) We apply our approach on a real taxicab pickup dataset [30] (an example map is shown in Fig. 1). We empirically show that our method outperforms one-at-a-time rollout with a simple base policy, and several classical benchmark algorithms operations research (OR).

The rest of the paper is organized as follows: In Sec. II, we present our formulation of the multiagent taxicab routing problem. In Sec. III, we present our approach, our application of online play with offline approximation, and our formulation of Wasserstein Ambiguity Sets to quantify regions of validity. Finally, in Sec. IV we present numerical results on the San Francisco taxicab pickup dataset [30].

II. Problem Formulation

Here, we present the formulation for a multiagent taxicab routing problem as a discrete time, finite horizon, stochastic DP problem. In the following subsections, we outline the environment, requests, state and control representations.

A. Environment

We assume that taxicabs are deployed in urban environments with a fixed street topology, represented as a directed graph, $\mathbb{G} = (\mathbb{V}, \mathbb{E})$. Here, $\mathbb{V} = \{1, \dots, n\}$ corresponds to the set of indices for the street intersections in the map of the city with a total of n intersections, and $\mathbb{E} \subseteq \{(i,j) | i, j \in \mathbb{V}\}$ corresponds to the set of directed streets connecting intersections i to j. The set of *neighbors* of intersection i is denoted by $\mathcal{N}(i) = \{j | j \in \mathbb{V}, (i,j) \in \mathbb{E}\}$.

B. Requests

A request r is represented as a tuple $r = (\rho_r, \delta_r, k_r, \phi_r)$, where $\rho_r \in \mathbb{V}$ and $\delta_r \in \mathbb{V}$ correspond to the closest intersections to the desired pickup and dropoff locations for the request, respectively; k_r corresponds to the time at which the request enters the system; and $\phi_r \in \{0, 1\}$ is an indicator variable which equals 0 if the request has not been assigned to any agent, otherwise $\phi_r = 1$. We model the number of requests that enter the system as a random variable η with an unknown underlying distribution p_η, and we denote its realization at time k as $\eta(k)$. p_η is fixed for the entire length of the time horizon N and we estimate it using either historical data or data for the last hour of execution of the system, to obtain categorical probability distributions $\tilde{p}_\eta$ (historical) or $\tilde{p}_{\eta,c}$ (current), respectively. We denote the set of requests that enter the system at time k as $\mathbf{r_k}$. Here the cardinality of the new request set at time k is $|\mathbf{r_k}| = \eta(k)$.

We define $\bar{\mathbf{r}}_\mathbf{k}$ as the list of outstanding requests that have not been assigned to any agent till time k, such that $\bar{\mathbf{r}}_\mathbf{k} = \{r | r \in \mathbf{r_t}, \phi_r = 0, 1 \leq t \leq k\}$. The pickup ρ_r and dropoff δ_r locations of a new request r are determined by underlying probability distributions p_ρ and p_δ, where ρ and δ correspond to random variables for the pickup and dropoff locations, respectively. These two probability distributions are also unknown a-priori. We estimate the corresponding categorical distributions $\tilde{p}_\rho$ and $\tilde{p}_\delta$ using historical data, and $\tilde{p}_{\rho,c}$ and $\tilde{p}_{\delta,c}$ using the last hour data. The estimated distributions $(\tilde{p}_\eta, \tilde{p}_\rho, \tilde{p}_\delta)$ compose the historical demand model and $(\tilde{p}_{\eta,c}, \tilde{p}_{\rho,c}, \tilde{p}_{\delta,c})$ compose the current demand model. We discuss the estimation process in Sec. IV-B.

C. State representation and control space

We assume there are a total of m agents and each agent can perfectly observe all requests, and all agents' locations and occupancy status. The state at time k is $x_k = (\boldsymbol{\nu}_k, \boldsymbol{\tau}_k, \bar{\mathbf{r}}_\mathbf{k})$. Here, $\boldsymbol{\nu}_k = [\nu_k^1, \dots, \nu_k^m]$, is a list of locations for all m agents at time k, where $\nu_k^\ell \in \mathbb{V}$ is the index of the closest intersection to the geographical coordinates of agent ℓ. $\boldsymbol{\tau}_k = [\tau_k^1, \dots, \tau_k^m]$, is a list of time remaining in assigned trip for all agents at time k, and $\bar{\mathbf{r}}_k$ is the set of outstanding requests at time k. If $\tau_k^\ell = 0$, then agent ℓ is available and new requests can be assigned to the agent, otherwise $\tau_k^\ell \in \mathbb{N}^+$.

The control space of agent ℓ at time k is denoted by $\mathbf{U}_k^\ell(x_k)$. If $\tau_k^\ell = 0$ (agent is available), then $\mathbf{U}_k^\ell(x_k) = \{\mathcal{N}(\nu_k^\ell), \nu_k^\ell, \zeta\}$, where ζ is a special pickup control that becomes available if there is a request $r \in \bar{\mathbf{r}}_t, 1 \leq t \leq k$, such that its pickup location $\rho_r = \nu_k^\ell$. If $\tau_k^\ell \neq 0$, then the agent must complete its current trip before picking up a new request. So $\mathbf{U}_k^\ell(x_k) = \{h\}$, where h is the next hop in the shortest path (given by Dijkstra's algorithm) between ν_k^ℓ and the dropoff location δ_r for the agent's assigned request r. Since this formulation represents a separable control constraint for each agent, the controls available to all agents at time k, $\mathbf{U}_k(x_k)$, is expressed as the Cartesian product of local control sets $\mathbf{U}_k^1(x_k) \times \cdots \times \mathbf{U}_k^m(x_k)$.

D. Stochastic dynamic programming formulation

We now present our formulation of the taxicab routing problem as a finite horizon, stochastic DP problem. The objective is to find a pickup strategy that minimizes the total wait time of requests (in minutes). The state transition function is denoted by f_k, and $x_{k+1} = f_k(x_k, u_k, \eta, \rho, \delta)$, where x_{k+1} is the state at time $k+1$ after application of control u_k at time k from the current state x_k, and $g_k(x_k, u_k, \eta, \rho, \delta)$ is the stage cost. A policy $\pi = \{\mu_1, \dots, \mu_N\}$ is a list of functions, where μ_k maps state x_k into control $u_k = \mu_k(x_k) \in \mathbf{U}_k(x_k)$. The cost of policy π at state x_1 is expressed

as $J_\pi(x_1) = E\left[g_N(x_N) + \sum_{k=1}^{N-1} g_k(x_k, \mu_k(x_k), \eta, \rho, \delta)\right]$, where $g_N(x_N)$ is the terminal cost. The Bellman equation for the optimal policy $\pi^* = \{\mu_1^*, \ldots, \mu_N^*\}$ is

$$\mu_k^*(x_k) \in \underset{u_k \in \mathbf{U}_k(x_k)}{\operatorname{argmin}} E[g_k(x_k, u_k, \eta, \rho, \delta) + J_{k+1}^*(x_{k+1})], \tag{1}$$

$k = 1, \ldots, N$. We set the stage cost as the number of outstanding requests or $g_k(x_k, u_k, \eta, \rho, \delta) = |\bar{\mathbf{r}}_\mathbf{k}|$. The set of outstanding requests, $\bar{\mathbf{r}}_\mathbf{k}$, at time k depends on the current control input u_k and the random variable η. We set $|\bar{\mathbf{r}}_\mathbf{k}| = |\bar{\mathbf{r}}_{\mathbf{k-1}}| + \eta(k) - \psi(x_k, u_k)$, where the function $\psi(x_k, u_k)$ determines the number of requests serviced by executing control u_k at state x_k. Since there is no request before $k = 1$, we set $|\bar{\mathbf{r}}_\mathbf{0}| = 0$, $\psi(x_1, u_1) = 0$ and $|\bar{\mathbf{r}}_\mathbf{1}| = |\mathbf{r_1}|$. The optimal cost for our taxicab problem is, $J_{\pi^*}(x_1) = \min_{\pi \in \Pi} E\left[\sum_{k=1}^{N} g_k(x_k, u_k, \eta, \rho, \delta) \mid \pi, \eta, \rho, \delta\right]$.

The size of the state space at a time k, with m available taxicabs is $O(|\mathbb{V}|^m(|\mathbb{V}| \times |\mathbb{V}|)^{|\mathbf{r_k}|})$ since each available taxicab can be located at any of the $|\mathbb{V}|$ locations and there may be $|\mathbb{V}| \times |\mathbb{V}|$ possible pickup-dropoff location pairs for each request. The control space grows exponentially with the number of agents. Finding an optimal policy for such a large multiagent taxicab routing problem is intractable, and hence, we look for suboptimal solutions.

III. Our Approach

Our approach can be characterized as a form of approximation in value space, built on top of a self-learning policy iteration scheme. It has: a) *An online play algorithm* that leverages the results of the offline policy iteration. b) *An offline training algorithm* that is based on approximate policy iteration, with the policy cost functions approximated using neural networks. c) *Agent-by-agent rollout policies* for faster generation of training samples for policy evaluation.

Through online play as an approximate policy improvement step, we obtain a competitive suboptimal policy that improves over the performance of the offline policy (see Sec. III-A). We choose an offline policy that approximates a variant of the rollout algorithm known as one-agent-at-a-time (one-at-a-time) rollout [31] [32], as it scales linearly with the number of agents, instead of exponentially like the standard rollout algorithm [33]. We implement the offline approximation using GNNs to leverage the topological structure of the street network (see Sec. III-B). Since our policy approximation captures the behavior of a rollout policy derived for a specific historical demand model, a deviation in the current demand model from the original historical demand model might affect the quality of the approximation. This change in the approximation performance will in turn affect the performance of the online play. To address this, we quantify regions of validity using Wasserstein Ambiguity Sets [29], and propose a method for switching to a different offline approximation based on the Wasserstein Distance [34] [35] between the current and the historical demand model used for training the offline approximation (see Sec. III-C).

A. Background: rollout algorithm, offline approximation, and online play

1) One-agent-at-a-time (one-at-a-time) rollout

We now discuss rollout [26], [36], denoted by $\tilde{\pi}$, where the optimal cost J_{k+1}^* in the Bellman equation (Eq. 1) is replaced with a cost approximation $\tilde{J}_{k+1}$. This rollout finds a non-myopic solution with consideration of the future using lookahead optimization. Each agent's control is obtained by performing one-step lookahead minimization over the agent's control components. Rollout's exhaustive expectation estimation performs better than other RL algorithms (including MCTS [23]) that use longer and sparser lookahead trees with inexact expectation estimation [37]. The cost approximation $\tilde{J}$ can be estimated by the cost of applying a base policy π, for t times followed by a terminal cost approximation $\tilde{J}$. A base policy can be given by simple heuristics. Agent ℓ's one-at-a-time rollout control at state x_k, is

$$\tilde{u}_k^\ell \in \underset{u_k^\ell \in \mathbf{U}_k^\ell(x_k)}{\operatorname{argmin}} E[g_k(x_k, \bar{u}, \eta, \rho, \delta) + \tilde{J}_{k+1}(x_{k+1})] \tag{2}$$

where $\bar{u} = (\tilde{u}_k^1, \ldots, \tilde{u}_k^{\ell-1}, u_k^\ell, \mu_k^{\ell+1}(x_k), \ldots, \mu_k^m(x_k))$. This agent by agent optimization scales linearly with the number of agents. [31], [32], and [26] show that the one-at-a-time rollout with one-step lookahead guarantees *cost improvement* [26] of the rollout policy $\tilde{\pi}$, improving over base policy π.

2) Offline Approximation and Online Play

Now, we discuss the offline trained policy $\hat{\mu}$ used to approximate the one-at-a-time rollout-based RL policy. To train the policy approximation, we generate a large set of random states, each with random initial taxicab locations. The state and other agent's actions are features, while the corresponding one-at-a-time rollout controls are labels. The training feature at state x_k for agent ℓ is $F(x_k, \ell) = (x_k, \tilde{u}_k^1, \ldots, \tilde{u}_k^{\ell-1}, \mu_k^{\ell+1}(x_k), \ldots, \mu_k^m(x_k))$ and the training label is $\tilde{\mu}_k^\ell, \ell \in \{1, \ldots, m\}$. $\hat{\mu}$ is a function that maps $F(x_k, \ell)$ to the rollout control for agent ℓ. Fig. 2 shows the training data generation for 2 agents.

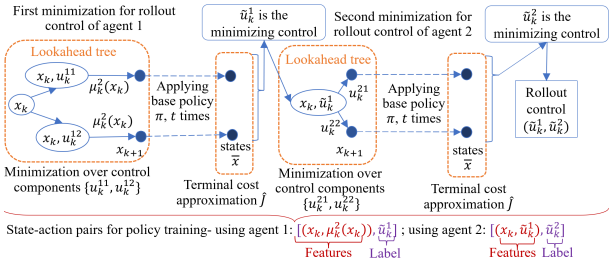

Fig. 2. Our pipeline for training data generation for the offline policy approximation of one-at-a-time rollout with two agents.

The online play [27], denoted by $\bar{\pi}$, evaluates the one-at-a-time rollout policy with the policy approximation $\hat{\pi}$ as base policy which makes online play act as an approximate policy iteration step over the rollout policy $\tilde{\pi}$. Online play's control $\bar{u}_k^\ell$ at state x_k is given by Eq. 2, with $\bar{u} = (\bar{u}_k^1, \ldots, \bar{u}_k^{\ell-1}, u_k^\ell, \hat{\mu}(F(x_k, \ell+1)), \ldots, \hat{\mu}(F(x_k, m)))$.

If the policy approximation, $\hat{\mu}$, correctly approximates the rollout policy, $\tilde{\pi}$, on the current demand model, we expect online play policy $\bar{\pi}$ to outperform $\tilde{\pi}$ following the cost

improvement property of the approximate policy iteration [33]. If $\hat{\mu}$ fails to approximate the rollout policy $\tilde{\pi}$ because of a change in the current demand model, the cost improvement property will not hold and the online play will not provide a significant improvement. We address these issues in our discussion of adaptivity in Sec. III-C. Next, we present our approach for policy approximation.

B. A hybrid framework for offline approximation and online play for policy generation in the taxi routing problem

Now, we present our approach of approximating the one-at-a-time rollout policy using GNNs (implemented as suggested in [38]). Since we encode the environment as a graph, using GNNs allows us to leverage the connectivity between intersections to boost the performance of the approximation. In our method, we use GNNs [28] as the offline approximation to approximate the one-at-a-time rollout with base policy π. Since encoding the behavior of one-at-a-time rollout is a complex task composed of two main actions, pickup and movement, we separate the offline approximation into two networks. The first GNN determines if an available agent should pickup a request in its current location, and the second GNN determines the next intersection towards which the agent should move. If the first GNN determines that an agent should pick a request, the output of the second GNN is ignored. The pickup GNN is composed of 3 graph convolutional layers, followed by 3 linear layers, while the move GNN is composed of 2 graph convolutional layers, followed by 4 linear layers (see Fig. 3). We select these architecture parameters after performing a hyperparameter search. We train a pair of GNNs for all agents for each representative demand model. The training data for the GNN is generated following the process shown in Fig. 2. We encode the state x_k as a set of node features and global features. Global features ($\in \mathbb{R}^m$) describe global properties of the state, containing only the list of time remaining $\boldsymbol{\tau_k}$ in current trips for all agents. Node features ($\in \mathbb{R}^{m+2}$), at each intersection, encode the presence of each of m agents, indicate if the intersection is chosen as the next move for any of the other agent's potential current actions, and include the number of pickup requests available at the intersection.

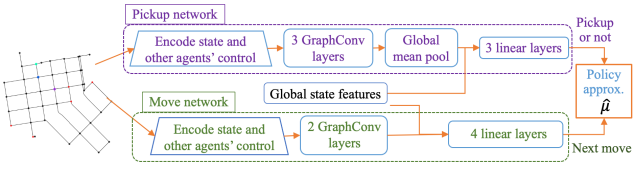

Fig. 3. Graph convolutional neural network architectures for the offline policy approximation $\bar{\pi}$.

With this implementation of the offline approximation, we apply online play (given in Sec. III-A.2). Next, we explain how we deal with fluctuations in the current demand model.

C. Adaptivity to changing demand distributions

We propose a method for providing adaptivity to our GNN-based policy approximation by quantifying regions of validity. These regions are defined using Wasserstein Ambiguity Sets which rely on the Wasserstein Distance, a distance function between two probability distributions on a given probability space. We choose the Wasserstein Distance metric because (i) this metric considers the closeness between support points while other metrics only consider their probabilities, and (ii) Wasserstein Ambiguity Sets are rich enough to consider discrete distributions outside of the original support while other metrics, including Kullback-Leibler divergence, do not allow for this.

We now formally define the Wasserstein distance. For this, we choose the estimated distribution for the number of requests, $\tilde{p}_\eta$, to be the reference distribution, since the variation in $\tilde{p}_\eta$ over the peak and non-peak hours is most significant. However, the full quantification of the distance between demand models may include the pickup ($\tilde{p}_\rho$) and dropoff ($\tilde{p}_\delta$) distributions, which change less dramatically over the hours of a day. In this setting, we define an atom as a measurable set which has a positive measure and contains no set of smaller positive measure [39]. For a finite support Ω, which includes X data points, let ξ_j^c denote the j-th atom of the current demand distribution $\tilde{p}_{\eta,c}$, and ξ_i be the i-th atom of the representative historical distribution $\tilde{p}_\eta$. p_j^c denotes the probability of ξ_j^c, p_i denotes the probability of ξ_i, and f_{ij} denotes the bivariate probability mass function for ξ_j^c and ξ_i, $\forall i,j \in \{1,\dots,X\}$. With these definitions, the Wasserstein Distance [34] $d_W(\tilde{p}_{\eta,c},\tilde{p}_\eta)$ of order 1, also known as the Kantorovich distance, is defined as $d_W(\tilde{p}_{\eta,c},\tilde{p}_\eta) = \inf_{f\geq 0}\sum_{i,j\in\{1,\dots,X\}} f_{ij}||\xi_j^c - \xi_i||$, subject to $\sum_{j\in\{1,\dots,X\}} f_{ij} = p_i, \forall i \in \{1,\dots,X\}$, and $\sum_{i\in\{1,\dots,X\}} f_{ij} = p_j^c, \forall j \in \{1,\dots,X\}$.

We now formally define the Wasserstein Ambiguity Set and q-valid radius. We let $\mathcal{P}(\Omega)$ represent the space of all underlying probability distributions $p_{\eta,c}$ supported on Ω. The Wasserstein Ambiguity Set, $\mathcal{D}_W$, is a ball of radius θ centered around the reference distribution $\tilde{p}_\eta$ defined as $\mathcal{D}_W := \{p_{\eta,c} \in \mathcal{P}(\Omega) | d_W(p_{\eta,c},\tilde{p}_\eta) < \theta\}$ [29]. We choose the radius θ for the ambiguity sets to be the lower bound for the q-valid radius as in Theorem 2 of [35]

$$\theta \geq (B+0.75)(-\log(1-q)/X + 2\sqrt{-\log(1-q)/X}),$$

where the q-valid radius, following [35], corresponds to the radius that ensures $\tilde{p}_{\eta,c} \in \mathcal{D}_W$ with probability at least q, defining the bounds for the region of validity; B is the diameter of the compact support Ω, which in the taxicab routing setting is the maximum number of requests/minute. Following this formulation, if the current demand distribution is outside the q-valid radius for a given representative historical distribution, then there is a different representative distribution that contains the current demand distribution inside its ambiguity set with a higher probability. Therefore, whenever the current demand model gets outside of the q-valid radius, we choose the policy approximation, trained on the representative demand closest to the current demand in Wasserstein distance.

IV. Numerical Experiments

Now, we outline our implementation details and simulation results on a real taxicab dataset [30]. We show that our method outperforms rollout and several OR based benchmarks, being robust towards changing demands.

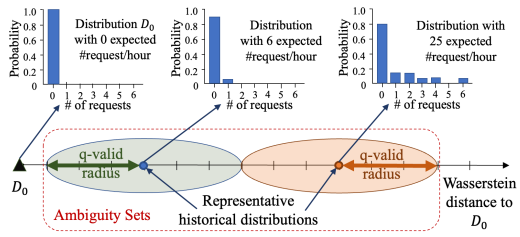

Fig. 4. Ambiguity sets induced by q-valid radii of the representative distributions from the historical demand model.

A. Experimental Setup

We consider a $400 \times 400m^2$ section of San Francisco's financial district [30] (see Fig. 1) with 42 nodes and 125 edges, and $m = 3$ taxicabs (with the state and control space sizes of 10^{78}, and 216, respectively). We consider 1-minute edge travel time and the time horizon of $N = 60$ minutes.

B. Estimating the demand model

The historical demand model is composed of three estimated categorical distributions $\tilde{p}_\eta$ for number of requests, $\tilde{p}_\rho$ for pickup locations, and $\tilde{p}_\delta$ for dropoff locations. We partition the historical data in 1-hour intervals, where each time step k spans 1 minute. We empirically estimate $\tilde{p}_\eta$ by looking at the number of requests that arrive at each minute within each 1-hour time span. $\tilde{p}_\rho$ and $\tilde{p}_\delta$ are derived from the historical requests [30] that originated and ended inside the map section. The probability of a request emerging at pickup node y is $\tilde{p}_\rho(y) = (s_y + 1/|\mathbb{V}|)/(1 + \sum_{j\in\mathbb{V}} s_j)$, where s_y is the number of requests with pickup location y. We assign a small nonzero probability of request origination to all intersections to represent the idea that requests may originate at any intersection. The dropoff location probability $\tilde{p}_\delta$ is estimated similarly from the historical dropoffs in [30]. For our experiments, we consider three different demand models: low, medium, and high, (see Fig. 5) with the same $\tilde{p}_\rho$ and $\tilde{p}_\delta$, but with different $\tilde{p}_\eta$. The low, medium and high demand models have $E[\eta] \cdot N$ of $3, 9$ and 25, respectively.

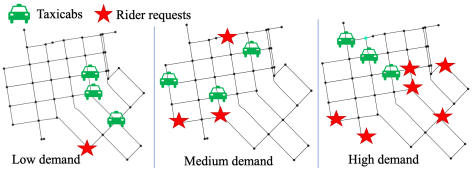

Fig. 5. Visualizing various demand models.

The current demand model $(\tilde{p}_{\eta,c}, \tilde{p}_{\rho,c}, \tilde{p}_{\delta,c})$ is derived similarly by looking at the latest hour of execution of the system, instead of looking at historical data. The historical demand models are used to train the policy approximation, while the current demand model is used as the demand model for evaluation of all policies.

C. Implementation details of one-at-a-time rollout

We consider a greedy base policy π where taxis are routed to their nearest request without coordination. The one-at-a-time rollout employs a 1-step lookahead optimization tree at stage k, where each leaf node estimates the cost using 5000 Monte-Carlo simulated trajectories. Each trajectory applies the greedy base policy $t = 10$ times before truncation. We set the terminal cost approximation $\hat{J} = |\overline{\mathbf{r}}_{k+t+1}|$; the remaining number of outstanding requests at the time of truncation.

D. GNN offline policy approximation architecture

We use the following setup to train the GNN architectures. We use Adam optimizer [40] with a learning rate of 0.005 and 0.002 for pickup and move networks, respectively, and a regularizing factor of 10^{-5} for generalization. We use 1.2×10^6 random state-(rollout) control pairs and 100 epochs. It took 16 hours to train each move network and 6 hours to train each pickup network on a single NVIDIA RTX A6000.

E. Parameters for Ambiguity Sets

To demonstrate adaptivity of our method, we introduce variability in the current demand model for p_η. We set the diameter B of $\tilde{p}_\eta$ as 6 (since there are between 0 and 6 requests per minute for the map section in Fig. 1 for the San Francisco taxicab dataset [30]). Since we use 5000 samples from the historical distribution to approximate the expectation during training of the offline approximation, we set $X = 5000$. Using these values, we set $q = 0.54$ and obtain that the minimum q-valid radius is 0.114. We choose q to be a little higher than 50% to guarantee that the sets are as small as possible, while still having a higher probability of containing the current demand distribution inside the set. We choose representative historical distributions that are centered around expected demand values with high relative frequencies in the historical data (See Fig. 4 for an example).

F. Benchmarks

In this section, we discuss benchmarks from relevant OR methods for our comparison study.

Instantaneous assignment: This algorithm, inspired by BLE [41] for iterative task assignment, performs a deterministic matching of outstanding requests and available taxis. This method does not consider future requests.

Two-step stochastic optimization (TSS): This multiagent task assignment algorithm [18] performs a maximization of the combined reward (negative wait time) of assigning taxicabs to requests using the set of currently outstanding requests and possible requests at the next stage predicted using the current demand model. We use 1000 sets of request samples to estimate the requests at the next stage.

Oracle: This method [18] solves an optimal multiagent task assignment using the *full knowledge of the current and all future requests* in the planning horizon by maximizing the combined reward (negative wait time) of all assignments. This method is only used as a lower bound on the cost assuming we had full a-priori information about the pickup/dropoff locations and request arrival times for all requests.

G. Numerical performance for taxicab pickup problem

Now, we present a comparative study of online play with the benchmark methods, and different components of online play, including greedy policy and one-at-a-time rollout from Sec. IV-C, and GNN policy approximation from Sec. IV-D. All results in this section use average wait time over 50 random starting states, and they are expressed using min-max normalization. This normalization transforms the average total wait time of a policy π over all evaluation

states at a given demand from $\bar{J}_\pi$ minutes to $\bar{J}_\pi^{norm} \in [0,1]$, where $\bar{J}_\pi^{norm} = (\bar{J}_\pi - min_{\pi' \in \Pi_e} \bar{J}_{\pi'})/(max_{\pi' \in \Pi_e} \bar{J}_{\pi'} - min_{\pi' \in \Pi_e} \bar{J}_{\pi'})$. Here, $\Pi_e \subseteq$ {Greedy policy, rollout, GNN, online play, instantaneous assignment, TSS, oracle}.

First, we are interested in the performance of online play when the current demand agrees with the historical demand used for training the offline approximation. Table I shows that online play outperforms all the other methods.

TABLE I

NORMALIZED WAIT TIME WITH GNN APPROXIMATION TRAINED ON THE SAME DEMAND DISTRIBUTION USED FOR EVALUATION

Policies	Normalized wait time for demand models: Low (L)	Medium (M)	High (H)
Greedy policy	0.94	0.99	1.0
One-at-a-time rollout	0.62	0.65	0.58
GNN	0.58	0.68	0.65
Online play w. GNN	**0.57**	**0.62**	**0.5**
Inst. assign.	1.0	0.92	0.88
TSS	0.96	1.0	0.89
Oracle	0.0	0.0	0.0
Min/Max for normalization (in minutes)	2.5/ 13.0	8.9/ 42.4	219.1/ 276.5

Second, we empirically show the robustness and limitations of the online play with an offline trained approximation by evaluating on out-of-distribution (OOD) demand. OOD demand corresponds to the case when the current demand model deviates from the historical demand model used for training. In Table II, we show that online play outperforms all the other methods for distributions that are inside the q-valid radius (0.114) of its Wasserstein Ambiguity Set. We also show the eventual performance degradation of online play for distributions outside the q-valid radius.

TABLE II

NORMALIZED WAIT TIME FOR DIFFERENT DEMAND DISTRIBUTIONS

Policies	Wasserstein Distance between training and evaluation model: Within q-valid radius			Outside q-valid radius		
	0.0	0.017	0.067	0.117	0.15	0.35
Greedy policy	1.0	1.0	1.0	1.0	1.0	0.77
One-at-a-time rollout	0.66	0.58	0.64	**0.62**	**0.6**	**0.46**
GNN trained on low demand	0.62	0.62	0.73	0.74	0.98	1.0
Online play w. GNN trained on low demand	**0.61**	**0.57**	**0.63**	0.73	0.68	0.52
Oracle	0.0	0.0	0.0	0.0	0.0	0.0
Min/Max for normalization (in minutes)	2.5 / 12.4	2.2 / 16.7	3.7 / 29.4	15.3 / 55.3	20.4 / 62.6	201.3 / 289.9

Once the current demand model is outside the q-valid radius of the historical demand model, our approach recovers the performance gain over rollout by exchanging the original policy approximation for one trained on a distribution that contains the current demand model inside its ambiguity set. Fig. 6 shows that switching to a better policy approximation gives a lower cost for online play (9% relative improvement).

H. Scalability

In this section, we present the numerical results in a map of $1500 \times 1500m^2$ (825 nodes and 1884 edges) with $m = 15$ agents (with a state-space size of 10^{459}, and a control-space size of 10^{12}). Increasing the map size linearly increases the size of the input feature for the GNN approximation. Efficiently tackling this curse of dimensionality prove to be a hard task, and hence the focus of future work.

The online optimization aspect of our approach can be used for bigger setups since rollout does online replanning

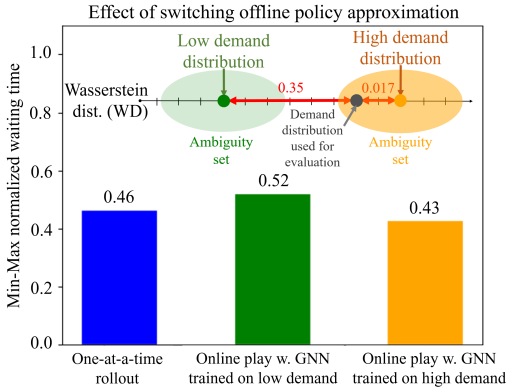

Fig. 6. Performance of online play evaluated on a distribution that lies inside of high demand ambiguity set and outside of low demand ambiguity set. The minimum and maximum values used for normalization are 201.6 and 289.9, respectively.

automatically and does not rely on the GNN approximation. To enable our approach to scale up, we consider the same one-agent-at-a-time one-step lookahead optimization (proposed in [31] [32]), with the lookahead step being fully stochastic. We estimate the cost approximation of the Q-factors at each leaf node of the lookahead tree by applying truncated rollout with a Certainty Equivalence [42] approximation. We use instantaneous assignment using an auction algorithm as suggested in [11] [43] [44] as the base policy, allowing our online optimization scheme to leverage the matching solution as the starting point for the optimization. In our implementation of the Certainty Equivalence approximation, we fix the disturbances η, ρ, and δ across all the rollout steps, only preserving the stochasticity of the order in which requests arrive and the pairing between pickup and dropoff locations for each request. Reducing uncertainty in disturbances enables us to use fewer (2000) Monte-Carlo simulations per leaf node.

Table III shows normalized results averaged over 50 trajectories. For our experiments, we evaluate our method on three different demand models: low, medium, and high, with the same $\tilde{p}_{\rho,c}$ and $\tilde{p}_{\delta,c}$, but with different $\tilde{p}_{\eta,c}$. The low, medium and high demand models have $E[\eta] \cdot N$ of $15, 45$ and 75, respectively. Table III shows that our approach outperforms the greedy policy and the instantaneous assignment baseline. We do not include results for TSS in this larger map, due to its prohibitively long runtime.

TABLE III

MIN/MAX NORMALIZED WAIT TIME FOR BIGGER SETUP

Demand	Policies: Greedy	Inst. assign.	Our method	Oracle	Min / Max
Low	1.0	0.86	**0.77**	0.0	9.9 / 137.2
Medium	1.0	0.86	**0.83**	0.0	207.1 / 502.3
High	1.0	0.99	**0.89**	0.0	684.4 / 995.4

V. CONCLUSION

In this paper, we apply online play in combination with offline policy approximations and verify that our approach allows the system to adapt to changes in the underlying demand conditions. Our future work includes, but is not limited to, considerations of travel time between intersections, time allocations for servicing vehicles, and predicting future demand to switch policy approximations proactively.

REFERENCES

[1] G. Berbeglia, J.-F. Cordeau, and G. Laporte, "Dynamic pickup and delivery problems," *European Journal of Operational Research*, vol. 202, no. 1, pp. 8–15, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221709002999

[2] S. Choudhury, K. Solovey, M. J. Kochenderfer, and M. Pavone, "Efficient large-scale multi-drone delivery using transit networks," *Journal of Artificial Intelligence Research*, vol. 70, pp. 757–788, 2021.

[3] B. Arbanas, A. Ivanovic, M. Car, T. Haus, M. Orsag, T. Petrovic, and S. Bogdan, "Aerial-ground robotic system for autonomous delivery tasks," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5463–5468.

[4] S. Zhang, C. Markos, and J. J. Q. Yu, "Autonomous vehicle intelligent system: Joint ride-sharing and parcel delivery strategy," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2022.

[5] W. Emanuelsson, A. P. Riveiros, Y. Li, K. H. Johansson, and J. Mårtensson, "Multiagent rollout with reshuffling for warehouse robots path planning," 2022. [Online]. Available: https://arxiv.org/abs/2211.08201

[6] M. Tsao, D. Milojevic, C. Ruch, M. Salazar, E. Frazzoli, and M. Pavone, "Model predictive control of ride-sharing autonomous mobility-on-demand systems," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6665–6671.

[7] G. Guo and Y. Xu, "A deep reinforcement learning approach to ride-sharing vehicle dispatching in autonomous mobility-on-demand systems," *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 1, pp. 128–140, 2022.

[8] B. Li, N. Ammar, P. Tiwari, and H. Peng, "Decentralized ride-sharing of shared autonomous vehicles using graph neural network-based reinforcement learning," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 912–918.

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

[10] M. Gueriau, F. Cugurullo, R. A. Acheampong, and I. Dusparic, "Shared autonomous mobility on demand: A learning-based approach and its performance in the presence of traffic congestion," *IEEE Intelligent Transportation Systems Magazine*, vol. 12, no. 4, pp. 208–218, 2020.

[11] D. Bertsekas, "A distributed algorithm for the assignment problem," *Lab. for Information and Decision Systems Report*, 05 1979.

[12] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, "An optimal algorithm for on-line bipartite matching," in *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, ser. STOC '90. New York, NY, USA: Association for Computing Machinery, 1990, p. 352–358. [Online]. Available: https://doi.org/10.1145/100216.100262

[13] R. Duan and S. Pettie, "Linear-time approximation for maximum weight matching," *J. ACM*, vol. 61, no. 1, 2014. [Online]. Available: https://doi.org/10.1145/2529989

[14] D. Bertsimas, P. Jaillet, and S. Martin, "Online vehicle routing: The edge of optimization in large-scale applications," *Oper. Res.*, vol. 67, pp. 143–162, 2019.

[15] G. A. Croes, "A method for solving traveling-salesman problems," *Operations Research*, vol. 6, no. 6, pp. 791–812, 1958. [Online]. Available: http://www.jstor.org/stable/167074

[16] M. Yannakakis, C. A. Tovey, J. H. M. Korst, and J. H. M. J. M. van Laarhoven, *Local Search in Combinatorial Optimization*. Princeton University Press, 2003. [Online]. Available: http://www.jstor.org/stable/j.ctv346t9c

[17] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.

[18] M. Lowalekar, P. Varakantham, and P. Jaillet, "Online spatio-temporal matching in stochastic and dynamic domains," *Artificial Intelligence*, vol. 261, pp. 71–112, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370218302030

[19] M. W. Ulmer, J. C. Goodson, D. C. Mattfeld, and M. Hennig, "Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests," *Transportation Science*, vol. 53, no. 1, pp. 185–202, 2019. [Online]. Available: https://doi.org/10.1287/trsc.2017.0767

[20] N. Parvez Farazi, B. Zou, T. Ahamed, and L. Barua, "Deep reinforcement learning in transportation research: A review," *Transportation Research Interdisciplinary Perspectives*, vol. 11, p. 100425, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590198221001317

[21] T. Ahamed, B. Zou, N. P. Farazi, and T. Tulabandhula, "Deep reinforcement learning for crowdsourced urban delivery," *Transportation Research Part B: Methodological*, vol. 152, pp. 227–257, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261521001636

[22] C. Wu, K. Shankari, E. Kamar, R. Katz, D. Culler, C. Papadimitriou, E. Horvitz, and A. Bayen, "Optimizing the diamond lane: A more tractable carpool problem and algorithms," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 1389–1396.

[23] D. Silver and J. Veness, "Monte-Carlo Planning in Large POMDPs," in *Proc. 23rd International Conf. on NeurIPS*, Red Hook, NY, USA, 2010, pp. 2164–2172.

[24] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization," in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/c2aee86157b4a40b78132f1e71a9e6f1-Paper.pdf

[25] R. W. Bent and P. Van Hentenryck, "Scenario-based planning for partially dynamic vehicle routing with stochastic customers," *Operations Research*, vol. 52, no. 6, pp. 977–987, 2004. [Online]. Available: https://doi.org/10.1287/opre.1040.0124

[26] D. Bertsekas, *Rollout, Policy Iteration, and Distributed Reinforcement Learning*, ser. Athena scientific optimization and computation series. Athena Scientific., 2020. [Online]. Available: https://books.google.com/books?id=Hbo-EAAAQBAJ

[27] ——, *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control*. Nashua, NH, USA: Athena Scientific, 2022.

[28] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[29] P. M. Esfahani and D. Kuhn, "Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations," 2015. [Online]. Available: https://arxiv.org/abs/1505.05116

[30] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD dataset epfl/mobility (v. 2009-02-24)," Downloaded from https://crawdad.org/epfl/mobility/20090224, Feb. 2009.

[31] D. Bertsekas, "Multiagent value iteration algorithms in dynamic programming and reinforcement learning," *Results in Control and Optimization*, vol. 1, p. 100003, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666720720300035

[32] ——, "Multiagent reinforcement learning: Rollout and policy iteration," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 2, pp. 249–272, 2021.

[33] ——, *Reinforcement Learning and Optimal Control*, ser. Athena Scientific optimization and computation series. Athena Scientific, 2019. [Online]. Available: https://books.google.com/books?id=2f85EAAAQBAJ

[34] C. Villani, *Optimal transport: old and new*. Springer, 2009, vol. 338.

[35] R. Ji and M. Lejeune, "Data-driven optimization of reward-risk ratio measures," *INFORMS Journal on Computing*, vol. 33, 11 2020.

[36] S. Bhattacharya, S. Kailas, S. Badyal, S. Gil, and D. Bertsekas, "Multiagent rollout and policy iteration for pomdp with application to multi-robot repair problems," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 1814–1828. [Online]. Available: https://proceedings.mlr.press/v155/bhattacharya21a.html

[37] S. Bhattacharya, S. Badyal, T. Wheeler, S. Gil, and D. Bertsekas, "Reinforcement learning for pomdp: Partitioned rollout and policy iteration with application to autonomous sequential repair problems," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3967–3974, 2020.

[38] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016. [Online]. Available: http://arxiv.org/abs/1609.02907

[39] M. Azram, F. Elfaki, and J. Daoud, "Classification of atoms," *Australian Journal of Basic and Applied Sciences*, vol. 5, pp. 5–8, 05 2011.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[41] B. B. Werger and M. J. Matarić, "Broadcast of local eligibility for multi-target observation," in *Distributed Autonomous Robotic Systems 4*. Springer, 2000, pp. 347–356.
[42] D. Bertsekas and D. Castanon, "Rollout algorithms for stochastic scheduling problems," in *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*, vol. 2, 1998, pp. 2143–2148 vol.2.
[43] D. Bertsekas, *Network Optimization: Continuous and Discrete Models*, ser. Athena scientific optimization and computation series. Athena Scientific, 1998. [Online]. Available: https://books.google.com/books?id=qUUxEAAAQBAJ
[44] ——, "Constrained multiagent rollout and multidimensional assignment with the auction algorithm," 2020. [Online]. Available: https://arxiv.org/abs/2002.07407