Multi-Objective Order Dispatch for Urban Crowd Sensing with For-Hire Vehicles

Jiahui Sun[†], Haiming Jin[†,*], Rong Ding[†], Guiyun Fan[†], Yifei Wei[‡], Lu Su[§]
[†]Shanghai Jiao Tong University, [‡]Carnegie Mellon University, [§]Purdue University
Email: {jhsun1997, jinhaiming, dingrong, fgy726}@sjtu.edu.cn, yifeiwei@andrew.cmu.edu, lusu@purdue.edu

IEEE INFOCOM 2023 - IEEE Conference on Computer Communications | 979-8-3503-3414-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/INFOCOM53939.2023.10229103

***Abstract*—For-hire vehicle-enabled crowd sensing (FVCS) has become a promising paradigm to conduct urban sensing tasks in recent years. FVCS platforms aim to jointly optimize both the order-serving revenue as well as sensing coverage and quality. However, such two objectives are often conflicting and need to be balanced according to the platforms' preferences on both objectives. To address this problem, we propose a novel cooperative multi-objective multi-agent reinforcement learning framework, referred to as MOVDN, to serve as the first preference-configurable order dispatch mechanism for FVCS platforms. Specifically, MOVDN adopts a decomposed network structure, which enables agents to make distributed order selection decisions, and meanwhile aligns each agent's local decision with the global objectives of the FVCS platform. Then, we propose a novel algorithm to train a single universal MOVDN that is optimized over the space of all preferences. This allows our trained model to produce the optimal policy for any preference. Furthermore, we provide the theoretical convergence guarantee and sample efficiency analysis of our algorithm. Extensive experiments on three real-world ride-hailing order datasets demonstrate that MOVDN outperforms strong baselines and can support the platform in decision-making effectively.**

***Index Terms*—urban crowd sensing, order dispatch, multi-objective multi-agent reinforcement learning**

I. Introduction

Urban crowd sensing, which is a type of spatial crowdsourcing paradigm that leverages humans or vehicles distributed in the city to collect data on urban metrics (e.g., traffic condition, infrastructure strain, air quality), has become increasingly popular. Compared to humans and private vehicles, *for-hire vehicles (FHVs)*, such as taxis and those operated by ride-hailing platforms, are usually the most desirable forces for urban crowd sensing. On one hand, since sensory data often contains locations and many other personal information (e.g., identity, arrival time), humans and private vehicles are normally unwilling to upload their data because of the privacy leakage risk. On the other hand, the installation of specialized sensing hardwares on private vehicles is not as convenient as FHVs that are managed by a centralized ride-hailing platform.

In *FHV-enabled crowd sensing (FVCS)* systems, order-serving revenue as well as sensing coverage and quality are two critical objectives. To jointly optimize both objectives, the order dispatch mechanism that decides which FHVs serve which orders plays an important role, since it not only directly affects the order-serving revenue, but also has a critical impact on the sensing outcome as it changes the distribution of FHVs across the city. However, such two objectives are often inconsistent in FVCS systems [1, 2]. In practice, FHVs tend to concentrate in busy areas, such as tourist attractions, business centers, and railway stations, where they often encounter considerably more passenger requests than neighborhoods that are socio-economically disadvantaged. However, many urban sensing tasks inherently require FHVs to distribute uniformly both spatially and temporally, so that sufficient sensory data can be collected continuously from everywhere in the city.

Clearly, different platforms may have different preferences on the order-serving and sensing outcomes, and even for the same platform, its preferences over different objectives could also evolve over time. As a result, the order-serving and sensing objectives have to be balanced according to the platforms' preferences. Therefore, in this paper, we aim to design a *preference-configurable order dispatch mechanism* for FVCS systems, which allows platforms to flexibly determine their preferences on the order-serving and sensing objectives.

Designing such a mechanism is naturally a sequential decision-making problem that aims to maximize the platform's cumulative rewards. A centralized decision-making framework (e.g., single-agent reinforcement learning) faces the issue of the exponential explosion in the action and state spaces caused by the large scale of a real-world FHV fleet, and is thus infeasible in FVCS systems. To resolve this challenge, we utilize the *multi-agent reinforcement learning (MARL)* framework, which views each FHV as an agent and trains a *distributed* policy for each agent that generates its own order selection decisions.

However, designing such a multi-agent distributed decision-making framework further raises the critical task of *aligning each agent's local decision with the global objectives* of the whole FVCS system. To tackle this problem, we factorize the joint state-action value function among agents by designing a *value decomposition network (VDN)*. Specifically, our VDN consists of one network for each agent that only depends on its local observations, and a mixing network that combines all agent networks' outputs as well as the global state to model the joint state-action value function. Such a mixing network is no longer needed in the execution stage as each agent can take actions according to its own network. Our carefully designed mixing network ensures that the optimal joint action performed using the joint state-action value function is consistent with the collection of each agent's optimal action that is individually performed using its own network.

Another challenge comes from *the conflicting order-serving and sensing objectives* of the FVCS system, which makes it impossible to simultaneously maximize both objectives. To this end, we adopt *Pareto-optimal* as the solution concept, and different preferences correspond to different Pareto-optimal

*Corresponding author.

policies. However, as there are typically infinitely many preferences, it is intractable to train a VDN for each possible preference. Therefore, we meticulously design a *multi-objective value decomposition network (MOVDN)* that takes as input the platform's preference, and propose a novel sample-efficient algorithm to train a single universal MOVDN that is optimized over the space of all preferences. This allows our trained model to produce the Pareto-optimal policy for any preference.

In summary, this paper makes the following contributions.

- To the best of our knowledge, this is the first work that designs a preference-configurable order dispatch mechanism, which allows FVCS platforms to flexibly determine their preferences on order-serving and sensing objectives.
- Technically, we (i) design a multi-objective MARL framework, which helps FHVs make distributed order selection decisions that cooperatively optimize the system-wide goals, (ii) propose a novel algorithm to train a single universal network that generates actions based on the input preference, and (iii) further provide the theoretical convergence guarantee and sample efficiency analysis of our algorithm.
- We conduct extensive experiments with three public large-scale real-world ride-hailing order datasets, including over 2 million orders in Haikou, China, over 5 million orders in Chengdu, China, and over 6 million orders in New York City, USA. The experimental results show that our MOVDN outperforms strong baselines and can support the platform in decision-making effectively.

II. Preliminaries

A. System Overview

Our FVCS system consists of a cloud-based platform and a set $\mathcal{N} = \{1, 2, \cdots, N\}$ of FHVs managed by the platform. In addition to serving the ride-hailing orders, FHVs also carry out urban sensing tasks via either specialized sensors or simply those on-board drivers' smartphones. The status of each FHV is either *on-service*, if it is serving orders at present, or *idle*, otherwise. As an industrial common practice [3], the time horizon is discretized into T time slots, denoted as $\mathcal{T} = \{1, 2, \cdots, T\}$, and the city area is divided into G grids, denoted as $\mathcal{G} = \{1, 2, \cdots, G\}$. Figure 1 demonstrates the workflow of our FVCS system, the details of which in each time slot are explained as follows.

- At the beginning of a time slot, the platform sends each idle FHV the information (e.g., the number of orders and vehicles) of its current and neighboring grids.
- Based on such information, each idle FHV uses its local decision module to select an order from nearby[1] candidate orders. Afterwards, the idle FHVs who pick up passengers become on service and drive to the passengers' destinations.
- Both the idle and on-service FHVs continuously collect sensory data while traveling in the city, and upload the collected data to the platform at the end of each time slot.

[1]Different platforms may have different rules of defining the maximum allowable serving distance. Our model is compatible with any such rule.

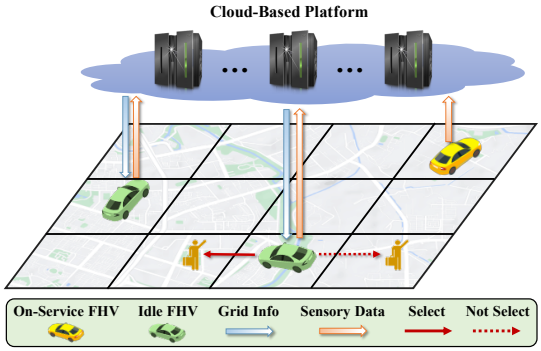


Fig. 1: A demonstration of interactions between FHVs and the cloud-based platform.

We in this paper consider to design and train the local decision module for FHVs, which helps them make appropriate order selection decisions.

B. Problem Description

In this paper, we aim to jointly optimize both the sensing and order-serving outcomes. As is the industrial common practice, we use the *gross merchandise volume* defined in Definition 1 to evaluate the platform's order-serving performance.

Definition 1 (GMV). *Let $\mathcal{O}_t$ be the set of orders whose service start from time slot t. The order-serving revenue of the platform in time slot t is $r_t^o = \sum_{j \in \mathcal{O}_t} p_j$, where p_j denotes the price of order $j \in \mathcal{O}_t$. The gross merchandise volume (GMV) is defined as $\sum_{t \in \mathcal{T}} r_t^o = \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{O}_t} p_j$, which is the sum of r_t^o along the whole time horizon.*

Next, we define *system sensing utility* in Definition 2 to measure the platform's sensing performance.

Definition 2 (SSU). *Let $N_{g,t}$ be the number of vehicles that traverse grid g during time slot t, and $f(\cdot)$ be the function that maps $N_{g,t}$ to the local sensing utility of grid g. Then, the platform's global sensing utility in time slot t is defined as $r_t^s = \sum_{g \in \mathcal{G}} \alpha_g f(N_{g,t})$, where α_g is the weight of grid g. The overall system sensing utility (SSU) is defined as $\sum_{t \in \mathcal{T}} r_t^s = \sum_{t \in \mathcal{T}} \sum_{g \in \mathcal{G}} \alpha_g f(N_{g,t})$, which is the sum of r_t^s along the whole time horizon.*

In practice, the function $f(\cdot)$ in Definition 2 is usually monotonically increasing with decreasing marginal returns. The reason lies in the fact that more sensory data will be collected when more FHVs pass a grid, but when the sensory data are already adequate in a grid, further increase of FHVs in that grid will bring minor gain to the sensing utility. Although the specific expression of the function $f(\cdot)$ differs in different applications, our model is compatible with any $f(\cdot)$. Furthermore, the weight α_g associated with each grid g helps the platform distinguish the importance degrees of different grids according to its own needs.

As which FHVs serve which orders highly affects both the sensing and order-serving outcomes, we thus aim to optimize FHVs' order selection decisions in order to jointly maximize the GMV and SSU of our FVCS system.

III. Formulation

In a real-world FVCS system, order selection decisions are made in every time slot with the objective of maximizing the platform's cumulative rewards. A typical way to resolve such sequential decision-making problems is to consider the platform as an agent, who makes centralized order selection decisions for all FHVs. However, such a method faces the exponential explosion issue in the action and state spaces due to the large scale of FHVs that typically exist in a real-world FVCS system, and is thus unscalable. Therefore, we adopt a decentralized multi-agent decision-making framework that implements an agent at each FHV, who works cooperatively with the other agents to jointly optimize GMV and SSU. Specifically, we formulate this multi-objective multi-agent decision-making problem as a *cooperative multi-objective Markov game* (referred to as FVCS-CMOMG), which contains the following elements.

- **Agent**: An agent is a local decision module that the platform implements at an FHV to make order selection decisions. We use $\mathcal{N}$ to denote the agent set.
- **State**: At the beginning of each time slot t, state $\mathbf{s} = \left[t, [\mathbf{f}_{g,t}]_{g\in\mathcal{G}}\right]$ of FVCS-CMOMG consists of the current time slot index t and each grid g's feature vector $\mathbf{f}_{g,t}$ that contains the grid index g and the current number of idle FHVs, on-service FHVs, and orders waiting to be served in grid g.
- **Observation**: At the beginning of each time slot t, each agent i receives the feature vector $\mathbf{f}_{g,t}$ of the grid g where it currently locates as its local observation $\mathbf{o}_i$.
- **Action**: At the beginning of each time slot t, the set of idle FHVs' agents, denoted as $\mathcal{N}_t$, take actions, and the action a_i of each agent $i \in \mathcal{N}_t$ indicates whether it remains idle, and which order it chooses to serve, if it decides to terminate idleness. We denote the action space of each agent i as $\mathcal{A}_i^t$, and the agents' joint action as $\mathbf{a} = [a_i]_{i\in\mathcal{N}_t}$ at time slot t.
- **Policy**: Each agent i's policy π_i specifies the probability $\pi_i(a_i|\mathbf{o}_i)$ that agent i takes each action a_i given observation $\mathbf{o}_i$. The joint policy of the agents are denoted as $\boldsymbol{\pi} = \left[\pi_1, \cdots, \pi_N\right]$.
- **Transition probability function**: Given the state $\mathbf{s}$ and joint action $\mathbf{a}$, the current state $\mathbf{s}$ transits to the next state $\mathbf{s}'$ with the probability $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$.
- **Reward**: At the end of each time slot t, the platform receives a vector immediate reward $\mathbf{r}(\mathbf{s}, \mathbf{a}) = [r_t^o, r_t^s]$, which consists of the order-serving revenue r_t^o and the global sensing utility r_t^s of the platform in time slot t. Note that both r_t^o and r_t^s are team rewards that are non-decomposable among agents, since they are the results of the cooperation of agents. Furthermore, a discount factor $\gamma \in [0, 1)$ is introduced to determine the importance of future rewards.

In our FVCS-CMOMG, the platform aims to find the joint policy $\boldsymbol{\pi}$ that maximizes the expected discounted cumulative vector reward $\mathbf{J}^{\boldsymbol{\pi}} = [J_o^{\boldsymbol{\pi}}, J_s^{\boldsymbol{\pi}}]$, where $J_o^{\boldsymbol{\pi}} = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t\in\mathcal{T}} \gamma^t r_t^o\right]$, and $J_s^{\boldsymbol{\pi}} = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t\in\mathcal{T}} \gamma^t r_t^s\right]$.

However, $J_o^{\boldsymbol{\pi}}$ and $J_s^{\boldsymbol{\pi}}$ are often conflicting, i.e., improving one of them is at the cost of reducing the other. As a result, the optimal joint policy that achieves the maximum on both objectives no longer exists in our FVCS-CMOMG. Thus, we adopt *Pareto-optimal* as the solution concept of our FVCS-CMOMG, which is defined in Definition 3.

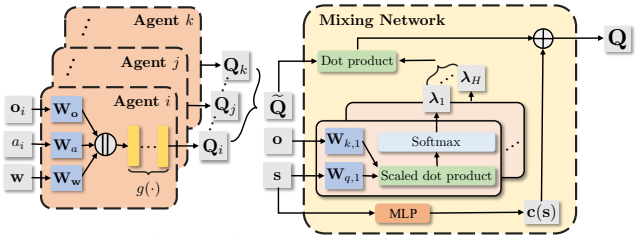


Fig. 2: The MOVDN framework.

Definition 3 (Pareto-Optimal)**.** *In our FVCS-CMOMG, policy $\boldsymbol{\pi}$ dominates policy $\boldsymbol{\pi}'$, if $J_o^{\boldsymbol{\pi}} \geq J_o^{\boldsymbol{\pi}'}, J_s^{\boldsymbol{\pi}} \geq J_s^{\boldsymbol{\pi}'}$, and at least one of the inequalities is strict. A policy $\boldsymbol{\pi}$ is Pareto-optimal, if and only if it is not dominated by any other policies.*

Solving FVCS-CMOMG is thus equivalent to finding the set of all Pareto-optimal policies. However, such a set is prohibitively expensive to obtain [4], especially when policies are represented by functions with large-scale parameters, e.g., neural networks leveraged in this paper. Instead, we consider to obtain the *convex coverage set* defined in Definition 4.

Definition 4 (CCS)**.** *In our FVCS-CMOMG, the convex coverage set (CCS) is defined as $\{\boldsymbol{\pi}_{\mathbf{w}} = \arg\max_{\boldsymbol{\pi}} \mathbf{w}\mathbf{J}^{\boldsymbol{\pi}} | \mathbf{w} \in \Omega\}$ with $\Omega = \{[w_1, w_2] | w_1 \geq 0, w_2 \geq 0, w_1 + w_2 = 1\}$ representing the space of the weight vector $\mathbf{w}$.*

By Definition 4, CCS contains the policy $\boldsymbol{\pi}_{\mathbf{w}}$ that maximizes the linearly scalarized objective for each weight vector $\mathbf{w} \in \Omega$. In fact, it is sufficient to compute CCS for episodic tasks as our FVCS-CMOMG that treats a day as one episode, since we can construct such mixture policies by selecting some $\boldsymbol{\pi}_{\mathbf{w}}$ under a probability distribution at the start of each episode, which can dominate the policies that are not in CCS [4]. The weight vector $\mathbf{w}$ formally represents the platform's preference, which specifies the relative importance of each objective.

In this paper, we aim to obtain the CCS of our FVCS-CMOMG. As we consider the practical scenario where the transition probability function is unknown *a priori*, we take the approach of learning the policies in the CCS via a novel framework of MARL, which will be elaborated in the following Section IV.

IV. Solution Method

A. Method Overview

Intuitively, one could learn the policy $\boldsymbol{\pi}_{\mathbf{w}}$ by solving the problem $\max_{\boldsymbol{\pi}} \mathbf{w}\mathbf{J}^{\boldsymbol{\pi}}$ for each $\mathbf{w}$ separately. However, such a method that trains different policies for the infinitely many $\mathbf{w}$'s in Ω is obviously intractable. Instead, we represent the vector state-action value function $\mathbf{Q}^{\boldsymbol{\pi}_{\mathbf{w}}}(\mathbf{s}, \mathbf{a})$ under policy $\boldsymbol{\pi}_{\mathbf{w}}$ as *one single neural network* $\mathbf{Q}(\mathbf{s}, \mathbf{a}, \mathbf{w})$ that treats the weight vector $\mathbf{w}$ as an input in addition to the state $\mathbf{s}$ and action $\mathbf{a}$. Our proposed training algorithm guarantees that $\mathbf{Q}(\mathbf{s}, \mathbf{a}, \mathbf{w})$ will converge to $\mathbf{Q}^{\boldsymbol{\pi}_{\mathbf{w}}}(\mathbf{s}, \mathbf{a})$ theoretically. As long as $\mathbf{Q}(\mathbf{s}, \mathbf{a}, \mathbf{w})$

is well-trained, $\boldsymbol{\pi}_{\mathbf{w}}(\mathbf{s})$ for any $\mathbf{s}$ and $\mathbf{w}$ could then be well approximated by $\mathbf{a}^* = \arg\max_{\mathbf{a}} \mathbf{wQ}(\mathbf{s}, \mathbf{a}, \mathbf{w})$. In this way, our model can produce the Pareto-optimal policy for any weight vector scalably.

To enable distributed decision making which avoids the computationally intractable operation of searching for $\mathbf{a}^*$ in the entire joint action space, we design a novel *multi-objective value decomposition network (MOVDN)* shown in Figure 2 as the structure of $\mathbf{Q}(\mathbf{s}, \mathbf{a}, \mathbf{w})$. In MOVDN, there exists a neural network $\mathbf{Q}_i(\mathbf{o}_i, a_i, \mathbf{w})$ (referred to as an *agent network*) for each agent i, whose inputs include agent i's observation $\mathbf{o}_i$ and action a_i, and the weight vector $\mathbf{w}$. Furthermore, MOVDN implements another neural network (referred to as the *mixing network*), which takes as inputs the global state, all agents' observations, as well as all agent networks' outputs, and yields the final output for $\mathbf{Q}(\mathbf{s}, \mathbf{a}, \mathbf{w})$.

After MOVDN is well-trained, the mixing network is no longer used, and each agent i distributedly select its action by maximizing its own $\mathbf{wQ}_i(\mathbf{o}_i, a_i, \mathbf{w})$ during execution. In fact, such distributed execution is equivalent to maximizing the global team rewards, because our carefully designed mixing network (further elaborated in Section IV-B2) ensures that the optimal joint action that maximizes $\mathbf{wQ}(\mathbf{s}, \mathbf{a}, \mathbf{w})$ is consistent with the collection of each agent i's optimal action that individually maximizes $\mathbf{wQ}_i(\mathbf{o}_i, a_i, \mathbf{w})$, i.e.,

$$\arg\max_{\mathbf{a}} \mathbf{wQ}(\mathbf{s}, \mathbf{a}, \mathbf{w}) = \Big[\arg\max_{a_i} \mathbf{wQ}_i(\mathbf{o}_i, a_i, \mathbf{w})\Big]_{i \in \mathcal{N}_t},$$

which is the *individual global maximization (IGM)* property.

B. MOVDN Framework

1) Agent Network: As aforementioned, each agent i's network $\mathbf{Q}_i(\mathbf{o}_i, a_i, \mathbf{w})$ takes its observation $\mathbf{o}_i$, its action a_i, and the weight vector $\mathbf{w}$ as inputs. The agent network then multiplies the inputs $\mathbf{o}_i$, a_i, and $\mathbf{w}$ respectively by matrices $\mathbf{W}_o$, $\mathbf{W}_a$, and $\mathbf{W}_w$ to transform them into embeddings, which are further concatenated and fed to a series of MLP layers to obtain a vector output $\mathbf{Q}_i(\mathbf{o}_i, a_i, \mathbf{w}) \in \mathbb{R}^2$. That is, the entire operation of each agent i's network is

$$\mathbf{Q}_i(\mathbf{o}_i, a_i, \mathbf{w}) = g\big([\mathbf{W}_o\mathbf{o}_i || \mathbf{W}_a a_i || \mathbf{W}_w \mathbf{w}]\big), \tag{1}$$

where $||$ denotes the concatenation operation of two vectors, and $g(\cdot)$ denotes the computation of the MLP layers. Due to the fact that $\mathcal{A}_i^t$ of each agent i will change over time in our setting, we accept a_i as an input, and evaluate all available actions for decision-making.

2) Mixing Network: Although only the agent networks are used in the execution stage, MOVDN introduces a mixing network to help us train the agent networks that satisfy the aforementioned IGM property, such that agents are able to work cooperatively to maximize the system-wide goals in a decentralized manner. Specifically, the mixing network takes all agent networks' outputs, as well as the global state and all agents' observations as inputs. It then feeds these values into a multi-head attention mechanism [5], whose output is further added to a bias term $\mathbf{c}(\mathbf{s})$ obtained by feeding the state $\mathbf{s}$ into a stack of several MLP layers. That is, the entire operation of the mixing network at each time slot t is

$$\mathbf{Q}(\mathbf{s}, \mathbf{a}, \mathbf{w}) = \sum_{h=1}^{H} \sum_{i \in \mathcal{N}_t} \lambda_{i,h} \mathbf{Q}_i(\mathbf{o}_i, a_i, \mathbf{w}) + \mathbf{c}(s), \tag{2}$$

where H denotes the number of heads, and the weight $\lambda_{i,h}$ that corresponds to each head h of each agent i is calculated by jointly performing a scaled dot-product and softmax normalization as

$$\lambda_{i,h} = \frac{\exp\Big(\big(\mathbf{o}_i \mathbf{W}_{k,h} \mathbf{W}_{q,h} \mathbf{s}^T\big) \frac{1}{\sqrt{F}}\Big)}{\sum_{j \in \mathcal{N}_t} \exp\Big(\big(\mathbf{o}_j \mathbf{W}_{k,h} \mathbf{W}_{q,h} \mathbf{s}^T\big) \frac{1}{\sqrt{F}}\Big)}$$

with $\mathbf{W}_{q,h}$ and $\mathbf{W}_{k,h}$ denoting respectively the matrices that transform $\mathbf{s}$ into a query and $\mathbf{o}_i$'s into keys, and F denoting the number of rows that the matrix $\mathbf{W}_{q,h}$ has.

Note that, in each time slot t, the summation in the denominator of $\lambda_{i,h}$ is over the set of agents $\mathcal{N}_t$ that actually take actions in the current time slot. Thus, such a mixing network is applicable in our FVCS-CMOMG where $\mathcal{N}_t$ changes in every time slot. By our carefully designed mixing network, MOVDN achieves the IGM property as shown in Theorem 1.

Theorem 1. *MOVDN satisfies the IGM property.*

Proof. By our design of the mixing network given by Equation (2), we have that

$$\begin{aligned}\mathbf{wQ}(\mathbf{s}, \mathbf{a}, \mathbf{w}) =& \mathbf{w} \sum_{h=1}^{H} \sum_{i \in \mathcal{N}_t} \lambda_{i,h} \mathbf{Q}_i(\mathbf{o}_i, a_i, \mathbf{w}) + \mathbf{wc}(\mathbf{s}) \\ =& \sum_{i \in \mathcal{N}_t} \left(\sum_{h=1}^{H} \lambda_{i,h} \mathbf{wQ}_i(\mathbf{o}_i, a_i, \mathbf{w}) \right) + \mathbf{wc}(\mathbf{s}).\end{aligned}$$

Then, we let $\widehat{\lambda}_i = \sum_{h=1}^{H} \lambda_{i,h}$, and have $\widehat{\lambda}_i > 0$ for each i as each $\lambda_{i,h}$ is the output of a softmax calculation. Thus,

$$\begin{aligned}\mathbf{a}^* =& \arg\max_{\mathbf{a}} \mathbf{wQ}(\mathbf{s}, \mathbf{a}, \mathbf{w}) = \arg\max_{\mathbf{a}} \sum_{i \in \mathcal{N}_t} \widehat{\lambda}_i \mathbf{wQ}_i(\mathbf{o}_i, a_i, \mathbf{w}) \\ =& \big[\arg\max_{a_i} \mathbf{wQ}_i(\mathbf{o}_i, a_i, \mathbf{w})\big]_{i \in \mathcal{N}_t},\end{aligned}$$

which proves that MOVDN satisfies the IGM property. □

C. MOVDN Training Algorithm

1) Algorithm Overview: We provide the overall training algorithm of MOVDN in Algorithm 1. Firstly, the algorithm initializes a replay buffer $\mathcal{D}$ that is used to store experiences as an empty set, and randomly initializes the parameters of all agent networks and the mixing network (line 2). The algorithm then enters an episodic training process (line 3-16). In each episode, agents interact with the environment under uniformly sampled weight vector, and the experiences of such interactions are collected into buffer $\mathcal{D}$ (line 5-10). Then, the algorithm samples a random mini-batch of experiences from $\mathcal{D}$ (line 11), and updates the parameters of MOVDN by minimizing the TD error over the mini-batch (line 12-16).

2) Experience Collection: At the start of each episode, the algorithm samples a weight vector $\mathbf{w}$ from Ω (line 4), under which agents make decisions. At the beginning of each

Algorithm 1: MOVDN Training Algorithm

1 // Initialization.
2 Initialize replay buffer $\mathcal{D}$ as $\emptyset$, and randomly initialize the parameters $\boldsymbol{\theta}$ of MOVDN;
// Episodic training process.
3 **foreach** *episode* $n = 0$ *to max-episodes* **do**
// Experience collection.
4 Uniformly sample a weight vector $\mathbf{w}$ from Ω;
5 **foreach** *time slot* $t \in \mathcal{T}$ **do**
6 **foreach** *agent* $i \in \mathcal{N}_t$ **do**
7 Observe $\mathbf{o}_i$ and generate a_i by Equation (3);
8 Collect joint observation $\mathbf{o}$, and joint action $\mathbf{a}$;
9 Observe global vector reward $\mathbf{r}$, next state $\mathbf{s}'$, and next joint observation $\mathbf{o}'$;
10 Store experience $(\mathbf{s}, \mathbf{o}, \mathbf{a}, \mathbf{r}, \mathbf{s}', \mathbf{o}', \mathbf{w})$ in $\mathcal{D}$;
// Parameter updating.
11 Uniformly sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$;
12 **foreach** *experience* $\mathbf{b}_k = (\mathbf{s}, \mathbf{o}, \mathbf{a}, \mathbf{r}, \mathbf{s}', \mathbf{o}', \mathbf{w}) \in \mathcal{B}$ **do**
// TD target computation.
13 $(\mathbf{a}', \mathbf{w}') \leftarrow \arg\max_{\widetilde{\mathbf{a}}, \widetilde{\mathbf{w}}} \mathbf{w}\mathbf{Q}(\mathbf{s}', \widetilde{\mathbf{a}}, \widetilde{\mathbf{w}})$;
14 $\mathbf{y}_k \leftarrow \mathbf{r} + \gamma \mathbf{Q}(\mathbf{s}', \mathbf{a}', \mathbf{w}')$;
// Squared TD error computation.
15 $L_k(\boldsymbol{\theta}) = ||\mathbf{y}_k - \mathbf{Q}(\mathbf{s}, \mathbf{a}, \mathbf{w})||_2^2$;
16 Update parameters $\boldsymbol{\theta}$ by minimizing the loss function $L(\boldsymbol{\theta}) = \frac{1}{|\mathcal{B}|} \sum_{k:\mathbf{b}_k \in \mathcal{B}} L_k(\boldsymbol{\theta})$;

time slot t, each agent i who is about to take an action gets observation $\mathbf{o}_i$, and takes action a_i ϵ-greedily (line 6-7) as

$$a_i = \begin{cases} \arg\max_{a \in \mathcal{A}_i^t} \mathbf{w}\mathbf{Q}_i(\mathbf{o}_i, a, \mathbf{w}), & \text{w.p. } 1-\epsilon, \\ \text{uniformly sampled action in } \mathcal{A}_i^t, & \text{w.p. } \epsilon, \end{cases} \tag{3}$$

with $\epsilon \in (0, 1)$. At the end of each time slot t, the environment returns a vector reward $\mathbf{r}$, and transits to the next state $\mathbf{s}'$ (line 9). The algorithm then stores the experience $(\mathbf{s}, \mathbf{o}, \mathbf{a}, \mathbf{r}, \mathbf{s}', \mathbf{o}', \mathbf{w})$ into buffer $\mathcal{D}$ (line 10). Such an experience collection procedure repeats until the end of training.

3) Parameter Updating: In the parameter updating phase of each episode, the algorithm samples a mini-batch of experiences $\mathcal{B}$ uniformly at random from the buffer $\mathcal{D}$ (line 11). For each experience $\mathbf{b}_k = (\mathbf{s}, \mathbf{o}, \mathbf{a}, \mathbf{r}, \mathbf{s}', \mathbf{o}', \mathbf{w})$ in $\mathcal{B}$, the algorithm firstly finds the $(\mathbf{a}', \mathbf{w}')$ that solves $\max_{\widetilde{\mathbf{a}}, \widetilde{\mathbf{w}}} \mathbf{w}\mathbf{Q}(\mathbf{s}', \widetilde{\mathbf{a}}, \widetilde{\mathbf{w}})$ (line 13). In our actual implementation of Algorithm 1, we restrict $\mathbf{w}'$ to be in a discrete and finite set Ω' uniformly sampled from Ω for computational efficiency. For each $\widetilde{\mathbf{w}} \in \Omega'$, instead of directly searching in the joint action space for the joint action $\mathbf{a}_{\widetilde{\mathbf{w}}}$ that maximizes $\mathbf{w}\mathbf{Q}(\mathbf{s}', \widetilde{\mathbf{a}}, \widetilde{\mathbf{w}})$, we set each agent i's action as $\arg\max_{a_i'} \mathbf{w}\mathbf{Q}_i(\mathbf{o}_i', a_i', \widetilde{\mathbf{w}})$, and combine them as a vector of actions to obtain $\mathbf{a}_{\widetilde{\mathbf{w}}}$. We then search the weight vector $\mathbf{w}' = \arg\max_{\widetilde{\mathbf{w}}} \mathbf{w}\mathbf{Q}(\mathbf{s}', \mathbf{a}_{\widetilde{\mathbf{w}}}, \widetilde{\mathbf{w}})$ within the space Ω', and finally set the value for $(\mathbf{a}', \mathbf{w}')$ as $(\mathbf{a}_{\widetilde{\mathbf{w}}'}, \mathbf{w}')$. After obtaining $(\mathbf{a}', \mathbf{w}')$, the algorithm sets the TD target $\mathbf{y}_k$ as $\mathbf{r} + \gamma \mathbf{Q}(\mathbf{s}', \mathbf{a}', \mathbf{w}')$ (line 14), and the squared TD error $L_k(\boldsymbol{\theta})$ of experience $\mathbf{b}_k$ as $||\mathbf{y}_k - \mathbf{Q}(\mathbf{s}, \mathbf{a}, \mathbf{w})||_2^2$ (line 15). Finally, the algorithm updates the parameters $\boldsymbol{\theta}$ of MOVDN by minimizing the following loss function (line 16)

$$L(\boldsymbol{\theta}) = \frac{1}{|\mathcal{B}|} \sum_{k:\mathbf{b}_k \in \mathcal{B}} L_k(\boldsymbol{\theta})$$

via stochastic gradient descent, which represents the mean squared TD error over the mini-batch $\mathcal{B}$.

D. Convergence Analysis

We now prove the convergence of Algorithm 1. To improve the readability, we leave the proofs for all the following theorems and lemmas in the appendix. Firstly, we present in Lemma 1 that Algorithm 1 is an approximation of repeatedly applying an augmented Bellman optimality operator.

Lemma 1. *Given that* $\mathbf{Q}(\mathbf{s}, \mathbf{a}, \mathbf{w})$ *is represented in a tabular form and* $\mathbb{E}_{\mathbf{s}' \sim P(\mathbf{s}'|\mathbf{s},\mathbf{a})}[\cdot]$ *is calculated exactly, the parameter updating process in Algorithm 1 is equivalent to applying the operator* $\mathcal{H}$*, which is defined for any* $(\mathbf{s}, \mathbf{a}, \mathbf{w})$ *as*

$$\mathcal{H}\mathbf{Q}(\mathbf{s}, \mathbf{a}, \mathbf{w}) = \mathbf{r}(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim P(\mathbf{s}'|\mathbf{s},\mathbf{a})}\big[\mathbf{Q}(\mathbf{s}', \mathbf{a}', \mathbf{w}')\big],$$

where $(\mathbf{a}', \mathbf{w}') = \arg\max_{\widetilde{\mathbf{a}}, \widetilde{\mathbf{w}}} \mathbf{w}\mathbf{Q}(\mathbf{s}', \widetilde{\mathbf{a}}, \widetilde{\mathbf{w}})$.

We use $\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})$ to denote the optimal function that accurately fits $\mathbf{Q}^{\boldsymbol{\pi}_\mathbf{w}}(\mathbf{s}, \mathbf{a})$ for any $(\mathbf{s}, \mathbf{a}, \mathbf{w})$. The following theorem demonstrates that $\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})$ is a fixed point of $\mathcal{H}$.

Theorem 2. $\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})$ *is a fixed point of* $\mathcal{H}$*, i.e.,*

$$\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}) = \mathcal{H}\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}), \forall \mathbf{s}, \mathbf{a}, \mathbf{w}.$$

In the following, we use $\mathbf{Q}_k(\mathbf{s}, \mathbf{a}, \mathbf{w})$ to denote the function produced by applying for k times the operator $\mathcal{H}$. Naturally, the immediate reward r_t^o is bounded. Without loss of generality, we choose such α_g and $f(\cdot)$ that the immediate reward r_t^s defined in Definition 2 is also bounded. $\mathbf{Q}_0(\mathbf{s}, \mathbf{a}, \mathbf{w})$ is initialized as zero for each $(\mathbf{s}, \mathbf{a}, \mathbf{w})$. We now prove that $\mathcal{H}$ is a contraction at $\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})$ in Theorem 3, which is also known as the pseudo-contraction [6].

Theorem 3. *Let* d *be a metric that satisfies*

$$d(\mathbf{Q}, \mathbf{Q}') = \max_{\mathbf{s},\mathbf{a},\mathbf{w}} \big|\mathbf{w}\big(\mathbf{Q}(\mathbf{s}, \mathbf{a}, \mathbf{w}) - \mathbf{Q}'(\mathbf{s}, \mathbf{a}, \mathbf{w})\big)\big|, \forall \mathbf{Q}, \mathbf{Q}'.$$

Then, we have $d\big(\mathcal{H}\mathbf{Q}_k, \mathcal{H}\mathbf{Q}^*\big) \leq \gamma d\big(\mathbf{Q}_k, \mathbf{Q}^*\big), \forall k \geq 0$*, i.e.,* $\mathcal{H}$ *is a pseudo-contraction at* $\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})$.

Based on Theorem 2 and 3, we are now in position to present the convergence of Algorithm 1 in Theorem 4.

Theorem 4. *By iteratively applying* $\mathcal{H}$ *for infinite times, the distance given by* d *between* $\mathbf{Q}_k(\mathbf{s}, \mathbf{a}, \mathbf{w})$ *and* $\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})$ *will converge to zero, i.e.,* $\lim_{k \to \infty} d\big(\mathbf{Q}_k, \mathbf{Q}^*\big) = 0$.

According to the definition of CCS in Definition 4, $\mathbf{Q}_k(\mathbf{s}, \mathbf{a}, \mathbf{w})$ is as optimal as $\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})$ when k approaching infinity, since they have the same value under $\mathbf{w}$. Theorem 4 thus provides a desirable theoretical convergence guarantee of applying the proposed operator $\mathcal{H}$.

E. Sample Complexity Analysis

Recall that the operator $\mathcal{H}$ jointly computes $(\mathbf{a}', \mathbf{w}') = \arg\max_{\widetilde{\mathbf{a}}, \widetilde{\mathbf{w}}} \mathbf{w}\mathbf{Q}(\mathbf{s}', \widetilde{\mathbf{a}}, \widetilde{\mathbf{w}})$, which differs from the corresponding operation in traditional Bellman optimality operator that would only search the joint action that maximizes $\mathbf{w}\mathbf{Q}(\mathbf{s}', \widetilde{\mathbf{a}}, \mathbf{w})$. As we will see in Theorem 5, such additional

dimension of optimization that $\mathcal{H}$ conducts over the weight vector space greatly benefits the sample complexity for learning the policies in CCS.

We adopt the widely used generative model setting [7, 8] for sample complexity analysis, which assumes the existence of an oracle that takes as input a state-action pair $(\mathbf{s}, \mathbf{a})$, and returns a sampled $\mathbf{s}' \sim P(\cdot|\mathbf{s}, \mathbf{a})$ and the reward $\mathbf{r}(\mathbf{s}, \mathbf{a})$. Under such a setting, we collect K transitions for each state-action pair $(\mathbf{s}, \mathbf{a})$. Based on these transitions, we iteratively apply $\mathcal{H}$ for infinite times to obtain a joint state-action value function $\widehat{\mathbf{Q}}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}; K)$. The policy induced by $\widehat{\mathbf{Q}}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}; K)$ for each $\mathbf{w}$ is denoted as $\widehat{\boldsymbol{\pi}}^K_{\mathbf{w}}$.

Theorem 5. *Under the generative model setting, given $\delta \in (0, 1)$ and $\epsilon > 0$, there exists constants c_1, c_2, if*

$$K \geq \frac{c_1 m}{\epsilon^2 (1-\gamma)^3} \log \frac{c_2 m |\mathcal{S}||\mathcal{A}|}{(1-\gamma)\epsilon\delta},$$

where m is the number of objectives, $\mathcal{S}$ is the state space, and $\mathcal{A}$ is the joint action space, we have $d\big(\mathbf{Q}^{\widehat{\boldsymbol{\pi}}^K_{\mathbf{w}}}, \mathbf{Q}^*\big) \leq \epsilon$ with probability at least $1-\delta$, i.e., $\widehat{\boldsymbol{\pi}}^K_{\mathbf{w}}$ is $\epsilon-$optimal almost surely.

Theorem 5 implies that the sample complexity of applying $\mathcal{H}$ for learning CCS is independent of the weight vector space Ω. In contrast, the naive approach that trains one policy for each $\mathbf{w}$ causes the sample complexity to depend on the cardinality of Ω, as it needs to repeatedly collect experiences for $|\Omega|$ times. Therefore, the operator $\mathcal{H}$ used in Algorithm 1 greatly benefits the sample complexity for learning CCS.

V. Experiments

A. Experimental Setups

1) Dataset: Our experiments are based on three real-world ride-hailing order datasets, two of which are released by Didi Chuxing GAIA Initiative [9], and one of which is released by New York City Taxi and Limousine Commission [10]. The first dataset contains over 2 million orders in Haikou, China, from September 1st to September 30th, 2017. The second dataset contains over 5 million orders in Chengdu, China, from November 1st to November 30th, 2016. The last dataset contains over 6 million orders in New York City, America, from September 1st to September 30th, 2019.

2) Simulator: Based on the datasets, we design an urban crowd sensing simulator to support the training and evaluation of MOVDN. Specifically, we divide the city area into square grids whose lengths are approximately 2 kilometers. The location of each driver or order is represented by a grid ID. For each grid g, we set the function $f(\cdot)$ as natural logarithm and the coefficient α_g as 1.0 in the following experiments. The length of each time slot is set as 2 minutes. We assume orders will get cancelled, if not served for more than 3 time slots since they are submitted. To evaluate MOVDN under different traffic conditions, we conduct experiments in three time periods, i.e., the morning period from 7 a.m. to 11 a.m., the noon period from 11 a.m. to 3 p.m., and the evening period from 3 p.m. to 7 p.m.. Similar to [1, 2], there are 500 FHVs in our simulation, whose initial locations are randomly sampled from a discrete uniform distribution defined over the grids.

3) Baselines: Since there is no off-the-shelf baselines, we adapt the state-of-the-art algorithms that are capable to fit into our setting for comparison. (I) **CVN-S**: CVN [3] is a state-of-the-art order dispatch framework, where each agent aims to maximize its own cumulative reward. To make CVN suitable in our setting, the immediate global vector reward $\mathbf{r}$ is decomposed into each agent's individual vector reward $\mathbf{r}_i$, which consists of the order-serving reward (i.e., the price of the order chosen by the agent), and the sensing reward (i.e., the average global sensing utility). CVN-S uses scalarized Q-learning [11] to train CVN, which utilizes a traditional Bellman optimality operator. (II) **CVN-E**: CVN-E also follows CVN to make order selection decisions. The only difference between CVN-E and CVN-S is that CVN-E uses the same training algorithm as MOVDN. (III) **SOVDN**: SOVDN modifies MOVDN to output a scalar value that estimates the function $\mathbf{w}\mathbf{Q}^{\boldsymbol{\pi}_{\mathbf{w}}}(\mathbf{s}, \mathbf{a})$ for each $(\mathbf{s}, \mathbf{a}, \mathbf{w})$, and is trained using multi-objective fitted Q-iteration [6]. (IV) **VDN-S**: VDN-S has the same framework as MOVDN, but is trained by scalarized Q-learning.

4) Metrics: In our experiments, after training MOVDN and the baselines, we test each of them under the same set of weight vectors $\overline{\Omega} = \big\{\mathbf{w} \in \Omega | w_1 = 0.05n, n \in \{1, 2, \cdots, 20\}\big\}$. Specifically, for each method, we collect its GMV and SSU under each weight vector $\mathbf{w} \in \overline{\Omega}$, filter out the dominated outcomes, and refer to the remaining set of undominated ones as the *Pareto frontier* of the method under $\overline{\Omega}$. We then compare the performances of all methods by evaluating their Pareto frontiers using two common metrics *hypervolume* and *coverage of two sets (CTS)* [4]. More specifically, we treat any solution $\mathbf{J} = [J_1, J_2]$ in a Pareto frontier $\mathcal{J}$ as a point in a 2-D x-y coordinate system, and denote the rectangular area bounded by $x = 0$, $y = 0$, $x = J_1$, and $y = J_2$ as $\mathcal{R}(\mathbf{J})$. The hypervolume of $\mathcal{J}$ represents the area of $\bigcup_{\mathbf{J} \in \mathcal{J}} \mathcal{R}(\mathbf{J})$. A larger hypervolume indicates a better Pareto frontier. Furthermore, given two Pareto frontiers $\mathcal{J}$ and $\mathcal{J}'$, the CTS of $\mathcal{J}$ to $\mathcal{J}'$ is the fraction of outcomes in $\mathcal{J}'$ that are dominated by at least one outcome in $\mathcal{J}$. We are only interested in, and thus only show the CTS of MOVDN's Pareto frontier to those of the baseline methods in Section V-B.

B. Experimental Results

1) Metrics Comparison: Table I and Table II compare the results of the hypervolume and CTS respectively. Table I demonstrates that our algorithm achieves the largest hypervolume compared with baselines in all scenarios. Table II further shows that more than 90% solutions computed by baselines are dominated by ours. Therefore, the Pareto frontier constructed by MOVDN is superior to those constructed by baselines.

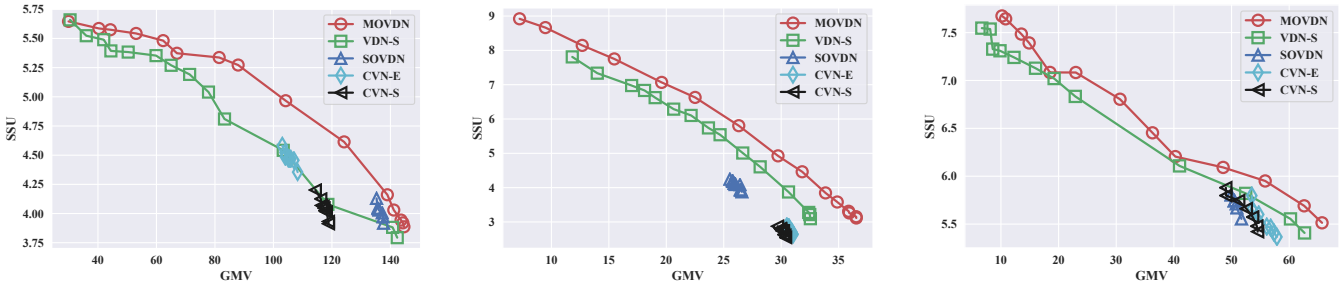
2) Pareto Frontier Visualization: To see clearly the relationship between GMV and SSU, we visualize the Pareto frontiers of all considered methods in Figure 3, from which we can observe that GMV and SSU are two conflicting goals that need to be balanced appropriately. Overall, MOVDN's Pareto frontier covers a wider area and is more evenly distributed

TABLE I: Hypervolume comparison of MOVDN and baselines in three periods on three datasets. The values are written in scientific notations that omit $\times 10^8$. Each value denotes the mean and standard deviation over 6 runs with different seeds.

Method	Morning			Noon			Evening		
	Haikou	Chengdu	NYC	Haikou	Chengdu	NYC	Haikou	Chengdu	NYC
CVN-S	5.02 ± 0.07	0.94 ± 0.10	3.08 ± 0.11	5.04 ± 0.37	0.72 ± 0.04	3.27 ± 0.12	4.67 ± 0.26	0.75 ± 0.03	3.27 ± 0.13
CVN-E	4.65 ± 0.22	1.03 ± 0.13	3.11 ± 0.14	4.91 ± 0.38	0.73 ± 0.06	3.16 ± 0.03	5.08 ± 0.11	0.77 ± 0.01	3.26 ± 0.09
SOVDN	5.37 ± 0.32	1.03 ± 0.07	3.02 ± 0.23	5.12 ± 0.11	0.82 ± 0.13	3.60 ± 0.20	5.05 ± 0.15	0.79 ± 0.02	3.80 ± 0.17
VDN-S	6.65 ± 0.31	1.79 ± 0.22	3.79 ± 0.14	6.75 ± 0.12	1.57 ± 0.16	5.79 ± 0.08	8.26 ± 0.15	1.75 ± 0.15	6.00 ± 0.31
MOVDN	$\mathbf{7.31 \pm 0.10}$	$\mathbf{2.08 \pm 0.26}$	$\mathbf{4.23 \pm 0.10}$	$\mathbf{7.76 \pm 0.12}$	$\mathbf{1.95 \pm 0.03}$	$\mathbf{6.28 \pm 0.05}$	$\mathbf{9.37 \pm 0.25}$	$\mathbf{2.17 \pm 0.03}$	$\mathbf{6.67 \pm 0.38}$

TABLE II: CTS of MOVDN's Pareto frontier to those of baseline methods in three periods on three datasets. Each value denotes the mean and standard deviation over 6 runs with different seeds.

Method	Morning			Noon			Evening		
	Haikou	Chengdu	NYC	Haikou	Chengdu	NYC	Haikou	Chengdu	NYC
CVN-S	0.98 ± 0.02	0.97 ± 0.04	0.97 ± 0.05	0.97 ± 0.04	0.96 ± 0.05	0.98 ± 0.04	0.96 ± 0.06	0.96 ± 0.04	0.98 ± 0.04
CVN-E	0.97 ± 0.05	0.95 ± 0.06	0.97 ± 0.04	0.96 ± 0.04	0.95 ± 0.05	0.96 ± 0.05	0.97 ± 0.04	0.95 ± 0.06	0.98 ± 0.03
SOVDN	0.97 ± 0.04	0.97 ± 0.04	0.96 ± 0.05	0.96 ± 0.05	0.97 ± 0.04	0.93 ± 0.06	0.94 ± 0.07	0.93 ± 0.07	0.96 ± 0.06
VDN-S	0.91 ± 0.03	0.94 ± 0.01	0.92 ± 0.06	0.92 ± 0.06	0.95 ± 0.03	0.91 ± 0.06	0.93 ± 0.05	0.91 ± 0.03	0.91 ± 0.03



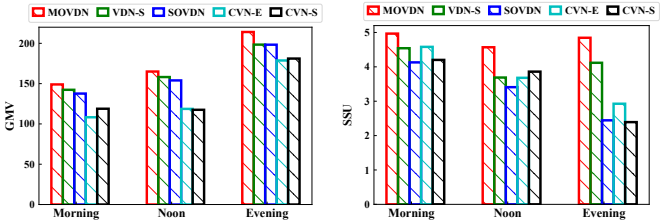
(a) Haikou (Morning). (b) Chengdu (Morning). (c) New York City (Morning).

Fig. 3: Pareto frontier visualization in morning period. Both GMV and SSU are written in scientific notations that omit $\times 10^3$.

than the Pareto frontiers of all baselines, which validates the results in Table I and II.

Figure 3 shows that the outcomes of both CVN-S and CVN-E gather in a small range, and are mostly dominated by MOVDN. Our reasoning of MOVDN's superior performance compared with CVN-S and CVN-E is as follows. First, in CVN-S and CVN-E, each agent learns independently to maximize its own cumulative reward instead of the team reward. Thus, agents compete with each other, which will degrade the system-wide performance. Second, the equal allocation of the team reward to each agent in CVN-S and CVN-E cannot precisely represent each agent's credit. Therefore, it is hard for agents to learn proper policies for team reward maximization for any weight vector.

Compared with MOVDN, SOVDN adopts a similar framework but outputs a scalar $Q(\mathbf{s}, \mathbf{a}, \mathbf{w})$ that fits $\mathbf{w}\mathbf{Q}^{\boldsymbol{\pi}_\mathbf{w}}(\mathbf{s}, \mathbf{a})$ for each $(\mathbf{s}, \mathbf{a}, \mathbf{w})$. However, SOVDN is not able to distinguish the input weight vectors, which can be seen from the following example. Suppose $\mathbf{r}_1$ is the cumulative vector reward if agents start in state $\mathbf{s}$ and always act according to $\boldsymbol{\pi}_{\mathbf{w}_1}$, $\mathbf{r}_2$ is the cumulative vector reward if agents also start in state $\mathbf{s}$ but act according to a different policy $\boldsymbol{\pi}_{\mathbf{w}_2}$, and $\mathbf{w}_1\mathbf{r}_1 = \mathbf{w}_2\mathbf{r}_2 = C$. Then, SOVDN learns towards minimizing the gap between $\max_\mathbf{a} Q(\mathbf{s}, \mathbf{a}, \mathbf{w})$ and C, no matter whether the input weight vector $\mathbf{w}$ is $\mathbf{w}_1$ or $\mathbf{w}_2$. In that case, the neural network will homogenize the inputs, and fails to learn different policies for different weight vectors. In Figure 3, the outcomes of SOVDN



(a) Prefer GMV scenario. (b) GMV no-less-than scenario.

Fig. 4: Two scenarios in Haikou dataset. ρ_o are set as 4500. The values omit $\times 10^3$.

almost gather into one point, which validates our thoughts.

VDN-S is the most competitive baseline as it uses the same framework as MOVDN. However, VDN-S trains each weight vector separately, and does not use the information of the weight vectors that have been explored before, which makes it suffer from sample inefficiency and performs less satisfactorily than our MOVDN.

C. Discussion of Usage

Next, we discuss how MOVDN supports the platform for decision making in practice. First, the platform can evaluate MOVDN under a large number of weight vectors offline, and stores the GMV and SSU of each weight vector in advance. Then in the planning phase, the platform chooses one weight vector whose outcomes meet its requirements, and sends that weight vector to FHVs. FHVs' decision modules take as input such a weight vector to make order selection decisions. If

the platform changes its requirements on order-serving and sensing objectives, it only needs to update the weight vector to FHVs. In the following, we design 2 scenarios that are often met by platforms in practice. (I) **Prefer GMV scenario**: The platform wants to maximize GMV. (II) **GMV no-less-than scenario**: The platform wants to maximize SSU under the condition that GMV is no less than a threshold ρ_o. In our experiments, we evaluate all methods under $\overline{\Omega}$. As shown in Figure 4, MOVDN achieves the maximum outcomes in 2 scenarios, which demonstrates that MOVDN can support the platform in decision-making effectively.

VI. Related Work

Researchers have devoted much effort [12–38] to design mobile crowd sensing (MCS) systems. Among them, [12–16] design incentive mechanisms to motivate workers' participation, [17–22] conduct truth inference for the collected data, [23–26] focus on preserving workers' privacy, [27, 28] investigate pricing strategies, and [29–33] utilize RL to navigate unmanned vehicles to optimize the data collection. A series of works [34–38] study task assignment problems, where [36–38] optimize the system utility under time or budget constrains, and [34, 35] takes workers' preferences on tasks into account when assigning tasks. Different from the above works, we focus on the FVCS system and study an essential unsolved problem of balancing the sensing and order-serving objectives according to the platforms' preferences.

Similar to this paper, [1, 2] also deal with the inconsistency of sensing and order-serving objectives in FVCS systems. Specifically, [2] prioritizes sensing over order-serving and incentivizes FHVs with a limited budget. [1] maximizes a compound objective of order-serving and sensing by designing a route planning mechanism. However, existing works fail to meet the diverse needs of FVCS platforms. Our paper differs from such works on two aspects. First, we optimize the order selection mechanism for FHVs instead of the incentive mechanism or route planning mechanism. Second, we formulate the problem from a multi-objective view, and solve a set of Pareto-optimal policies for platforms. To the best of our knowledge, we are the first work to obtain CCS in support of the platform.

MOVDN itself is also a novel multi-objective MARL framework. Specifically, the independent learning approach [39] train a policy for each agent to maximize its own cumulative reward. [40] stimulates multi-agent cooperation by designing a counterfactual baseline for credit assignment. [41, 42] coordinates agents' actions by designing value decomposition networks that satisfy the IGM principle. MOVDN also falls into this category. However, previous works focus on the multi-agent cooperation with single objective, while our problem contains multiple conflicting objectives that needs to be balanced according to the platforms' preferences.

VII. Conclusion

We have proposed a cooperative multi-objective MARL framework, referred to as MOVDN, to help FHVs make order selection decisions in FVCS systems. MOVDN is preference-configurable that allows platforms to flexibly determine their preferences on order-serving and sensing objectives. Specifically, MOVDN adopts a decomposed network structure to factorize the joint state-action value function among agents, where each agent possesses a network to take actions based on its local observations, and a mixing network ensures that the optimal joint action is consistent with the collection of each agent's optimal action. To obtain the Pareto-optimal policy for any preference, we propose a novel algorithm to train a single universal MOVDN that is optimized over the space of all preferences. The key of our algorithm is utilizing an augmented Bellman optimality operator, which enjoys theoretical convergence guarantee and is provably more sample-efficient than the traditional Bellman optimality operator. Extensive experiments on three real-world ride-hailing order datasets illustrate that MOVDN outperforms strong baselines and can support the platform in decision-making effectively.

Acknowledgment

This work was supported in part by NSF China (No. U21A20519, U20A20181, 62202298), and in part by Fellowship of China Postdoctoral Science Foundation (No. 22Z020702116).

Appendix

VIII. Proof of Lemma 1

Ideally, $\mathcal{H}$ is applied as $\mathbf{Q}_{k+1}(\mathbf{s},\mathbf{a},\mathbf{w}) = \mathcal{H}\mathbf{Q}_k(\mathbf{s},\mathbf{a},\mathbf{w})$. However, as the transition function is unknown in practice, we approximate the expectation in $\mathcal{H}\mathbf{Q}(\mathbf{s},\mathbf{a},\mathbf{w})$ by $\mathbf{Q}(\mathbf{s}',\mathbf{a}',\mathbf{w}')$ (line 15), which is calculated on one sampled next state $\mathbf{s}'$ and is an unbiased approximation. In addition, $\mathbf{Q}(\mathbf{s},\mathbf{a},\mathbf{w})$ is represented by a neural network, whose parameters are updated by minimizing the mean squared TD error over a mini-batch of experiences $\mathcal{B}$, instead of the naive operation of assigning $\mathcal{H}\mathbf{Q}_k(\mathbf{s},\mathbf{a},\mathbf{w})$ to $\mathbf{Q}_{k+1}(\mathbf{s},\mathbf{a},\mathbf{w})$.

IX. Proof of Theorem 2

For any given weight vector $\mathbf{w}$, we have

$$\mathbf{w}\mathbf{Q}^*(\mathbf{s},\mathbf{a},\mathbf{w}) = \mathbf{w}\mathbf{Q}^{\boldsymbol{\pi}_\mathbf{w}}(\mathbf{s},\mathbf{a}) \geq \mathbf{w}\mathbf{Q}^{\boldsymbol{\pi}_{\mathbf{w}'}}(\mathbf{s},\mathbf{a}) = \mathbf{w}\mathbf{Q}^*(\mathbf{s},\mathbf{a},\mathbf{w}')$$

for each $(\mathbf{s},\mathbf{a},\mathbf{w}')$, where the equalities result from the definition of $\mathbf{Q}^*(\mathbf{s},\mathbf{a},\mathbf{w})$, and the inequality holds because of our definition of the CCS given in Definition 4. Thus, we have $\mathbf{w} = \arg\max_{\widetilde{\mathbf{w}}} \mathbf{w}\mathbf{Q}^*(\mathbf{s}',\widetilde{\mathbf{a}},\widetilde{\mathbf{w}})$ for any $(\mathbf{s}',\widetilde{\mathbf{a}})$. Then, we have

$$\begin{aligned}\mathcal{H}\mathbf{Q}^*(\mathbf{s},\mathbf{a},\mathbf{w}) &= \mathbf{r}(\mathbf{s},\mathbf{a}) + \mathbb{E}_{\mathbf{s}'\sim P(\mathbf{s}'|\mathbf{s},\mathbf{a})}\big[\mathbf{Q}^*(\mathbf{s}',\mathbf{a}',\mathbf{w}')\big] \\ &= \mathbf{r}(\mathbf{s},\mathbf{a}) + \mathbb{E}_{\mathbf{s}'\sim P(\mathbf{s}'|\mathbf{s},\mathbf{a})}\big[\mathbf{Q}^*(\mathbf{s}',\boldsymbol{\pi}_\mathbf{w}(\mathbf{s}'),\mathbf{w})\big] \\ &= \mathbf{Q}^*(\mathbf{s},\mathbf{a},\mathbf{w})\end{aligned}$$

for each $(\mathbf{s},\mathbf{a},\mathbf{w})$, which ends the proof of Theorem 2.

X. Proof of Theorem 3

Lemma 2. *$\forall k \geq 0$, $\mathbf{Q}^*(\mathbf{s},\mathbf{a},\mathbf{w})$ and $\mathbf{Q}_k(\mathbf{s},\mathbf{a},\mathbf{w})$ satisfy*

$$\mathbf{w}\mathbf{Q}^*(\mathbf{s},\mathbf{a},\mathbf{w}) \geq \mathbf{w}\mathbf{Q}_k(\mathbf{s},\mathbf{a},\mathbf{w}'), \forall \mathbf{s},\mathbf{a},\mathbf{w},\mathbf{w}'.$$

Proof. We prove Lemma 2 by induction. First, we rewrite the operation of applying $\mathcal{H}$ as

$$\mathbf{Q}_{k+1}(\mathbf{s},\mathbf{a},\mathbf{w}') = \mathbf{r}(\mathbf{s},\mathbf{a}) + \gamma\mathbb{E}\big[\mathbf{Q}_k(\mathbf{s}',\mathbf{a}'',\mathbf{w}'')\big],$$

where $(\mathbf{a}'', \mathbf{w}'') = \arg\max_{\widetilde{\mathbf{a}},\widetilde{\mathbf{w}}} \mathbf{w}'\mathbf{Q}_k(\mathbf{s}', \widetilde{\mathbf{a}}, \widetilde{\mathbf{w}})$.

For $k = 0$, we have $\mathbf{w}\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}) \geq \mathbf{w}\mathbf{Q}_0(\mathbf{s}, \mathbf{a}, \mathbf{w}')$ for any $(\mathbf{s}, \mathbf{a}, \mathbf{w}, \mathbf{w}')$. Suppose for $k = l > 0$, it also holds that $\mathbf{w}\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}) \geq \mathbf{w}\mathbf{Q}_l(\mathbf{s}, \mathbf{a}, \mathbf{w}'), \forall \mathbf{s}, \mathbf{a}, \mathbf{w}, \mathbf{w}'$. For $k = l + 1$,

$$\begin{aligned}\mathbf{w}\mathbf{Q}_{l+1}(\mathbf{s}, \mathbf{a}, \mathbf{w}') &= \mathbf{w}\mathbf{r}(\mathbf{s}, \mathbf{a}) + \gamma\mathbb{E}\big[\mathbf{w}\mathbf{Q}_l(\mathbf{s}', \mathbf{a}'', \mathbf{w}'')\big] \\ &\leq \mathbf{w}\mathbf{r}(\mathbf{s}, \mathbf{a}) + \gamma\mathbb{E}\big[\mathbf{w}\mathbf{Q}^*(\mathbf{s}', \mathbf{a}'', \mathbf{w})\big] \\ &\leq \mathbf{w}\mathbf{r}(\mathbf{s}, \mathbf{a}) + \gamma\mathbb{E}\Big[\max_{\widetilde{\mathbf{a}},\widetilde{\mathbf{w}}} \mathbf{w}\mathbf{Q}^*(\mathbf{s}', \widetilde{\mathbf{a}}, \widetilde{\mathbf{w}})\Big] \\ &= \mathbf{w}\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}).\end{aligned}$$

The last equality holds because $\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})$ is a fixed point of $\mathcal{H}$ as shown in Theorem 2. Therefore, for any $k \geq 0$, $\mathbf{w}\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}) \geq \mathbf{w}\mathbf{Q}_k(\mathbf{s}, \mathbf{a}, \mathbf{w}'), \forall \mathbf{s}, \mathbf{a}, \mathbf{w}, \mathbf{w}'$ holds. □

By the definition of d, we have

$$\begin{aligned}d(\mathcal{H}\mathbf{Q}_k, \mathcal{H}\mathbf{Q}^*) &= \max_{\mathbf{s},\mathbf{a},\mathbf{w}} \Big|\mathbf{w}\big(\mathcal{H}\mathbf{Q}_k(\mathbf{s}, \mathbf{a}, \mathbf{w}) - \mathcal{H}\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})\big)\Big| \\ &= \gamma \max_{\mathbf{w}} \Big|\mathbb{E}\big[\mathbf{w}\mathbf{Q}_k(\mathbf{s}', \mathbf{a}', \mathbf{w}') - \mathbf{w}\mathbf{Q}^*(\mathbf{s}', \mathbf{a}'', \mathbf{w})\big]\Big|,\end{aligned}$$

where $(\mathbf{a}', \mathbf{w}') = \arg\max_{\widetilde{\mathbf{a}},\widetilde{\mathbf{w}}} \mathbf{w}\mathbf{Q}_k(\mathbf{s}', \widetilde{\mathbf{a}}, \widetilde{\mathbf{w}})$, and $(\mathbf{a}'', \mathbf{w}) = \arg\max_{\widehat{\mathbf{a}},\widehat{\mathbf{w}}} \mathbf{w}\mathbf{Q}^*(\mathbf{s}', \widehat{\mathbf{a}}, \widehat{\mathbf{w}})$.

$$\begin{aligned}d(\mathcal{H}\mathbf{Q}_k, \mathcal{H}\mathbf{Q}^*) &\leq \gamma \max_{\mathbf{s}',\mathbf{w}} \big|\mathbf{w}\mathbf{Q}_k(\mathbf{s}', \mathbf{a}', \mathbf{w}') - \mathbf{w}\mathbf{Q}^*(\mathbf{s}', \mathbf{a}'', \mathbf{w})\big| \\ &= \gamma \max_{\mathbf{s}',\mathbf{w}} \big|\max_{\widetilde{\mathbf{a}},\widetilde{\mathbf{w}}} \mathbf{w}\mathbf{Q}_k(\mathbf{s}', \widetilde{\mathbf{a}}, \widetilde{\mathbf{w}}) - \max_{\widehat{\mathbf{a}}} \mathbf{w}\mathbf{Q}^*(\mathbf{s}', \widehat{\mathbf{a}}, \mathbf{w})\big|,\end{aligned}$$

where the first inequality is because $\big|\mathbb{E}[\cdot]\big| \leq \mathbb{E}\big[|\cdot|\big] \leq \max|\cdot|$, and the second equality results from the cancellation of $\arg\max$ operation. Based on Lemma 2, we can easily get $\max_{\widetilde{\mathbf{a}},\widetilde{\mathbf{w}}} \mathbf{w}\mathbf{Q}_k(\mathbf{s}', \widetilde{\mathbf{a}}, \widetilde{\mathbf{w}}) \leq \max_{\widehat{\mathbf{a}}} \mathbf{w}\mathbf{Q}^*(\mathbf{s}', \widehat{\mathbf{a}}, \mathbf{w})$. Thus,

$$\begin{aligned}&d(\mathcal{H}\mathbf{Q}_k, \mathcal{H}\mathbf{Q}^*) \\ &\leq \gamma \max_{\mathbf{s}',\mathbf{w}} \big(\max_{\widehat{\mathbf{a}}} \mathbf{w}\mathbf{Q}^*(\mathbf{s}', \widehat{\mathbf{a}}, \mathbf{w}) - \max_{\widetilde{\mathbf{a}}} \mathbf{w}\mathbf{Q}_k(\mathbf{s}', \widetilde{\mathbf{a}}, \mathbf{w})\big) \\ &= \gamma \max_{\mathbf{s}',\mathbf{w}} \big(\max_{\widehat{\mathbf{a}}} \mathbf{w}\mathbf{Q}^*(\mathbf{s}', \widehat{\mathbf{a}}, \mathbf{w}) - \mathbf{w}\mathbf{Q}_k(\mathbf{s}', \widehat{\mathbf{a}}, \mathbf{w}) + \\ &\qquad \mathbf{w}\mathbf{Q}_k(\mathbf{s}', \widehat{\mathbf{a}}, \mathbf{w}) - \max_{\widetilde{\mathbf{a}}} \mathbf{w}\mathbf{Q}_k(\mathbf{s}', \widetilde{\mathbf{a}}, \mathbf{w})\big) \\ &\leq \gamma \max_{\mathbf{s}',\mathbf{w}} \big(\max_{\widehat{\mathbf{a}}} \mathbf{w}\mathbf{Q}^*(\mathbf{s}', \widehat{\mathbf{a}}, \mathbf{w}) - \mathbf{w}\mathbf{Q}_k(\mathbf{s}', \widehat{\mathbf{a}}, \mathbf{w})\big) \\ &\leq \gamma \max_{\mathbf{s}',\widehat{\mathbf{a}},\mathbf{w}} \big|\mathbf{w}\mathbf{Q}_k(\mathbf{s}', \widehat{\mathbf{a}}, \mathbf{w}) - \mathbf{w}\mathbf{Q}^*(\mathbf{s}', \widehat{\mathbf{a}}, \mathbf{w})\big| = \gamma d(\mathbf{Q}_k, \mathbf{Q}^*),\end{aligned}$$

which proves $\mathcal{H}$ is a pseudo-contraction at $\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})$.

XI. Proof of Theorem 4

Based on Theorem 2 and 3, we have

$$\begin{aligned}d(\mathbf{Q}_k, \mathbf{Q}^*) &= d(\mathcal{H}\mathbf{Q}_{k-1}(\mathbf{s}, \mathbf{a}, \mathbf{w}), \mathcal{H}\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})) \\ &\leq \gamma d(\mathbf{Q}_{k-1}(\mathbf{s}, \mathbf{a}, \mathbf{w}), \mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})) \\ &\leq \gamma^k d(\mathbf{Q}_0(\mathbf{s}, \mathbf{a}, \mathbf{w}), \mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})).\end{aligned}$$

We use $R_{\max}$ to denote the largest possible immediate reward of r_t^o and r_t^s. Then, $||\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})||_\infty \leq \frac{R_{\max}}{1-\gamma}$, which indicates $\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})$ is also bounded. Thus, with k approaching infinity, we have $\lim_{k\to\infty} d\big(\mathbf{Q}_k, \mathbf{Q}^*\big) = 0$.

XII. Proof of Theorem 5

For any $\mathbf{w}_1, \mathbf{w}_2 \in \Omega$, and any $(\mathbf{s}, \mathbf{a})$, we have

$$\begin{aligned}&\mathbf{w}_1\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}_1) - \mathbf{w}_2\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}_2) \\ =&\mathbf{w}_1\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}_1) - \mathbf{w}_2\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}_1) \\ &+ \mathbf{w}_2\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}_1) - \mathbf{w}_2\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}_2) \\ \leq&\mathbf{w}_1\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}_1) - \mathbf{w}_2\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}_1) \\ \leq&||\mathbf{w}_1 - \mathbf{w}_2||_\infty ||\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}_1)||_1 \leq \frac{mR_{\max}}{1-\gamma}||\mathbf{w}_1 - \mathbf{w}_2||_\infty,\end{aligned}$$

where the second inequality holds due to Holder's inequality, and the last one holds since $||\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}_1)||_1 \leq m||\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}_1)||_\infty \leq \frac{mR_{\max}}{1-\gamma}$. By taking maximum over $(\mathbf{s}, \mathbf{a})$, we have

$$\max_{\mathbf{s},\mathbf{a}} |\mathbf{w}_1\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}_1) - \mathbf{w}_2\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}_2)| \frac{mR_{\max}}{1-\gamma}||\mathbf{w}_1 - \mathbf{w}_2||_\infty. \quad (4)$$

Let C_ξ be $\xi-$covering set of Ω in terms of $\ell_\infty-$norm. Then, $\forall \mathbf{w} \in \Omega, \exists \mathbf{w}_\xi \in C_\xi, ||\mathbf{w} - \mathbf{w}_\xi||_\infty \leq \xi$, and C_ξ satisfies $|C_\xi| \leq (\frac{2}{\xi})^m$. Under the generative model setting, we construct an empirical transition probability function $\widehat{P}(\mathbf{s}'|\mathbf{s}, \mathbf{a}) = \frac{count(\mathbf{s},\mathbf{a},\mathbf{s}')}{K}$, where $count(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ is the number of times that $(\mathbf{s}, \mathbf{a})$ transitions to $\mathbf{s}'$. The sub-optimality bound [7] in single-objective RL is that, with probability at least $1-\delta$,

$$||Q^*(\mathbf{s}, \mathbf{a}) - Q^{\widehat{\boldsymbol{\pi}}^K}(\mathbf{s}, \mathbf{a})||_\infty \leq \gamma\sqrt{\frac{c}{(1-\gamma)^3}\frac{\log(c|\mathcal{S}||\mathcal{A}|/\delta)}{K}}, \quad (5)$$

where $Q^{\widehat{\boldsymbol{\pi}}^K}(\mathbf{s}, \mathbf{a})$ is the optimal state-action value function under $\widehat{P}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$. By the definition of $\mathbf{Q}(\mathbf{s}, \mathbf{a}, \mathbf{w})$, for a given $\mathbf{w}$, we have

$$Q^*(\mathbf{s}, \mathbf{a}) = \mathbf{w}\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}), Q^{\widehat{\boldsymbol{\pi}}^K}(\mathbf{s}, \mathbf{a}) = \mathbf{w}\mathbf{Q}^{\widehat{\boldsymbol{\pi}}^K_{\mathbf{w}}}(\mathbf{s}, \mathbf{a}, \mathbf{w}). \quad (6)$$

For the ease of notation, we omit the superscript K of $\widehat{\boldsymbol{\pi}}^K_{\mathbf{w}}$ in the following proof. By substituting Equation (6) into Equation (5) and take an union bound over all $\mathbf{w}_\xi \in C_\xi$, we have that with probability at least $1 - \delta$,

$$\begin{aligned}&\max_{\mathbf{s},\mathbf{a}} |\mathbf{w}_\xi\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}_\xi) - \mathbf{w}_\xi\mathbf{Q}^{\widehat{\boldsymbol{\pi}}_{\mathbf{w}_\xi}}(\mathbf{s}, \mathbf{a}, \mathbf{w}_\xi)| \\ \leq&\gamma\sqrt{\frac{c}{(1-\gamma)^3}\frac{\log(c|\mathcal{S}||\mathcal{A}||C_\xi|/\delta)}{K}}.\end{aligned} \quad (7)$$

Now, we are in position to prove Theorem 5. For any $\mathbf{w} \in \Omega$,

$$\begin{aligned}&\max_{\mathbf{s},\mathbf{a}} |\mathbf{w}\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}) - \mathbf{w}\mathbf{Q}^{\widehat{\boldsymbol{\pi}}_{\mathbf{w}}}(\mathbf{s}, \mathbf{a}, \mathbf{w})| \\ =&\max_{\mathbf{s},\mathbf{a}} |\mathbf{w}\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}) - \mathbf{w}_\xi\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}) + \mathbf{w}_\xi\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}) \\ &\quad - \mathbf{w}_\xi\mathbf{Q}^{\widehat{\boldsymbol{\pi}}_{\mathbf{w}}}(\mathbf{s}, \mathbf{a}, \mathbf{w}) + \mathbf{w}_\xi\mathbf{Q}^{\widehat{\boldsymbol{\pi}}_{\mathbf{w}}}(\mathbf{s}, \mathbf{a}, \mathbf{w}) - \mathbf{w}\mathbf{Q}^{\widehat{\boldsymbol{\pi}}_{\mathbf{w}}}(\mathbf{s}, \mathbf{a}, \mathbf{w})| \\ \leq&\max_{\mathbf{s},\mathbf{a}} |\mathbf{w}\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}) - \mathbf{w}_\xi\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w})| \\ &+ \max_{\mathbf{s},\mathbf{a}} |\mathbf{w}_\xi\mathbf{Q}^*(\mathbf{s}, \mathbf{a}, \mathbf{w}) - \mathbf{w}_\xi\mathbf{Q}^{\widehat{\boldsymbol{\pi}}_{\mathbf{w}}}(\mathbf{s}, \mathbf{a}, \mathbf{w})| \\ &+ \max_{\mathbf{s},\mathbf{a}} |\mathbf{w}_\xi\mathbf{Q}^{\widehat{\boldsymbol{\pi}}_{\mathbf{w}}}(\mathbf{s}, \mathbf{a}, \mathbf{w}) - \mathbf{w}\mathbf{Q}^{\widehat{\boldsymbol{\pi}}_{\mathbf{w}}}(\mathbf{s}, \mathbf{a}, \mathbf{w})| \\ \leq&\frac{2mR_{\max}}{1-\gamma}||\mathbf{w} - \mathbf{w}_\xi||_\infty + \gamma\sqrt{\frac{c}{(1-\gamma)^3}\frac{\log(c|\mathcal{S}||\mathcal{A}||C_\xi|/\delta)}{K}} \\ \leq&\frac{2mR_{\max}\xi}{1-\gamma} + \gamma\sqrt{\frac{c}{(1-\gamma)^3}\frac{\log(c|\mathcal{S}||\mathcal{A}|(\frac{2}{\xi})^m/\delta)}{K}} \\ =&\frac{2mR_{\max}\xi}{1-\gamma} + \gamma\sqrt{\frac{c'm}{(1-\gamma)^3}\frac{\log(c''|\mathcal{S}||\mathcal{A}|/(\delta\xi))}{K}},\end{aligned}$$

where the first inequality holds due to triangle inequality, the second inequality holds due to Equation (4) and (7), the third inequality holds due to the property of $\xi-$covering set.

Finally, uniformly splitting the error ϵ by setting $\xi = \frac{(1-\gamma)\epsilon}{4mR_{max}}$, and selecting N that

$$\gamma\sqrt{\frac{c'm}{(1-\gamma)^3}\frac{\log(c''|\mathcal{S}||\mathcal{A}|m/(\delta\xi))}{K}} \leq \frac{\epsilon}{2},$$

we can have $d\big(\mathbf{Q}^{\widehat{\boldsymbol{\pi}}_{\mathbf{w}}}, \mathbf{Q}^*\big) \leq \epsilon$ with probability at least $1 - \delta$. Solving out N ends our proof.

REFERENCES

[1] R. Ding, Z. Yang, Y. Wei, H. Jin, and X. Wang, "Multi-agent reinforcement learning for urban crowd sensing with for-hire vehicles," *INFOCOM*, 2021.

[2] C. Xiang, Y. Li, Y. Zhou, S. He, Y. Qu, Z. Li, L. Gong, and C. Chen, "A comparative approach to resurrecting the market of MOD vehicular crowdsensing," in *INFOCOM*, 2022.

[3] X. Tang, Z. T. Qin, F. Zhang, Z. Wang, Z. Xu, Y. Ma, H. Zhu, and J. Ye, "A deep value-network based approach for multi-driver order dispatching," *KDD*, 2019.

[4] D. M. Roijers, P. Vamplew, and S. Whiteson, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, 2013.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, and L. Kaiser, "Attention is all you need," *NIPS*, 2017.

[6] R. Yang, X. Sun, and K. Narasimhan, "A generalized algorithm for multi-objective reinforcement learning and policy adaptation," *NIPS*, 2019.

[7] A. Agarwal, S. M. Kakade, and L. F. Yang, "Model-based reinforcement learning with a generative model is minimax optimal," *COLT*, 2020.

[8] M. G. Azar, R. Munos, and B. Kappen, "On the sample complexity of reinforcement learning with a generative model," *ICML*, 2012.

[9] *https://gaia.didichuxing.com*.

[10] *https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page*.

[11] A. Abels, D. M. Roijers, T. Lenaerts, and A. Nowé, "Dynamic weights in multi-objective deep reinforcement learning," *ICML*, 2019.

[12] Z. Shi, G. Yang, X. Gong, S. He, and J. Chen, "Quality-aware incentive mechanisms under social influences in data crowdsourcing," *TON*, 2022.

[13] B. Li and J. Liu, "Achieving information freshness with selfish and rational users in mobile crowd-learning," *JSAC*, 2021.

[14] M. Karaliopoulos, I. Koutsopoulos, and L. Spiliopoulos, "Optimal user choice engineering in mobile crowdsensing with bounded rational users," *INFOCOM*, 2019.

[15] H. Jin, L. Su, H. Xiao, and K. Nahrstedt, "Incentive mechanism for privacy-aware data aggregation in mobile crowd sensing systems," *TON*, 2018.

[16] G. Fan, Y. Zhao, Z. Guo, H. Jin, X. Gan, and X. Wang, "Towards fine-grained spatio-temporal coverage for vehicular urban sensing systems," in *INFOCOM*, 2021.

[17] E. Wang, M. Zhang, Y. Xu, H. Xiong, and Y. Yang, "Spatiotemporal fracture data inference in sparse urban crowdsensing," in *INFOCOM*, 2022.

[18] X. Gong and N. B. Shroff, "Truthful mobile crowdsensing for strategic users with private data quality," *TON*, 2019.

[19] X. Li, K. Xie, X. Wang, G. Xie, D. Xie, Z. Li, J. Wen, Z. Diao, and T. Wang, "Quick and accurate false data detection in mobile crowd sensing," *TON*, 2020.

[20] M. Zhang, B. Swenson, J. Huang, and H. V. Poor, "Truthful mobile crowd sensing with interdependent valuations," *Mobihoc*, 2020.

[21] Z. Shi, S. Jiang, L. Zhang, Y. Du, and X.-Y. Li, "Crowdsourcing system for numerical tasks based on latent topic aware worker reliability," in *INFOCOM*, 2021.

[22] C. Huang, H. Yu, J. Huang, and R. A. Berry, "Strategic information revelation in crowdsourcing systems without verification," in *INFOCOM*, 2021.

[23] X. Pang, Z. Wang, D. Liu, J. C. S. Lui, Q. Wang, and J. Ren, "Towards personalized privacy-preserving truth discovery over crowdsourced data streams," *TON*, 2022.

[24] M. Zhang, L. Yang, S. He, M. Li, and J. Zhang, "Privacy-preserving data aggregation for mobile crowdsensing with externality: An auction approach," *TON*, 2021.

[25] H. Jiang, P. Zhao, and C. Wang, "Roblop: Towards robust privacy preserving against location dependent attacks in continuous LBS queries," *TON*, 2018.

[26] Y. Huang, Z. Xiao, D. Wang, H. Jiang, and D. Wu, "Exploring individual travel patterns across private car trajectory data," *TITS*, 2020.

[27] W. Liu, Y. Yang, E. Wang, H. Wang, Z. Wang, and J. Wu, "Dynamic online user recruitment with (non-) submodular utility in mobile crowdsensing," *TON*, 2021.

[28] Q. Hu, S. Wang, X. Cheng, J. Zhang, and W. Lv, "Cost-efficient mobile crowdsensing with spatial-temporal awareness," *TMC*, 2021.

[29] Z. Dai, C. H. Liu, Y. Ye, R. Han, Y. Yuan, G. Wang, and J. Tang, "Aoi-minimal UAV crowdsensing by model-based graph convolutional reinforcement learning," in *INFOCOM*, 2022.

[30] D. Zhao, M. Cao, L. Ding, Q. Han, Y. Xing, and H. Ma, "Dronesense: Leveraging drones for sustainable urban-scale sensing of open parking spaces," in *INFOCOM*, 2022.

[31] Z. Dai, H. Wang, C. H. Liu, R. Han, J. Tang, and G. Wang, "Mobile crowdsensing for data freshness: A deep reinforcement learning approach," in *INFOCOM*, 2021.

[32] C. H. Liu, Z. Dai, H. Yang, and J. Tang, "Multi-task-oriented vehicular crowdsensing: A deep learning approach," in *INFOCOM*, 2020.

[33] C. H. Liu, C. Piao, and J. Tang, "Energy-efficient UAV crowdsensing with multiple charging stations by deep learning," in *INFOCOM*, 2020.

[34] F. Yucel, M. Yuksel, and E. Bulut, "Qos-based budget constrained stable task assignment in mobile crowdsensing," *TMC*, 2021.

[35] C. Dai, X. Wang, K. Liu, D. Qi, W. Lin, and P. Zhou, "Stable task assignment for mobile crowdsensing with budget constraint," *TMC*, 2021.

[36] R. Zhou, Z. Li, and C. Wu, "A truthful online mechanism for location-aware tasks in mobile crowd sensing," *TMC*, 2018.

[37] Y. Sun, C. Lin, H. Dai, P. Wang, L. Wang, G. Wu, and Q. Zhang, "Trading off charging and sensing for stochastic events monitoring in wrsns," *TON*, 2022.

[38] G. Einziger, C. Chiasserini, and F. Malandrino, "Scheduling advertisement delivery in vehicular networks," *TMC*, 2018.

[39] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," *ICML*, 1993.

[40] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," *AAAI*, 2018.

[41] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. N. Foerster, and S. Whiteson, "QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning," *ICML*, 2018.

[42] K. Son, D. Kim, W. J. Kang, D. Hostallero, and Y. Yi, "QTRAN: learning to factorize with transformation for cooperative multi-agent reinforcement learning," *ICML*, 2019.