Surge Routing: Event-informed Multiagent Reinforcement Learning for Autonomous Rideshare

Daniel Garces
Harvard University
Boston, MA, United States
dgarces@g.harvard.edu

Stephanie Gil
Harvard University
Boston, MA, United States
sgil@seas.harvard.edu

ABSTRACT

Large events such as conferences, concerts and sports games, often cause surges in demand for ride services that are not captured in average demand patterns, posing unique challenges for routing algorithms. We propose a learning framework for an autonomous fleet of taxis that leverages event data from the internet to predict demand surges and generate cooperative routing policies. We achieve this through a combination of two major components: (i) a demand prediction framework that uses textual event information in the form of events' descriptions and reviews to predict event-driven demand surges over street intersections, and (ii) a scalable multiagent reinforcement learning framework that leverages demand predictions and uses one-agent-at-a-time rollout, combined with limited sampling certainty equivalence, to learn intersection-level routing policies. For our experimental results we consider real NYC ride share data for the year 2022 and information for more than 2000 events across 300 unique venues in Manhattan. We test our approach with a fleet of 100 taxis on a map with 2235 street intersections. Our experimental results demonstrate that our method learns routing policies that reduce wait time overhead per serviced request by 25% to 75%, while picking up 1% to 4% more requests than other model-based RL frameworks and classical methods from operations research.

KEYWORDS

Multiagent Reinforcement Learning; Autonomous Vehicle Routing; Language-informed Demand Prediction.

ACM Reference Format:
Daniel Garces and Stephanie Gil. 2024. Surge Routing: Event-informed Multiagent Reinforcement Learning for Autonomous Rideshare. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 10 pages.

1 INTRODUCTION

Large events such as conferences, concerts, and sports games lead to large agglomerations of people and hence tend to produce demand surges for ride services. Efficiently servicing these surges requires the orchestration of fleet-wise coordinated plans that leverage accurate estimates of event-driven demand fluctuations (see Fig 1). Unfortunately, most on-demand mobility routing algorithms in the literature do not address event-driven demand surges, either because they do not consider potential future requests and thus do not plan ahead [5, 11, 21, 24, 33, 41, 61], or because they plan ahead using demand models that are based on averaged or short-term history and hence do not account for point disturbances associated with event-related surges [1, 16, 17, 23, 28, 30, 44, 48, 58].

The plethora of event data freely available on the internet and recent advances in language models allow for the development of event-informed demand prediction mechanisms [38, 39, 51]. Additionally, recent advances in multiagent Reinforcement Learning (RL) [26, 29], including rollout-based mechanisms [7, 9, 30], allow for learning multiagent cooperative plans. Ideally, demand prediction mechanisms and RL-based routing algorithms could be combined to obtain a fine-grained event-informed multiagent routing framework that predicts event-driven demand surges and routes taxis accordingly. However, designing and implementing such a framework for a city-scale application is a non-trivial task. Tackling this task requires addressing two major challenges: (i) demand predictions should leverage data from multiple events to generate accurate estimates of future demand, that additionally, are usable by intersection-level multiagent RL routing algorithms and (ii) the expected cost of agents' actions should be approximated by the multiagent RL routing algorithm at a city-scale without incurring prohibitively long execution times.

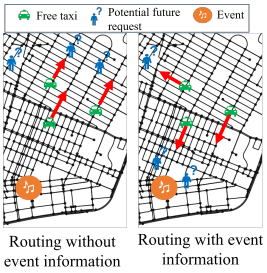

Figure 1: Motivating example

In this paper we address these two challenges by introducing a multiagent routing framework that leverages event-informed hourly demand predictions to learn cooperative routing policies on a city-scale environment. To our knowledge, our proposed method is the first work to integrate event-driven surge demand prediction and intersection-level multiagent routing. Our approach solves the first challenge by leveraging an unsupervised aggregation mechanism that we develop. This mechanism aggregates data from several overlapping events over city blocks (or "sectors") to generate sector-level demand predictions. These demand predictions are then mapped to intersections using a probabilistic assignment mechanism that we design. In this way, demand predictions over intersections can then be used to inform future cost estimation for our RL routing algorithm. Since the state space for a city-scale application is very large, standard rollout methods [7] tend to need

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

a large number of samples to approximate the expected future costs of actions. In order to make rollout algorithms applicable to these large settings, we consider a certainty equivalence approximation, where the demand distribution is replaced by a semi-deterministic mean value. We also use a limited sampling modification to the standard certainty equivalence to reduce the sampling space to the surrounding area of a given sector when estimating future costs for an agent inside that sector. This modification allows our method to reduce the number of samples required for estimating the expected costs of actions while still maintaining similar accuracy, successfully addressing the second challenge. We are able to maintain accuracy since taxis that are very far away from a sector can't reach the sector during their planning horizon and hence have very little effect on that sector's short-term planning.

More specifically, our proposed framework is composed of four modules as shown in Fig. 2: 1) the event processing module, 2) the demand prediction module, 3) the demand assignment module, and 4) the model-based RL routing module. The event processing module captures event information from the internet in the form of event reviews and descriptions by leveraging sentence embeddings [3, 14, 20, 36, 56, 60] generated from a pre-trained Masked Language Model (MLM). Sentence embeddings for events in overlapping sectors of the map are aggregated using an unsupervised aggregation scheme. This aggregation scheme combines spectral clustering [22, 25, 63] and graph summarization [27, 64] to produce representative dense vectors for each sector. These dense vectors are then used by the demand prediction module to predict hourly demand for the taxi service at each sector. In order to integrate the demand prediction into the multiagent RL-based routing mechanism, we propose a novel demand assignment module. This module uses a probabilistic assignment routine that leverages locale maximum occupancy data and occupancy schedules (percentage of maximum occupancy)[2, 19, 31, 47] to map from demand predictions over sectors to demand predictions over street intersections. These demand predictions over intersections are then used by the multiagent routing scheme to estimate expected future costs for each agent's actions. Our proposed multiagent routing scheme builds off one-agent-at-a-time rollout [6, 8, 9, 30] and combines it with a limited certainty equivalence approximation to reduce the sampling complexity of estimating expected future costs.

We test our approach using real NYC's Taxi and Limousine High-Volume For-Hire-Vehicle (HV-FHV) data [18] and information for more than 2000 events across 300 venues. We consider a region of Manhattan with 2235 intersections and a fleet of 100 autonomous taxis. This setup has 3X more intersections and 6X more taxis than previous work in this area [30], demonstrating the scalability of our approach. We empirically demonstrate that our event-informed framework learns routing policies that reduce wait time overhead per serviced request by 25% to 75% while picking up 1% to 4% more requests than other model-based RL frameworks and other classical algorithms in operations research. [1]

[1]We define wait time overhead as the additional time that a request will have to wait in a realistic stochastic setting. In contrast to a setting in which the locations and entry times of all requests are known a-priori, taxis cannot be routed to the exact locations of requests from the beginning of the episode.

2 RELATED WORKS

In this section, we review the current state-of-the-art for the two problems related to event-informed multiagent routing: Dynamic Vehicle Routing (DVR), and demand prediction.

Dynamic Vehicle Routing. Earlier works tackled this problem using instantaneous assignment approaches [5, 11, 24, 33], and routing heuristics, [21, 41, 61]. Instantaneous assignment approaches, however, produce myopic policies since they do not consider potential future requests. Sampling-based stochastic optimization [37] tries to solve this issue, but incurs long computation times due to the multistep planning objective and large state space. To improve computation times, several authors considered offline trained approximations [1, 23, 44, 48, 58]. Offline learning methods tend to be computationally faster at inference time, but they do not generalize to unknown scenarios not represented in the training data, and tend to not scale well as the state space becomes larger. This makes them infeasible for deployment in city-scale urban environments, where the state space is very large and new changes in demand are not necessarily represented in the historical demand data. To try to address this issue, and allow policies to adapt to changes in the demand, other authors have considered online optimization methods [4, 7, 8, 54], queuing theory [55, 57, 62], and hybrid approaches [30]. These online learning methods, however, do not consider event-driven surges in demand, and hence they tend to not be applicable to realistic urban settings. In this paper we aim to address this limitation by integrating event-informed demand prediction into model-based RL routing.

Demand Prediction. Some authors have considered time series analysis techniques [34, 40, 42]. These approaches, however, rely on the data being stationary or following predictable seasonal changes that can easily be removed by seasonal-differencing. Demand for transportation services, however, tends to be highly dynamic and it is affected by external events like concerts or traffic accidents. This situation prevents time series analysis methods from performing as expected. To address this shortcoming, various authors have considered learning approaches [13, 15, 35, 45] to predict demand based on spatial and temporal features (time of day, weather conditions, etc.). However, these methods generally do not employ information about events, and hence do not generalize to scenarios where events cause demand surges. Other authors also look at predicting demand using spatial and temporal features, as well as event information [38, 39, 51]. One limitation of these methods that we address in this paper is the ability to include information from an arbitrary number of events over overlapping regions of the map. The ability of our method to aggregate event information makes it applicable to city-scale environments where there are multiple venues hosting simultaneous events in the same region of the map.

3 PROBLEM FORMULATION

In this section, we present the formulation for our multiagent taxicab routing problem, casting it as a city-scale discrete time, finite horizon, stochastic Dynamic Programming (DP) problem. We define the environment, requests, state and control space, the basics of rollout and our problem of interest in the following subsections.

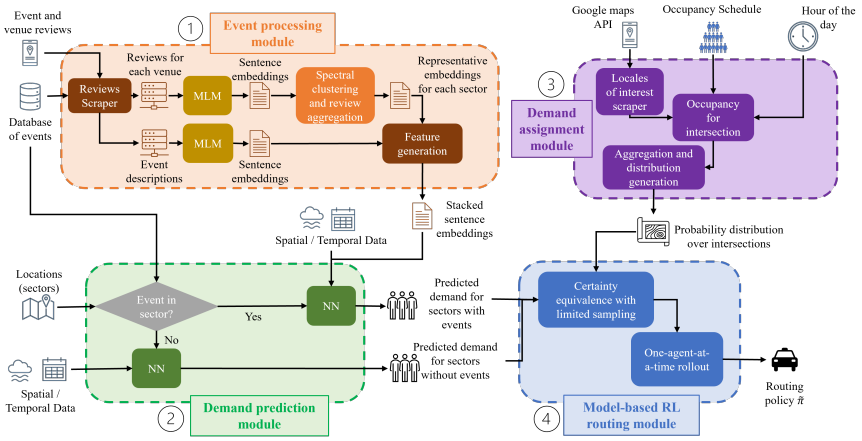

Figure 2: General System Overview showing our proposed approach's four modules: 1) the event processing module, 2) the demand prediction module, 3) the demand assignment module, and 4) the model-based RL routing module.

3.1 Environment

We assume that taxicabs are deployed in an urban environment with a fixed street topology (see Fig. 3). The environment is expressed as a graph $G = (V, E)$, where $V = \{1, \ldots, n\}$ corresponds to the set of intersections in the map numbered 1 through n, while $E \subseteq \{(i, j)|i, j \in V\}$ corresponds to the set of directed streets that connect intersections i and j in the map. The set of adjacent intersections to intersection i is denoted as $\mathcal{N}_i = \{j|j \in V, (i, j) \in E\}$. We also assume that the environment is divided into sectors $s_k \subseteq V$, such that $V = \bigcup_k s_k$ and $s_k \cap s_h = \emptyset, \forall k \neq h$. We denote the set of all sectors in the map as the set S, where S is then a set of sets. We define $\mathcal{A}(s_k) \subseteq S$ as the set of all the sectors adjacent to sector s_k. We define $\mathcal{I}_{s_k} \subseteq s_k$ as the set of intersections in sector s_k that can be used as pickup or drop-off locations, following local regulations. We denote the set of intersections that can be used for pickup or drop-off of requests over the entire map as $\mathcal{I}_V = \bigcup_{s_k \in S} \mathcal{I}_{s_k}$.

3.2 Requests

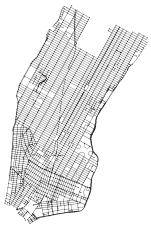

Figure 3: Our NYC environment with 2235 intersections

We define a request r for the ride service as a tuple $r = \langle \rho_r, \delta_r, t_r, \phi_r \rangle$, where $\rho_r \in \mathcal{I}_V$ and $\delta_r \in \mathcal{I}_V$ correspond to the nearest available intersection to the request's desired pickup and drop-off locations, respectively; t_r corresponds to the time at which the request entered the system; and $\phi_r \in \{0, 1\}$ is an indicator, such that $\phi_r = 1$ if the request has been picked up by a vehicle, $\phi_r = 0$ otherwise. We model the number of new pickup requests for a specific sector s_k as a random variable η_{s_k} with an unknown underlying distribution $p_{\eta_{s_k}}$. Its estimation is denoted as $\tilde{p}_{\eta_{s_k}}$. We denote the realization of η_{s_k} at time t as $\eta_{s_k}(t)$. We denote the set of new pickup requests at time t for a specific sector s_k as $\mathbf{r_{s_k,t}} = \{r|\rho_r \in \mathcal{I}_{s_k}, t_r = t\}$, such that $|\mathbf{r_{s_k,t}}| = \eta_{s_k}(t)$. We model the exact pickup intersections for an arbitrary request given that the request originated at sector s_k as the random variable ρ_{s_k} with support $\mathcal{I}_{s_k}$. Similarly, we model the exact drop-off intersection for an arbitrary request given that the request will be dropped off at sector s_k as the random variable δ_{s_k} with support $\mathcal{I}_{s_k}$. Both of these random variables have unknown underlying distributions that we denote as $p_{\rho_{s_k}}$ and $p_{\delta_{s_k}}$, respectively. We also denote their estimated probability distributions as $\tilde{p}_{\rho_{s_k}}$ and $\tilde{p}_{\delta_{s_k}}$, respectively. We model the drop-off sector for an arbitrary request given that the request has pick up sector s_k as the random variable β_{s_k} with an unknown probability distribution $p_{\beta_{s_k}}$. We denote its estimated probability distribution as $\tilde{p}_{\beta_{s_k}}$. We denote the demand model for a given sector s_k as the set of random variables $D_{s_k} = \{\eta_{s_k}, \rho_{s_k}, \delta_{s_k}, \beta_{s_k}\}$. We denote the global demand model as $\mathcal{D} = \bigcup_{s_k \in S} D_{s_k}$. We define $\bar{\mathbf{r}}_{\mathbf{s_k,t}} = \{r|r \in \mathbf{r_{s_k,t'}}, \phi_r = 0, t' = 1, \ldots t\}$ as the set of outstanding pickup requests for sector s_k that have not been picked up by any taxi till time t. We denote the set of all outstanding pickup requests at time t as $\bar{\mathbf{r}}_\mathbf{t} = \bigcup_{s_k \in S} \bar{\mathbf{r}}_{(\mathbf{s_k,t})}$.

3.3 State Representation and Control Space

We assume there is a total of m agents and all agents can perfectly observe all available requests, and all other agents' locations and occupancy status. We assume that this m is fixed over time. We represent the state of the system at time t as a tuple $x_t = \langle \vec{v}_t, \vec{\tau}_t, \bar{\mathbf{r}}_\mathbf{t} \rangle$. The vector $\vec{v}_t = [v_t^1, \ldots, v_t^m]$ contains the locations for all m agents at time t, where $v_t^\ell \in V$ corresponds to the closest intersection to the geographical position of agent ℓ. The vector $\vec{\tau}_t = [\tau_t^1, \ldots, \tau_t^m]$ contains the time remaining in current trip for all m agents. If agent ℓ is available, then it has not picked up a request and $\tau_t^\ell = 0$, otherwise $\tau_t^\ell \in \mathbb{N}^+$.

We denote the control space for agent ℓ at time t as $\mathbf{U}_t^\ell(x_t)$. If the agent is available (i.e. $\tau_t^\ell = 0$), then $\mathbf{U}_t^\ell(x_t) = \{\mathcal{N}_{v_t^\ell}, v_t^\ell, \psi\}$, where $\mathcal{N}_{v_t^\ell}$ corresponds to the set of adjacent intersections to the current location v_t^ℓ, and ψ corresponds to a special pickup control that becomes available if there is a request $r \in \bar{\mathbf{r}}_\mathbf{t}$ a the location of agent ℓ such that $\rho_r = v_t^\ell$. On the other hand, if the agent is currently servicing a request r (i.e. $\tau_t^\ell > 0$), then $\mathbf{U}_t^\ell(x_t) = \{\zeta\}$, where ζ corresponds to the next hop in Dijkstra's shortest path between agent ℓ's current location v_t^ℓ and the destination of the

request δ_r. We denote all possible controls for all m agents at state x_t as $\mathbf{U}_t(x_t) = \mathbf{U}_t^1(x_t) \times \cdots \times \mathbf{U}_t^m(x_t)$.

3.4 Rollout-based Routing

We are interested in learning a cooperative pickup and routing policy that minimizes the total wait for all passengers over a finite time horizon of length N. We denote the state transition function as f, such that $x_{t+1} = f_t(x_t, u_t, \mathcal{D})$, where x_{t+1} is the resulting state after control $u_t \in \mathbf{U}_t(x_t)$ has been applied from state x_t considering the realizations for all random variables in $\mathcal{D}$. We define the stage cost $g_t(x_t, u_t, \mathcal{D}) = |\overline{\mathbf{r_t}}|$ as the number of outstanding requests over the entire map at time t. We define a policy $\pi = \{\mu_1, \dots \mu_N\}$ as a set of functions that maps state x_t into control $u_t = \mu_t(x_t) \in \mathbf{U}_t(x_t)$, with its cost given by $J_\pi(x_1) = E\left[g_N(x_N) + \sum_{t=1}^{N-1} g_t(x_t, \mu_t(x_t), \mathcal{D})\right]$, where $g_N(x_N) = |\overline{\mathbf{r_N}}|$ is the terminal cost. Since the control space is a Cartesian product and hence it grows exponentially with the number of agents, searching the entire control space to find the optimal policy is computationally intractable. For this reason, we consider policy improvement schemes, such as rollout [6, 8], that allows us to obtain a lower cost policy by improving upon a base policy with a reasonable initial behavior. We define a base policy $\bar{\pi} = \{\bar{\mu}_1, \dots \bar{\mu}_N\}$ as an easy to compute heuristic that is given. Our objective becomes then to find an approximate policy $\tilde{\pi} = \{\tilde{\mu}_1, \dots \tilde{\mu}_N\}$, such that given base policy $\bar{\pi}$, the minimizing action for state x_t at time t is given by:

$$\tilde{\mu}_t(x_t) \in \arg\min_{u_t \in \mathbf{U}_t(x_t)} E_{\mathcal{D}}\left[g_t(x_t, u_t, \mathcal{D}) + \tilde{J}_{\bar{\pi},t}(x_{t+1})\right] \quad (1)$$

Where $\tilde{J}_{\bar{\pi},t}(x_{t+1}) = \hat{J}_T + \sum_{t'=(t+1)}^{T} g_{t'}(x_{t'}, \bar{\mu}_{t'}(x_{t'}), \mathcal{D})$ is a cost approximation derived from applying the base policy $\bar{\pi}$ for H time steps from state x_{t+1} with a terminal cost approximation $\hat{J}_T = |\overline{\mathbf{r_T}}|$, where $T = (t+1) + H$. The expectation in eq. 1 is estimated using Monte-Carlo simulations.

3.5 Problem

Recent success of rollout methods in combinatorial applications [7, 9, 12], including small scale routing [30], makes these rollout-based methods a natural choice for tackling city-scale routing. However, applying rollout methods to a city-scale environment with event-driven demand surges is a non-trivial task. To integrate the rollout formulation presented in Sec. 3.4 into our city-scale application, we need to solve two Problems.

Problem 1: We must estimate the probability distributions of the random variables that specify the demand model $\mathcal{D}$ to obtain $\tilde{\mathcal{D}} = \bigcup_{s_k \in S} \{\tilde{\eta}_{s_k}, \tilde{\rho}_{s_k}, \tilde{\delta}_{s_k}, \tilde{\beta}_{s_k}\}$ with underlying distributions $\tilde{p}_{\eta_{s_k}}$, $\tilde{p}_{\rho_{s_k}}$, $\tilde{p}_{\delta_{s_k}}$, and $\tilde{p}_{\beta_{s_k}}$ that capture demand surges produced by events.

Problem 2: We must reduce the computational cost of approximating the expectation Eq. 1, for the multiagent case, in order to make rollout-based methods amenable to city-scale environments.

4 OUR APPROACH

We first provide a brief overview of our method, but details of all the components are described in the following subsections. To solve **Problem 1** in Sec. 3.5, we need to derive estimates $\tilde{p}_{\eta_{s_k}}$, $\tilde{p}_{\rho_{s_k}}$, $\tilde{p}_{\delta_{s_k}}$, and $\tilde{p}_{\beta_{s_k}}$. To estimate $\tilde{p}_{\eta_{s_k}}$, we develop a novel *event-informed demand estimation procedure* that leverages sentence embeddings [36] and spectral clustering techniques[22] to generate vector representations for events. We propose a demand prediction scheme that predicts the number of requests that will enter the system at each sector $s_k \in S$ using these vector representations. Finally, We build off of the idea of Certainty Equivalence [10] to obtain a semi-deterministic approximation for $\tilde{p}_{\eta_{s_k}}$ by uniformly distributing the predicted number of requests for sector s_k over the entire time horizon N. To estimate $\tilde{p}_{\rho_{s_k}}$ and $\tilde{p}_{\delta_{s_k}}$, we leverage occupancy schedules [2, 19] to derive a novel probabilistic method that maps demand over a sector s_k to individual intersections $j \in \mathcal{I}_{s_k}$. To estimate $\tilde{p}_{\beta_{s_k}}$, we use the relative frequency of pickup and drop-off sectors in historical data conditioned on having pickup sector s_k. The combination of all these estimation procedures, allows our method to account for demand surges and transfer that information to our routing framework.

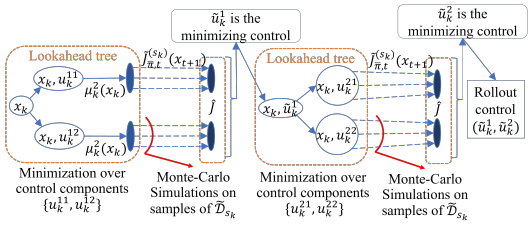

Figure 4: Example of one-agent-at-a-time rollout with two agents, where each agent only has two available actions.

To solve **Problem 2** in Sec. 3.5, we build off one-agent-at-a-time rollout [6, 8] and Certainty Equivalence [10]. We develop a novel rollout-based scalable routing framework that scales linearly with the number of agents and has a reduced sampling space. This allows our system to decrease the number of simulations required for the Monte-Carlo estimation of the expectation in Eq. 1. The combination of one-agent-at-a-time rollout and our novel limited Certainty Equivalence approximation makes our system capable of handling large taxi fleets over large maps without incurring prohibitively long computation times.

We will cover the specifics of our scalable routing framework first, in order to motivate the need for the estimation of $\tilde{p}_{\eta_{s_k}}$, $\tilde{p}_{\rho_{s_k}}$, $\tilde{p}_{\delta_{s_k}}$, and $\tilde{p}_{\beta_{s_k}}$.

4.1 Rollout-based Scalable Routing Framework

The proposed rollout-based scalable routing framework is mostly composed of the model-based RL module shown in Fig. 2 module 4, which leverages one-agent-at-a-time rollout [7],[12], combined with a limited sampling formulation derived from the idea of scenarios in certainty equivalence [10] to obtain policy $\tilde{\pi}$. We choose one-agent-at-a-time rollout as the foundation for our algorithm since the control space for this rollout variation scales linearly with the number of agents, instead of exponentially [7]. Under our proposed framework, we replace the minimization given in the problem statement (Eq. 1) for a one-agent-at-a-time formulation. More specifically, agent ℓ's one-at-a-time rollout control at state x_t, given that agent ℓ is at an intersection j that is inside sector s_k, is

then:

$$\tilde{u}_t^\ell \in \arg\min_{u_t^\ell \in \mathrm{U}_k^\ell(x_t)} E_{\tilde{\mathcal{D}}_{s_k}}\left[g_t(x_t, \bar{u}, \tilde{\mathcal{D}}_{s_k}) + \tilde{J}_{\bar{\pi},t}^{(s_k)}(x_{t+1})\right] \quad (2)$$

where $\bar{u} = (\tilde{u}_t^1, \ldots, \tilde{u}_t^{\ell-1}, u_t^\ell, \bar{\mu}_t^{\ell+1}(x_t), \ldots, \bar{\mu}_t^m(x_t))$; the *local* set $\tilde{\mathcal{D}}_{s_k} = \{\tilde{\eta}_{s_h}, \tilde{\rho}_{s_h}, \tilde{\delta}_{s_h}, \tilde{\beta}_{s_h} | \forall s_h \in \mathcal{A}(s_k) \bigcup \{s_k\}\}$ contains the random variables for demand in sectors s_h that are adjacent to sector s_k, where each random variable has estimated probability distributions derived from the *event-informed demand estimation procedure*; and $\tilde{J}_{\bar{\pi},t}^{(s_k)}(x_{t+1}) = \hat{J}_T + \sum_{t'=(t+1)}^{T} g_{t'}(x_{t'}, \bar{\mu}_{t'}(x_{t'}), \tilde{\mathcal{D}}_{s_k})$ corresponds to the cost approximation of executing the base policy for H steps, considering samples from the estimated demand distributions in $\tilde{\mathcal{D}}_{s_k}$, and a terminal cost approximation $\hat{J}_T = |\bar{\mathbf{r}}_\mathbf{T}|$ with $T = (t+1) + H$. The expectation in Eq. 2 is estimated using Monte-Carlo approximation. A graphical example of the proposed formulation is presented in Fig. 4. By considering only the current and adjacent sectors when obtaining samples for the Monte-Carlo approximation of the expected cost of each potential action for a given agent we are able to decrease the sample space and reduce the computational complexity of the sampling procedure for the estimation of the expectation. If we combine this approach with certainty equivalence scenarios [10], where we consider a finite number of representative scenarios instead of a full stochastic set of samples, the resulting system incurs lower execution times when approximating the expectation in Eq. 2, while still obtaining an accurate approximation.

4.2 Event-informed Demand Estimation

In this section we explain how we estimate the underlying probability distributions $\tilde{p}_{\eta_{s_k}}$, $\tilde{p}_{\rho_{s_k}}$, $\tilde{p}_{\delta_{s_k}}$, and $\tilde{p}_{\beta_{s_k}}$, which compose $\tilde{\mathcal{D}}_{s_k}$ the demand model for sector s_k and hence are needed to approximate the expectation in Eq. 2. This corresponds to modules 1-3 in in Fig. 2.

4.2.1 Estimating the Probability Distribution for the Number of Requests $\tilde{p}_{\eta_{s_k}}$. We estimate $\tilde{p}_{\eta_{s_k}}$ by leveraging sentence embeddings, spectral clustering and averaging, the demand prediction module, and a novel probabilistic approach that derives a semi-deterministic approximation for $\tilde{p}_{\eta_{s_k}}$ from the predicted demand, obtaining a minute by minute demand distribution from the hourly demand prediction.

Event Data and Sentence Embeddings. We need a list of events that includes the event's date, title, description, and venue. Such a list of events can be scraped from the internet using queries [38] or an online database [50]. Events' attendance depends on multiple factors, including the public's attitude towards such an event. For this reason, we try to estimate a proxy of the public's attitude towards an event by considering reviews for event-venues pairs. We collect Q reviews for each event-venue pair. For each review's textual excerpt we generate a d-dimensional sentence embedding using a pre-trained MLM. Sentence embeddings tend to capture semantic information and inter-word relations in a dense vector representation [14, 60]. We denote the q-th sentence embedding for a specific event as $\vec{e}_q \in \mathbb{R}^d$, and we denote the set of all Q embeddings as $\mathcal{E}$.

Spectral Clustering. For each event, we consider its associated Q review embeddings. We want to obtain semantically relevant representations that aggregate the reviews as dense vectors. To do this we consider spectral clustering on the latent space of the review embeddings. We assume that the latent space for the embeddings is approximately euclidean based on the results presented in [53]. For our specific application, since we are dealing with an euclidean space, we consider a Gaussian radial basis function as the similarity score $s(\vec{e}_q, \vec{e}_h) = \exp(-\gamma \cdot ||\vec{e}_q - \vec{e}_h||_2^2)$ with $\gamma = 1$. To execute spectral clustering, we construct a similarity graph $G_C = (V_C, E_C)$, where each vertex $v_q \in V_C$ corresponds to the review embedding $\vec{e}_q$. We consider a fully connected graph, where all vertices are connected to all other vertices, and the weight of each edge (v_q, v_h) is given by the similarity score $s(\vec{e}_q, \vec{e}_h)$. Using this complete graph, we then apply spectral clustering for b clusters using the algorithm proposed in Damle et al. [22]. We denote the cluster label assigned to embedding $\vec{e}_q$ as $\hat{b}(\vec{e}_q) \in \{1, \ldots, b\}$. We denote the set of embeddings that have cluster label a for $a \in \{1, \ldots, b\}$ as $\mathcal{B}_a = \{\vec{e}_q | \hat{b}(\vec{e}_q) = a, \vec{e}_q \in \mathcal{E}\}$.

Cluster Averaging. Once all reviews for an event are clustered, we average the embeddings for each assigned cluster to obtain semantically relevant averaged embedding $\hat{e}_a = \frac{\sum_{\vec{e}_q \in \mathcal{B}_a} \vec{e}_q}{|\mathcal{B}_a|}$ for each cluster label $a \in \{1, \ldots, b\}$. We then stack the resulting b embeddings to obtain a dense representation $R_w = [\hat{e}_1^\top, \ldots \hat{e}_b^\top]^\top$ for arbitrary event w. We denote the set of events in sector s_k as w_{s_k}. we denote the average dense representation for events in sector s_k as $\hat{R}_{s_k} = \frac{\sum_{w \in w_{s_k}} R_w}{|w_{s_k}|}$. For each event title and description we generate an additional sentence embedding using the same pre-trained MLM. This embedding provides additional context to the NN to differentiate between events that share the same venue and hence might have some of the same reviews. For arbitrary event w, we denote this embedding as $\vec{z}_w \in \mathbb{R}^d$. We denote the average title embedding for events in sector s_k as $\hat{z}_{s_k} = \frac{\sum_{w \in w_{s_k}} \vec{z}_w}{|w_{s_k}|}$. We define the final unified sector feature for an arbitrary sector s_k as $F_{s_k} = [\hat{z}_{s_k}^\top, \hat{R}_{s_k}^\top]^\top \in \mathbb{R}^{(b+1)\cdot d}$. The vector F_{s_k} is the output of the event processing module (see Fig. 2 module 1).

Demand Prediction Module. The demand prediction module is composed of a temporal (day of the week, month, and hour) and spatial (weather) data collection pipeline followed by a two NN prediction mechanism (see Fig. 2 module 2). If there are no events happening on sector s_k, the system considers a NN that takes as input only temporal and spatial data in the form of a vector $\vec{f}$. If there is an event on sector s_k, the system enhances $\vec{f}$ with the unified sector feature F_{s_k} from the *event processing module* to create input feature $F_{s_k}^+ = [\vec{f}^\top, F_{s_k}^\top]^\top$. For simplicity, we denote the predicted demand, irrespective of the NN that produced it, as $\hat{y}_{s_k}$.

Deriving Minute Demand Distribution from Hourly Predicted Demand. Following the concept of Certainty Equivalence presented in Bertsekas and Castanon [10], we derive a semi-deterministic approximation for $\tilde{p}_{\eta_{s_k}}$ by evenly distributing the predicted hourly number of requests for sector s_k over the time horizon $N = 60$ to obtain the number of requests that will enter the system at each minute. In this sense, we obtain a single descriptor $\tilde{\eta}_{s_k}^{(\mathrm{det})}$ for all realizations $\tilde{\eta}_{s_k}(t), 1 \le t \le N$. More formally, We define $\tilde{\eta}_{s_k}^{(\mathrm{det})} = \frac{\hat{y}_{s_k}}{N}$. If $\tilde{\eta}_{s_k}^{(\mathrm{det})} > 1$, we round this value to obtain a deterministic quantity

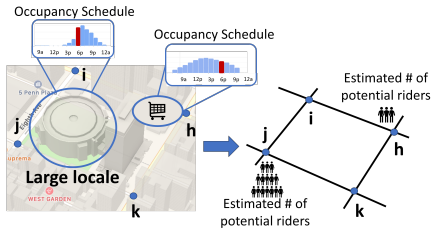

Figure 5: Mapping probability distributions over sectors to intersections for pickup and drop-off. The potential number of riders at each intersection is estimated using occupancy schedules and the maximum occupancy of the locale

$\bar{\eta}_{s_k}$ that will be used as the number of requests that enter sector s_k at each minute t. If $\tilde{\eta}^{(\text{det})}_{s_k} < 1$, then we define a Bernoulli random variable $\tilde{\eta}^{\text{bern}}_{s_k} \sim Bern(\tilde{\eta}^{(\text{det})}_{s_k})$, and its realization at time t as $\tilde{\eta}^{\text{bern}(t)}_{s_k}$, such that $\tilde{\eta}^{\text{bern}(t)}_{s_k} = 1$ with probability $\tilde{\eta}^{(\text{det})}_{s_k}$. We set $\bar{\eta}_{s_k} = \tilde{\eta}^{\text{bern}}_{s_k}$. For each minute t in time horizon $N = 60$, we instantiate $\tilde{\eta}^{\text{bern}(t)}_{s_k}$ to determine if a request appeared in sector s_k at minute t. In contrast to standard certainty equivalence [10], our approach does not fully remove the stochasticity for the random variable $\tilde{\eta}_{s_k}$ to avoid underestimating demand.

4.2.2 Estimating Probability Distributions for the Pickup and Drop-off Intersections, $\tilde{p}_{\rho_{s_k}}$ and $\tilde{p}_{\delta_{s_k}}$, and the Conditional Matching $\tilde{p}_{\beta_{s_k}}$. Here, we tackle the problem of going from sector level demand to intersection-level demand that can be used as input to the routing optimization in Eq.2. To achieve this, the probability distributions for pickups and drop-offs over intersections, $\tilde{p}_{\rho_{s_k}}$ and $\tilde{p}_{\delta_{s_k}}$, are estimated by leveraging the demand assignment module (see Fig. 2 module 3), which given a set of intersections $\mathcal{I}_{s_k}$ for sector s_k, returns a probability distribution over $\mathcal{I}_{s_k}$ based on the number of locales of interest (i.e. restaurants, bars, cafes, etc) near that intersection, their respective occupancy schedules [31, 47], and an estimate of their maximum occupancy. A graphical representation of this process is shown in Fig. 5. More formally, let's consider sector s_k and an intersection j such that $j \in \mathcal{I}_{s_k}$. We denote $n_\lambda(j)$, as the number of locales for a specific locale type $\lambda \in \Lambda$ near intersection j, where Λ is the set of all locale types. We denote o_λ and p_λ, as the estimated maximum occupancy and the occupancy schedule (percentage of maximum occupancy) for locale type λ, respectively. We denote the hour of interest for pickups as h_ρ and the hour for drop-offs as h_δ. If we are interested in pickups and drop-offs during the same hour, then we have $h_\rho = h_\delta$. For simplicity, we denote an arbitrary hour irrespective of whether it is for pickups or drop-offs as h_t. We denote the percentage occupancy of locale type λ as $p_\lambda(h_t)$. We denote the estimate of the number of potential customers at node j given an hour h_t as $O(j, h_t) = \sum_{\lambda\in\Lambda} n_\lambda(j) \cdot (p_\lambda(h_t) \cdot o_\lambda)$. Therefore, for an intersection $j \in \mathcal{I}_{s_k}$ we have $\tilde{p}_{\rho_{s_k}}(j) = \frac{O(j,h_\rho)}{\sum_{i\in\mathcal{I}_{s_k}} O(i,h_\rho)}$. Similarly, we have $\tilde{p}_{\delta_{s_k}}(j) = \frac{O(j,h_\delta)}{\sum_{i\in\mathcal{I}_{s_k}} O(i,h_\delta)}$.

To estimate $\tilde{p}_{\beta_{s_k}}$, we consider historical demand data. We denote the number of historical requests that had sector s_k as origin and sector s_l as destination as $Y(s_k, s_l)$. The probability of dropping off a request at sector s_l given that it was picked up at sector s_k is given by $\tilde{p}_{\beta_{s_k}}(s_l) = \frac{Y(s_k,s_l)}{\sum_{s_a\in S} Y(s_k,s_a)}$.

5 EXPERIMENTAL EVALUATION

To evaluate our approach, we consider ride-share trip data for the year 2022 obtained from NYC's HV-FHV datasets [18]. We consider a section of the map with 38 sectors (2235 intersections and 4566 streets) across mid and lower Manhattan (see Fig. 3). We collect data for more than 2, 000 different events during the year 2022 hosted on 300 unique venues across all 38 sectors. We consider that each time step in the system corresponds to 1 minute, and the time horizon $N = 60$ represents an hour.

5.1 Implementation Details

To implement the system, we need to deal with two main areas:

5.1.1 Review Data and Spectral Clustering: We obtain a list of events for a specific date using the PredictHQ API [50]. We consider seven event categories: conferences, expos, concerts, festivals, performing arts, community gatherings, and sports events. For each of these events, we scrape Google Maps using SerpAPI [52] to obtain reviews related to an event and its venue. We set $Q = 100$ to collect a maximum of 100 reviews. We choose RoBERTa large v1 [36] as the MLM for sentence embeddings, We choose the number of clusters for the review embeddings $b = 3$ based on previous work on sentiment analysis [25, 65] that suggests 3 clusters (positive, negative and neutral reviews). However, b can be set to any number, as long as it is large enough to capture the semantic diversity of the reviews, but still small enough to produce reasonable sized input features for the neural networks.

5.1.2 Demand Prediction, Demand Assignment and Model-based RL Routing Modules: All the weather data is obtained using Open-Meteo Historical Weather Data API[46], which uses ERA5[32, 43]. We train both NNs for 100 epochs using the first 8 months of data for the year 2022. The remaining 4 months were held out as a test set. For both NNs, we select mean squared error (MSE) as the loss function, we use Adam with a learning rate of 1×10^{-4} and a weight decay of 1×10^{-6} as an optimizer, we choose a batch size of 64, and we shuffle the batches after every epoch. The NN that predicts demand using $\vec{f}$ as input has 2 fully-connected hidden layers, each with 256 neurons. The NN that predicts demand using the features $F^+_{s_k}$ has 2 fully-connected hidden layers with 4096 neurons each. Architectural parameters for both NNs were selected using a hyper-parameter search, and a simple feed forward architecture was chosen as it was the fastest architecture to train, while still obtaining comparable results to other more complex architectures. All training was done on a single NVIDIA RTX A6000. All evaluation results are calculated using the predictions for the 4 months of data in the test set. For the demand assignment module, we retrieve all locales of interest for each intersection $j \in \mathcal{I}_V$ using Google Maps' Nearby Search API [49]. We consider four locale types: retail spaces, restaurants, hotels, and hospitals. We retrieve occupancy schedules from COMNET [19]. For the rollout-based routing module, we set the planning horizon $H = 10$ and use 1000 Monte-Carlo simulations to estimate the expectation for the one-agent-at-a-time online minimization, and choose $\bar{\pi}$ to be the fastest computing heuristic,

the greedy policy (see Sec. 5.2). It is important to note that our proposed method will still work for base policies that are easy to compute and have a reasonable behavior.

5.2 Baselines

Our main results compare our approach against the following four baselines: **1) Greedy policy:** available taxis are routed to their nearest outstanding request as given by Dijkstra's shortest path algorithm without coordination. **2) Instantaneous assignment:**uses a variation of broadcast of local eligibility (BLE) [59] for iterative task assignment, performing a deterministic instantaneous matching of available taxis to request currently in the system. **3) Garces et al. [30]:** in its scalable implementation, this methods uses a one-agent-a-time rollout with instantaneous assignment as the base policy. It estimates the current demand using the demand of the previous hour of operation of the system. This method uses a standard application of certainty equivalence, utilizing the mean values for $\tilde{\eta}_{s_h}, \tilde{\rho}_{s_h}, \tilde{\delta}_{s_h}$, but preserving the stochasticity for $\tilde{\beta}_{s_h}$ and the order in which requests arrive by independently sampling from a precomputed pool of pickups and drop-offs. **4) Oracle:** This method has full a-priori knowledge of the exact time, pickup, and drop-off locations for the requests entering the system in the future. For this reason, this method is able to route taxis to the exact location of each request even before the request has entered the system. This method minimizes the total wait time for all requests by executing a series of assignments that leverage the auction algorithm [5] with full future information. This method is not achievable in practice, but it provides a lower bound on the cost.

Note that we do not compare against policy gradient methods as applying these methods to a large scale multiagent routing environment with demand surges is still an open problem.

5.3 Main Results

In this section, we present a comparative study of our proposed approach and the baselines described in Sec. 5.2. We evaluate all policies using the held out test dataset described in Sec. 5.1. We use the real requests in the ride service data [18] as the ground truth requests entering the system, and we select a time window from 3pm to 9pm to include pre-surge scenarios (3pm), event-driven surge times (5pm and 7pm), and post-surge times (9pm). In the selected time window events happen 4 to 6 days a week usually in the selected time range (start time of concerts and theater shows). All results are obtained by averaging results for 25 random initial states for a randomly chosen date (in this case, November 17, 2022).

First, we compare the average total cost (total wait time for all requests at the end of the horizon) incurred by each policy. We present these results in Table 1. We report both the raw cost and the percent difference from the oracle (% diff). The percent difference from the oracle is calculated by subtracting the cost for the oracle from the cost of a given policy and then dividing the result by the cost of the oracle. As shown in Table 1, our proposed approach obtains the lowest average total cost compared to all the other feasible methods, having a 2% to 10% lower raw cost and 1% to 10% lower % diff than all the other baselines depending on the hour considered.

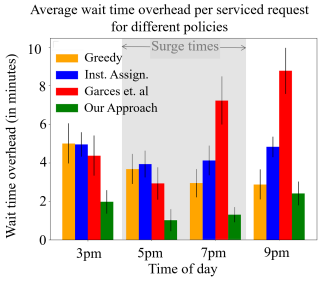

Figure 6: Average wait time overhead per serviced request (in minutes).Our proposed approach (green) results in a serviced request waiting 1 to 2 minutes, while for all the other baselines a serviced request has to wait 3 to 10 minutes. Surge times are shaded in gray.

We also present results for the average wait time overhead per serviced request in order to better understand the impact of each policy on rider experience. Intuitively wait time overhead is the additional amount of time that a request will have to wait in a realistic stochastic setting compared to a setting in which the locations and times of all requests are known a-priori. To calculate the wait time overhead per serviced request we take the total wait time for all requests for a given policy, subtract the total wait time for the oracle and then divide the resulting value by the number of serviced requests for the given policy. We present these results in Fig. 6. As shown in Fig. 6, our approach results in 25% to 75% improvement on wait time overhead per serviced request over all the other methods, which translates to faster pickups and a better customer experience overall. Our method is particularly useful in surge times between 5pm and 7pm, as it results in wait time overheads that are 3X smaller than wait time overheads for all the other methods.

Additionally, we present the number of outstanding requests at the end of the horizon to see which feasible policy services more requests. Since we are dealing with a fixed fleet-size that is chosen to satisfy average demand scenarios, demand surges usually lead to accumulation of unserviced requests at the end of the horizon. Table 2 contains the results for the number of outstanding requests at the end of the horizon for all methods. As shown in Table 2, our method has 1% to 4% fewer outstanding requests at the end of the horizon than all the other baselines.

From all these results, we obtain that our method does not only decrease wait time overhead for serviced requests by 25% to 75%, depending on the hour, but it also services 1% to 4% more requests than all the other methods.

Table 3: Average computation time for planning for a single time step (in seconds)

Policy	Runtime (in seconds)
Garces et al.	153.42 ± 7.65
Our approach	56.47 ± 5.40

To emphasize the computational time reductions associated with our approach compared to standard one-agent-at-a-time rollout, we present computation times for the method in Garces et al. [30] (see Sec. 5.2) and for our approach in Table 3. As shown in Table 3 our method is able to plan for the next time step in approximately a third of the time compared to Garces et al. [30], making it more suitable for city-scale applications.

Table 1: Average total cost and percent difference from the oracle for different policies. Percent difference from the oracle is denoted as % diff.

	Policies								
	Greedy		Inst. Assign.		Garces et al.		Our approach		Oracle
Time of day	raw cost	% diff.	raw cost	% diff.	raw cost	% diff.	raw cost	% diff.	raw cost
3pm	6671.2 ± 269.0	16.4	6670.6 ± 193.9	16.4	6563.9 ± 243.5	14.5	**6121.5 ± 217.5**	**6.8**	5732.1 ± 165.4
5pm	11795.7 ± 202.6	6.4	11848.0 ± 165.1	6.9	11667.9 ± 174.5	5.3	**11291.7 ± 169.2**	**1.9**	11082.0 ± 134.7
7pm	17199.8 ± 187.7	3.8	17432.2 ± 187.0	5.2	18070.6 ± 210.6	9.0	**16868.2 ± 141.2**	**1.8**	16573.4 ± 103.8
9pm	12102.4 ± 143.1	4.8	12460.4 ± 107.4	7.9	13147.4 ± 217.5	13.8	**12020.6 ± 154.6**	**4.1**	11552.6 ± 80.7

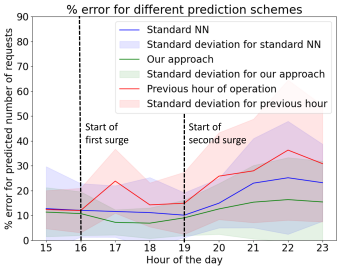

Figure 7: PE_{s_k} for different demand prediction schemes

Table 2: Average number of outstanding requests at the end of the hour for different policies

	Policies			
Time of day	Greedy	Inst. Assign.	Garces et al.	Our approach
3pm	241.92 ± 7.87	240.92 ± 3.75	238.42 ± 4.66	**230.17 ± 4.30**
5pm	434.00 ± 4.07	433.33 ± 3.52	426.83 ± 4.76	**422.58 ± 4.25**
7pm	626.23 ± 5.15	630.15 ± 5.49	632.38 ± 4.89	**611.46 ± 3.71**
9pm	471.14 ± 5.71	474.93 ± 3.63	479.79 ± 3.53	**467.14 ± 5.04**

5.4 Ablation Studies

In this section we consider modifications of our approach, mainly dealing with two aspects: the demand prediction system and the base policy used for the rollout-based routing algorithm.

5.4.1 Demand Prediction System. To better understand the effect of including event information in the demand prediction, we isolate the demand prediction system and compare it against a NN that only uses temporal and spatial data as input (we call this standard NN). We also compare our demand prediction against a probabilistic estimation of demand that uses the previous hour of operation of the system as a proxy for the current demand as proposed in [30]. Results for the prediction errors of all three methods are shown in Fig. 7.

Table 4: Average total cost of our method for two base policies

	Base Policies	
Time of day	Inst. Assign.	Greedy
3pm	103.73 ± 3.32	102.57 ± 3.34
5pm	189.25 ± 3.23	189.03 ± 2.12
7pm	279.37 ± 2.47	280.88 ± 2.79
9pm	202.84 ± 2.36	201.79 ± 2.06

The figure presents prediction errors for each hour in the chosen time window averaged over all days in November 2022, where we compute average percent error $PE_{s_k} = \frac{1}{|X|}\sum_X \frac{|y_{s_k} - \hat{y}_{s_k}|}{y_{s_k}}$, where X is the number of data points in the test set for sector s_k at the hour of interest, $\hat{y}_{s_k}$ is the predicted number of requests entering sector s_k, and y_{s_k} is the actual number of requests entering s_k. Our prediction scheme outperforms all other methods, obtaining 3% – 10% improvement on average percent error.

5.4.2 Base Policy. Our proposed approach does not depend on the choice of the base policy as long as the base policy is reasonable and easy to compute. To illustrate this point, we consider an ablation study where we compare the performance of our approach when we change the base policy. We consider instantaneous assignment and a greedy policy as defined in Sec. 5.2 as potential candidates for base policies. The results of this comparison are shown in Table 4, where we can see that the total cost of our method at the end of the time horizon is similar for both base policies.

6 LIMITATIONS AND FUTURE WORK

Since we consider a fixed fleet size, larger demand surges still lead to a larger number of outstanding requests compared to hours where there are less events. As future work, we want to consider a system where there are taxis stored at warehouses throughout the map, and we can dynamically change the fleet size to address this rise in demand. Another important limitation of our approach is that it relies on sampling to estimate the expected cost of each action, and hence there is still a small, but non-zero, probability that the samples obtained are not representative of the real demand, leading to a degrade in performance. As shown in the simulations, this probability is very small and performance degradation happens very infrequently.

7 CONCLUSION

In this paper we presented an event-informed multi-agent RL routing framework that leverages event data from the internet to predict demand surges and route taxis accordingly. The framework proposed in this paper is applicable to multiple multiagent sequential decision making problems where unstructured textual data can be used to predict changes in the environment. One of such problems is the routing application covered in the paper, in which event data (textual data) is leveraged to predict changes in the demand (the environment). Other examples of potential applications are: multirobot repair/maintenance problems where there are textual descriptions of the conditions of facilities, robotic search-and-rescue tasks where humans collaborate with robots, among others. In the routing application, our method decreases wait time overhead for serviced requests by 25% to 75%, while servicing 1% to 4% more requests than the baselines.

ACKNOWLEDGMENTS

The authors gratefully acknowledge ONR award #N00014-21-1-2714, AFOSR award #FA9550-22-1-0223 and NSF CAREER award #2114733 for partial support of this work.

REFERENCES

[1] Tanvir Ahamed, Bo Zou, Nahid Parvez Farazi, and Theja Tulabandhula. 2021. Deep Reinforcement Learning for Crowdsourced Urban Delivery. *Transportation Research Part B: Methodological* 152 (2021), 227–257. https://doi.org/10.1016/j.trb.2021.08.015

[2] ASHRAE. 2013. ANSI/ASHRAE/IES Standard 90.1-2013: Energy Standard for Buildings Except Low-Rise Residential Buildings.

[3] Oren Barkan, Noam Razin, Itzik Malkiel, Ori Katz, Avi Caciularu, and Noam Koenigstein. 2019. Scalable Attentive Sentence-Pair Modeling via Distilled Sentence Embedding. arXiv:1908.05161 [cs.LG]

[4] Russell Bent and Pascal Van Hentenryck. 2004. The Value of Consensus in Online Stochastic Scheduling.. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling, ICAPS 2004*. American Association for Artificial Intelligence, Providence, RI, 219–226.

[5] D.P. Bertsekas. 1979. A Distributed Algorithm for the Assignment Problem. *Lab for Information and Decision Systems Report* (05 1979).

[6] Dimitri Bertsekas. 2020. Multiagent value iteration algorithms in dynamic programming and reinforcement learning. *Results in Control and Optimization* 1 (2020), 100003. https://doi.org/10.1016/j.rico.2020.100003

[7] D. Bertsekas. 2020. *Rollout, Policy Iteration, and Distributed Reinforcement Learning*. Athena Scientific. https://books.google.com/books?id=Hbo-EAAAQBAJ

[8] Dimitri Bertsekas. 2021. Multiagent Reinforcement Learning: Rollout and Policy Iteration. *IEEE/CAA Journal of Automatica Sinica* 8, 2 (2021), 249–272. https://doi.org/10.1109/JAS.2021.1003814

[9] Dimitri Bertsekas. 2022. *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control*. Athena Scientific, Nashua, NH, USA.

[10] D.P. Bertsekas and D.A. Castanon. 1998. Rollout algorithms for stochastic scheduling problems. In *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*, Vol. 2. 2143–2148 vol.2. https://doi.org/10.1109/CDC.1998.758655

[11] Dimitris Bertsimas, Patrick Jaillet, and Sébastien Martin. 2019. Online Vehicle Routing: The Edge of Optimization in Large-Scale Applications. *Oper. Res.* 67 (2019), 143–162.

[12] Sushmita Bhattacharya, Siva Kailas, Sahil Badyal, Stephanie Gil, and Dimitri Bertsekas. 2021. Multiagent Rollout and Policy Iteration for POMDP with Application to Multi-Robot Repair Problems. In *Proceedings of the 2020 Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 155)*, Jens Kober, Fabio Ramos, and Claire Tomlin (Eds.). PMLR, 1814–1828. https://proceedings.mlr.press/v155/bhattacharya21a.html

[13] Dun Cao, Kai Zeng, Jin Wang, Pradip Kumar Sharma, Xiaomin Ma, Yonghe Liu, and Siyuan Zhou. 2022. BERT-Based Deep Spatial-Temporal Network for Taxi Demand Prediction. *IEEE Transactions on Intelligent Transportation Systems* 23, 7 (2022), 9442–9454. https://doi.org/10.1109/TITS.2021.3122114

[14] Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder. arXiv:1803.11175 [cs.CL]

[15] Long Chen, Piyushimita Thakuriah, and Konstantinos Ampountolas. 2021. Short-Term Prediction of Demand for Ride-Hailing Services: A Deep Learning Approach. *Journal of Big Data Analytics in Transportation* 3 (08 2021). https://doi.org/10.1007/s42421-021-00041-4

[16] Shukai Chen, Hua Wang, and Qiang Meng. 2020. Solving the first-mile ridesharing problem using autonomous vehicles. *Computer-Aided Civil and Infrastructure Engineering* 35, 1 (2020), 45–60. https://doi.org/10.1111/mice.12461 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.12461

[17] Samuel Chng, Sabreena Anowar, and Lynette Cheah. 2022. Understanding Shared Autonomous Vehicle Preferences: A Comparison between Shuttles, Buses, Ridesharing and Taxis. *Sustainability* 14, 20 (Oct 2022), 13656. https://doi.org/10.3390/su142013656

[18] NYC Taxi & Limousine Commission. 2020-2022. TLC Trip Record Data. https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[19] COMNET. 2016. Factsheet COMNET Overview - Appendix C: Schedules. https://www.comnet.org/appendix-c-schedules

[20] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2018. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. arXiv:1705.02364 [cs.CL]

[21] G. A. Croes. 1958. A Method for Solving Traveling-Salesman Problems. *Operations Research* 6, 6 (1958), 791–812. http://www.jstor.org/stable/167074

[22] Anil Damle, Victor Minden, and Lexing Ying. 2018. Simple, direct and efficient multi-way spectral clustering. *Information and Inference: A Journal of the IMA* 8, 1 (06 2018), 181–203. https://doi.org/10.1093/imaiai/iay008 arXiv:https://academic.oup.com/imaiai/article-pdf/8/1/181/28053156/iay008.pdf

[23] Arthur Delarue, Ross Anderson, and Christian Tjandraatmadja. 2020. Reinforcement Learning with Combinatorial Actions: An Application to Vehicle Routing. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 609–620.

[24] Ran Duan and Seth Pettie. 2014. Linear-Time Approximation for Maximum Weight Matching. *J. ACM* 61, 1, Article 1 (2014), 23 pages. https://doi.org/10.1145/2529989

[25] Ikram el Karfi, Sanaa El Fkihi, and Rdouan Faizi. 2018. A SPECTRAL CLUSTERING-BASED APPROACH FOR SENTIMENT CLASSIFICATION IN MODERN STANDARD ARABIC. In *Proceedings of the International Conferences Big Data Analytics, Data Mining and Computational Intelligence 2018*.

[26] Tobias Enders, James Harrison, Marco Pavone, and Maximilian Schiffer. 2023. Hybrid multi-agent deep reinforcement learning for autonomous mobility on demand systems. In *Learning for Dynamics and Control Conference*. PMLR, 1284–1296.

[27] Günes Erkan and Dragomir R Radev. 2004. The university of michigan at duc 2004. In *Proceedings of the Document Understanding Conferences Boston, MA*. Citeseer.

[28] J. Farhan and T. Donna Chen. 2018. Impact of ridesharing on operational efficiency of shared autonomous electric vehicle fleet. *Transportation Research Part C: Emerging Technologies* 93 (08 2018), 310–321. https://doi.org/10.1016/j.trc.2018.04.022

[29] Daniele Gammelli, Kaidi Yang, James Harrison, Filipe Rodrigues, Francisco Pereira, and Marco Pavone. 2022. Graph meta-reinforcement learning for transferable autonomous mobility-on-demand. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2913–2923.

[30] Daniel Garces, Sushmita Bhattacharya, Stephanie Gil, and Dimitri Bertsekas. 2023. Multiagent Reinforcement Learning for Autonomous Routing and Pickup Problem with Adaptation to Variable Demand. *International Conference on Robotics and Automation* (2023).

[31] Gabriel Happle, Jimeno Fonseca, and Arno Schlueter. 2019. Data-driven occupancy schedules for commercial buildings. https://doi.org/10.3929/ethz-b-000341619

[32] H. Hersbach, P. Bell, B.and Berrisford, G. Biavati, A. Horányi, J. Muñoz Sabater, J. Nicolas, C. Peubey, R. Radu, D. Rozum, I.and Schepers, A. Simmons, C. Soci, D. Dee, and J-N. Thépaut. 2018. ERA5 hourly data on single levels from 1959 to present. https://doi.org/10.24381/cds.adbb2d47

[33] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. 1990. An Optimal Algorithm for On-Line Bipartite Matching. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing* (Baltimore, Maryland, USA) *(STOC '90)*. Association for Computing Machinery, New York, NY, USA, 352–358. https://doi.org/10.1145/100216.100262

[34] Xiaolong Li, Gang Pan, Z. Wu, Guande Qi, Shijian Li, Daqing Zhang, Wangsheng Zhang, and Zonghui Wang. 2012. Prediction of urban human mobility using large-scale taxi traces and its applications. *Frontiers of Computer Science in China* 6 (02 2012), 111–121. https://doi.org/10.1007/s11704-011-1192-6

[35] Lingbo Liu, Zhilin Qiu, Guanbin Li, Qing Wang, Wanli Ouyang, and Liang Lin. 2019. Contextualized Spatial–Temporal Network for Taxi Origin-Destination Demand Prediction. *IEEE Transactions on Intelligent Transportation Systems* 20, 10 (2019), 3875–3887. https://doi.org/10.1109/TITS.2019.2915525

[36] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv* abs/1907.11692 (2019).

[37] Meghna Lowalekar, Pradeep Varakantham, and Patrick Jaillet. 2018. Online spatio-temporal matching in stochastic and dynamic domains. *Artificial Intelligence* 261 (2018), 71–112. https://doi.org/10.1016/j.artint.2018.04.005

[38] Ioulia Markou, Kevin Kaiser, and Francisco C. Pereira. 2019. Predicting taxi demand hotspots using automated Internet Search Queries. *Transportation Research Part C: Emerging Technologies* 102 (2019), 73–86. https://doi.org/10.1016/j.trc.2019.03.001

[39] Ioulia Markou, Filipe Rodrigues, and Francisco C. Pereira. 2020. Is Travel Demand Actually Deep? An Application in Event Areas Using Semantic Information. *IEEE Transactions on Intelligent Transportation Systems* 21, 2 (2020), 641–652. https://doi.org/10.1109/TITS.2019.2897341

[40] Fei Miao, Shuo Han, Shan Lin, Qian Wang, John Stankovic, Abdeltawab Hendawi, Desheng Zhang, Tian He, and George J. Pappas. 2017. Data-Driven Robust Taxi Dispatch under Demand Uncertainties. arXiv:1603.06263 [cs.SY]

[41] Melanie Mitchell. 1998. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA.

[42] Luis Moreira-Matias, João Gama, Michel Ferreira, João Mendes-Moreira, and Luis Damas. 2013. Predicting Taxi–Passenger Demand Using Streaming Data. *IEEE Transactions on Intelligent Transportation Systems* 14, 3 (2013), 1393–1402. https://doi.org/10.1109/TITS.2013.2262376

[43] J. Muñoz Sabater. 2019. ERA5-Land hourly data from 1981 to present. https://doi.org/10.24381/cds.e2161bac

[44] MohammadReza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takac. 2018. Reinforcement Learning for Solving the Vehicle Routing Problem. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/9fb4651c05b2ed70fba5afe0b039a550-Paper.pdf

[45] Alireza Nejadettehad, Hamid Mahini, and Behnam Bahrak. 2019. Short-term Demand Forecasting for Online Car-hailing Services Using Recurrent Neural

Networks. *Applied Artificial Intelligence* 34 (2019), 674 – 689.
[46] Open-Meteo. 2022. Historical Weather API. https://open-meteo.com/en/docs/historical-weather-api
[47] Karthik Panchabikesan, Fariborz Haghighat, and Mohamed El Mankibi. 2021. Data driven occupancy information for energy simulation and energy use assessment in residential buildings. *Energy* 218 (2021), 119539. https://doi.org/10.1016/j.energy.2020.119539
[48] Nahid Parvez Farazi, Bo Zou, Tanvir Ahamed, and Limon Barua. 2021. Deep reinforcement learning in transportation research: A review. *Transportation Research Interdisciplinary Perspectives* 11 (2021), 100425. https://doi.org/10.1016/j.trip.2021.100425
[49] Google Maps Platform. 2022. Nearby Search API. https://developers.google.com/maps/documentation/places/web-service/search-nearby
[50] PredictHQ. 2022. Attended events API. https://www.predicthq.com/apis/event-api
[51] Filipe Rodrigues, Ioulia Markou, and Francisco C. Pereira. 2019. Combining time-series and textual data for taxi demand prediction in event areas: A deep learning approach. *Information Fusion* 49 (2019), 120–129. https://doi.org/10.1016/j.inffus.2018.07.007
[52] SerAPI. 2022. Google Maps API. https://serpapi.com/google-maps-api
[53] Hang Shao, Abhishek Kumar, and P. Thomas Fletcher. 2017. The Riemannian Geometry of Deep Generative Models. arXiv:1711.08014 [cs.LG]
[54] David Silver and Joel Veness. 2010. Monte-Carlo Planning in Large POMDPs. In *Proc. 23rd International Conf. on NeurIPS* (Vancouver, British Columbia, Canada). Red Hook, NY, USA, 2164–2172.
[55] K. Spieser, Kyle Treleaven, Rick Zhang, Emilio Frazzoli, Daniel Morton, and Marco Pavone. 2014. Toward a Systematic Approach to the Design and Evaluation of Automated Mobility-on-Demand Systems: A Case Study in Singapore. *Road Vehicle Automation. Lecture Notes on Mobility* (04 2014), 229–245.
[56] Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning General Purpose Distributed Sentence Representations via Large Scale Multi-task Learning. arXiv:1804.00079 [cs.CL]
[57] Kyle Treleaven, Marco Pavone, and Emilio Frazzoli. 2013. Asymptotically Optimal Algorithms for One-to-One Pickup and Delivery Problems With Applications to Transportation Systems. *IEEE Trans. Automat. Control* 58, 9 (2013), 2261–2276. https://doi.org/10.1109/TAC.2013.2259993
[58] Marlin W. Ulmer, Justin C. Goodson, Dirk C. Mattfeld, and Marco Hennig. 2019. Offline–Online Approximate Dynamic Programming for Dynamic Vehicle Routing with Stochastic Requests. *Transportation Science* 53, 1 (2019), 185–202. https://doi.org/10.1287/trsc.2017.0767 arXiv:https://doi.org/10.1287/trsc.2017.0767
[59] Barry Brian Werger and Maja J Matarić. 2000. Broadcast of local eligibility for multi-target observation. In *Distributed Autonomous Robotic Systems 4*. Springer, 347–356.
[60] Ziyi Yang, Yinfei Yang, Daniel Cer, Jax Law, and Eric Darve. 2021. Universal Sentence Representation Learning with Conditional Masked Language Model. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 6216–6228. https://doi.org/10.18653/v1/2021.emnlp-main.502
[61] Mihalis Yannakakis, Craig A. Tovey, Jan H. M. Korst, and Jan H. M. J. M. van Laarhoven. 2003. *Local Search in Combinatorial Optimization*. Princeton University Press. http://www.jstor.org/stable/j.ctv346t9c
[62] Rick Zhang and Marco Pavone. 2016. Control of Robotic Mobility-on-Demand Systems: A Queueing-Theoretical Perspective. *The International Journal of Robotics Research* 35, 1–3 (jan 2016), 186–203. https://doi.org/10.1177/0278364915581863
[63] Linhong Zhu, Aram Galstyan, James Cheng, and Kristina Lerman. 2014. Tripartite graph clustering for dynamic sentiment analysis on social media. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1531–1542.
[64] Linhong Zhu, Sheng Gao, Sinno Jialin Pan, Haizhou Li, Dingxiong Deng, and Cyrus Shahabi. 2013. Graph-based informative-sentence selection for opinion summarization. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 408–412.
[65] David Zimbra, Ahmed Abbasi, Daniel Zeng, and Hsinchun Chen. 2018. The State-of-the-Art in Twitter Sentiment Analysis: A Review and Benchmark Evaluation. *ACM Trans. Manage. Inf. Syst.* 9, 2, Article 5 (aug 2018), 29 pages. https://doi.org/10.1145/3185045