IEEE Transactions on Intelligent Transportation Systems, Volume 25, Issue 2, February 2024, pp. 1882-1896
DOI: 10.1109/TITS.2023.3317358

Privacy-Preserving Federated Deep Reinforcement Learning for Mobility-as-a-Service

Kai-Fung Chu, *Member, IEEE*, Weisi Guo, *Senior Member, IEEE*

***Abstract*—Mobility-as-a-service (MaaS) is a new transport model that combines multiple transport modes in a single platform. Dynamic passenger behavior based on past experiences requires reinforcement-based optimization of MaaS services. Deep reinforcement learning (DRL) may improve passenger satisfaction by offering the most appropriate transport services based on individual passenger experiences and preferences. However, this produces a new privacy risk to the MaaS platform using the centralized DRL method. Information leakage will occur if the platform is not carefully designed with privacy-preserving mechanisms. In this paper, we propose a federated deep deterministic policy gradient (FDDPG) that maximizes passenger satisfaction and MaaS long-term profit while preserving privacy. We enforce an equally weighted experience sampling mechanism to prevent sampling bias such that the solution quality of FDDPG is statistically equivalent to the centralized algorithm. During the model training and inference, information is processed locally, and only the gradients are shared, which prevents information leakage to any semi-honest participants and eavesdroppers. Secure aggregation protocol in line with the dynamic property of the mobile agent is also used in the gradient sharing step to ensure that the algorithm is prevented from inference attacks. We perform experiments on New York City-based real-world and synthetic scenarios. The results show that the proposed FDDPG can improve the MaaS profit and passenger satisfaction by about 90% and 15%, respectively, and maintain stable training against agent dropout. Our approach and findings could enhance MaaS utility as well as facilitate passenger trust and participation in MaaS and other data-driven transportation systems.**

***Index Terms*—Privacy-preserving machine learning, federated reinforcement learning, mobility-as-a-service, passenger behavior, secret sharing.**

I. Introduction

Mobility-as-a-service (MaaS) is a new mobility concept aiming to integrate different transport modes by providing a single platform for passengers. The passengers can access real-time traffic information, plan the journey, obtain bundle offers, book the transport service, etc. Besides addressing the routine transport demand, novel intelligent transportation systems applications can be integrated together with MaaS. For example, predicting future traffic information [1], enabling sharing economy by ride-sharing [2], and increasing utilization of idle vehicles by rebalancing [3]. As a promising component of intelligent transportation systems, MaaS may provide various benefits to the modern smart city, such as reducing traffic congestion, energy consumption, and air pollution [4]. All these features can be consolidated into a single mobile application. To realize the MaaS concept, a mature software system connecting the mobility providers and passengers and managing the traffic information is necessary. The software system should have an intelligent scheduler to handle and optimize passengers' journey queries and system resources. A successful software system and intelligent scheduler may significantly enhance the public acceptance of MaaS.

Passengers may query their origin and destination to obtain all the necessary traffic information, including the planned journey and offered price. In a complex transportation system with a tremendous amount of real-time data, the algorithm of an intelligent scheduler is an essential component to process their query and recommend the journey. Unlike standard path planning algorithms, the planned journey of MaaS may be personalized for each passenger based on their experiences and preferences. This may be achieved by formulating the problem as a multi-objective journey planning problem where each objective term indicates different utilities, including travel time, discomfort, price, and operation cost. Moreover, the dynamic passenger experiences process of each travel can be modeled by Markov decision process and solved by deep reinforcement learning (DRL) as presented in [5]. Journeys planned by multi-objective journey planning problem with the DRL generated weights are expected to be preferred by passengers. The MaaS with this algorithm may improve the passenger satisfaction with the journey and overall profit of MaaS. However, during the training and inference processes, passengers must provide their personal information and travel experience as input to the agent, which may cause information leakage issues at the MaaS central server and communication channel. This type of centralized system where the server holding all the information is vulnerable to privacy threats. Multiple adversaries such as developers, service administrators, and managers may perform various attacks on the system [6], resulting in information leakage from the centralized MaaS platform. Furthermore, learning-based system, as the one discussed, usually vulnerable to inference attacks such as membership inference attacks [7], model inversion attacks [8], and property inference attacks [9]. These attacks may obtain certain information from the system such as to determine whether a data belongs to the training dataset or not. As a system containing massive amount of passenger information, we should mitigate the security loopholes in the algorithmic

This work was supported by EPSRC MACRO - Mobility as a service: MAnaging Cybersecurity Risks across Consumers, Organisations and Sectors (EP/V039164/1)

Kai-Fung Chu is with the School of Aerospace, Transport and Manufacturing, Cranfield University, Milton Keynes, MK43 0AL, UK, and the Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, UK. (e-mail: kaifung.chu@cranfield.ac.uk; kfc35@cam.ac.uk)

Weisi Guo is with the School of Aerospace, Transport and Manufacturing, Cranfield University, Milton Keynes, MK43 0AL, UK. (e-mail: weisi.guo@cranfield.ac.uk)

level.

Motivated by the potential privacy risks [10] and research gap in the centralized MaaS platform, we explore the possible technologies for building a privacy-preserving MaaS. Among the technologies, federated learning is a distributed machine learning technique that allows the training to be performed in distributed client agents by their private and local dataset. Each data owner may train their client model and securely share the learned gradient to update the server model without invasion of data privacy. It could be integrated into deep deterministic policy gradient (DDPG) [11], where it trains a set of client agents in a distributed manner, and the trained models are securely aggregated to the server agent. Federated DDPG (FDDPG) is privacy-preserving since it restricts the interaction between the local environment and agents other than the environment owner [12]. Therefore, a federated MaaS platform could be a promising solution to address the aforementioned privacy issues.

In this paper, we aim to build a privacy-preserving MaaS by the federated architecture. To do this, we first integrate the multi-objective MaaS journey planning problem for passenger behavior and the Markov decision process-based passenger satisfaction problem into a bi-level problem. Then, we transform the centralized MaaS problem into a federated version so that each passenger is responsible for their own sub-problem. A FDDPG algorithm is presented to securely solve the problem without sharing sensitive information. To maintain the solution quality compared to centralized version, we enforce an equally weighted experience sampling mechanism. Also, we integrate a secure aggregation protocol to the FDDPG algorithm to ensure the information is secure from inference attacks. Detailed experiments on New York City-based real-world and synthetic datasets are conducted to study the solution quality of the FDDPG algorithm with secure aggregation protocol. The results show that the proposed FDDPG can improve the MaaS profit and passenger satisfaction by about 90% and 15%, respectively, while preserving privacy and maintaining stable training against agent dropout. Our approach and findings could enhance MaaS utility as well as facilitate passenger trust and participation in MaaS and other data-driven transportation systems. The current MaaS system faces several key challenges, which include:

1. Security threat from centralized server: The centralized server that stores passenger information poses a risk of potential information leakage, thereby compromising the security of the system.
2. Privacy concerns and inference attacks: The utilization of passenger information for journey planning introduces privacy vulnerabilities, such as inference attacks. These attacks involve extracting sensitive information from the model training process, thereby jeopardizing the privacy of the passengers.
3. Dropout issue in dynamic transportation systems: In highly dynamic transportation systems, passengers frequently move around, and their client devices may experience disconnections from the server. This poses a challenge during the training procedure, as dropouts can occur, leading to interruptions in the training process.

To address these challenges, the proposed approach offers appropriate solutions:

1. Adoption of distributed architecture: To enhance system security and robustness, the proposed approach transforms the MaaS system into a distributed architecture. This decentralization helps to avoid centralizing information and instructions, thereby mitigating security risks.
2. Implementation of federated learning: To address privacy concerns during model training, the proposed approach incorporates federated learning. This approach ensures that each client passenger is responsible for their own sub-problem, and only intermediate information is shared in order to prevent information leakage.
3. Resilience to dropouts: The model training algorithm in the proposed approach is designed to allow dropouts during any step of the training process. It can handle such disruptions without affecting the functionality of the algorithms, provided that the number of passengers exceeds a predefined threshold value.

The main contributions of this paper can be summarized as follows:

1. We formalize and integrate the multi-objective MaaS journey planning problem for passenger behavior and the Markov decision process-based passenger satisfaction problem into a bi-level problem.
2. Motivated by the information leakage issue in the formulated problem and classic DDPG method, we present an FDDPG solution approach that can achieve similar solution quality compared to the centralized algorithm while preserving passenger privacy. Since each client agent only explores their local space, an equally weighted experience sampling mechanism is enforced in the FDDPG algorithm to avoid bias in the experience sampling process.
3. Besides the information leakage issue, the method is vulnerable to inference attacks. Also, agent in MaaS is highly dynamic and they may dropout from the system during the algorithmic steps. Therefore, a secure aggregation protocol that allows dynamic dropout from the system is integrated into the FDDPG algorithm to further enhance the system security and stability.

The rest of this paper is organized as follows. Section II reviews the related work of MaaS, DRL approaches in transportation, and privacy-preserving technologies. Section III illustrates the problem formulation of privacy-preserving MaaS and the threat models. The privacy-preserving FDDPG algorithm for MaaS is presented in Section IV. Experiments and results are presented in Section V. Finally, Section VI concludes this paper.

II. Related Work

A. Multimodal Journey Planning Problems

MaaS contains multiple mobility providers for a set of passengers to be transiting over the multimodal transport network. Most of the existing methods of determining the optimal path consider the transport characteristics only. For example, reference [13] formulated a dynamic shortest path problem using historical and real-time information and solved the problem by

a clustering-based hybrid approximate dynamic programming algorithm that combines a deterministic lookahead policy and a value function approximation. However, prices are not the only factor affecting passenger route choices. Individual passenger may have their own preference related to the passenger profiles such as the reference [14] studied the correlations between loyalty and passengers' profiles, experience, and values for different transport services. The study indicates that those factors are influencing their choice of transport mode. Therefore, there are research works considering the passenger profile and preference to plan a customized journey for each individual passenger. Reference [15] proposed a design framework for multi-modal transportation systems integrating multiple mobility services that consider travelers' choices for describing travelers' behavior and estimating agencies' costs, so that the total cost of mobility providers and passengers can be minimized by the formulated optimal design problem. Therefore, a problem formulation considering the passenger experiences and preferences for maximizing their satisfaction and MaaS profit is desired.

B. Cybersecurity Threats of MaaS

MaaS faces various cybersecurity threats and attacks, and there is existing literature studying the vulnerable component of MaaS [16]. Reference [6] identified the potential threats of individual coordinators and markets of mobility providers. The study indicates that the information of the user profile can be stolen from a poorly managed system. Reference [17] built a privacy-preserving data publishing algorithm for trajectory data using differential privacy with a prefix tree structure, an incremental privacy budget allocation model, and a spatial-temporal dimensionality reduction model to improve the data utility and run-time efficiency. In [18], the authors aimed to eliminate the need for a central MaaS coordinator and proposed a blockchain-based MaaS that improves the trust and transparency of all stakeholders in a decentralized way. However, there is no existing literature focusing on the information leakage issue and the potential threat of inference attacks for the journal planning problem of MaaS.

C. Privacy-preserving Machine Learning

A large and growing body of literature has studied privacy-preserving machine learning. The long pipeline of general machine learning has provided lots of potential threats in the process-chain in the system, including data preparation, model training and evaluation, model deployment, and model inference [19]. Most of the privacy issues are about data and model privacy. One of the architecture-based approaches to preserve data and model privacy is federated learning (FL) [20], [21]. FL was proposed to enable joint training of a machine learning model using the client data samples in each client without revealing the data to others [22]. FL can be classified as vertical and horizontal FL depending on whether the feature or sample spaces of the datasets across the clients are different. For vertical FL, the sample spaces of datasets across the clients are the same but differ in feature space. For horizontal FL, the feature spaces are the same but not the sample spaces, which is similar to the problem case in this paper. To preserve data privacy, as proposed in [23], each client trains a client machine learning model using the client data sample and uploads the trained models or gradients to the server. The server then updates the server model by calculating the average of the models or gradients and broadcasts the updated server model to the clients in each iteration. Using this method, the server model is seamlessly trained by the data samples in the client, while the data is stored locally without sharing it with the server or other clients. On the other hand, the shared model may be vulnerable to attacks focused on inferring private information from it, such as membership inference attacks [7], model inversion attacks [8], and property inference attacks [9]. To preserve model privacy from these inference attacks, secure aggregation can be used to ensure the server only gets the aggregated information but not the information from an individual client. The authors in [24] proposed a practical secure aggregation method for federated learning. The method leverages secret sharing in secure multiparty computation to compute sums of model parameter updates from the individual client. Compared to existing methods, it can still be functioning if clients dropout, and it avoids transmitting lots of data.

D. Federated Reinforcement Learning

Federated learning can be integrated into reinforcement learning to form federated reinforcement learning (FRL). Many recent research of federated reinforcement learning have been conducted in various engineering domains. Reference [25] use an FRL architecture and a knowledge fusion algorithm for navigation in cloud robotic systems, which effectively use prior knowledge and quickly adapt to new environments. In smart grid research, the authors in [26] enhance the scalability and privacy by transforming the voltage control problem into a decentralized problem in FRL architecture. The secure communication and data resource allocation problem in cyber-physical systems was formulated in [27] as FRL architecture to address the non-stationary issue caused by the heterogeneity of cyber-physical systems devices. For unmanned aerial vehicles, the FRL algorithm was proposed in a semi-distributed framework to solve the power minimization problem in the multi-access edge computing system [28]. FRL was also introduced into vehicular communication to speed up the training process across vehicles [29]. However, no one has developed privacy-preserving federated reinforcement learning specifically for the operation of MaaS that addresses privacy issues, which is the research gap that we aim to bridge in this paper.

III. System Model

In this section, we introduce the threat models first. Then, we present the multi-objective journey planning problem and passenger satisfaction problem in MaaS. Table I summarizes the notation used in this section.

A. Threat Model

We assume all the participants, including the MaaS coordinator, mobility providers, and passengers, are semi-honest,

TABLE I
NOTATION SUMMARY.

Notations	Meaning
$G(\mathcal{N}, \mathcal{A})$	Graph of the transport network
$\mathcal{F}$	Set of mobility (service) providers
f	Mobility (service) provider $f \in \mathcal{F}$
$\mathcal{N}$	Set of nodes in the network
$\mathcal{A}$	Set of links in the network
$\mathcal{A}_f$	Subset of $\mathcal{A}$ operated by $f \in \mathcal{F}$
$\mathcal{N}^+(i)$	Set of incoming locations of i
$\mathcal{N}^-(i)$	Set of outgoing locations of i
$\mathcal{K}$	Set of passengers
k	Passenger k
o^k	Origin node of passenger $k \in \mathcal{K}$
d^k	Destination node of passenger $k \in \mathcal{K}$
β_{ij}^f	Travel time cost of link $(i,j) \in \mathcal{A}_f$
δ_{ij}^f	Discomfort index of link $(i,j) \in \mathcal{A}_f$
ρ_{ij}^f	Price of link $(i,j) \in \mathcal{A}_f$
μ_{ij}^f	Operating cost of link $(i,j) \in \mathcal{A}_f$
W^k	Utility weights of passenger k
w_β^k	Weight of β_{ij}^f
w_δ^k	Weight of δ_{ij}^f
w_ρ^k	Weight of ρ_{ij}^f
w_μ^f	Weight of μ_{ij}^f
C_{ij}^f	Capacity of link $(i,j) \in \mathcal{A}_f$
x_{ij}^k	Binary variable for link $(i,j) \in \tilde{\mathcal{A}}_s$ that recommends to passenger $k \in \mathcal{K}$
y_{ij}^f	Binary variable for a coordinator to serve a link $(i,j) \in \mathcal{A}_f$
θ^π	Parameters of actor local
θ^Q	Parameters of critic local
$\theta^{\pi'}$	Parameters of actor target
$\theta^{Q'}$	Parameters of critic target

which is a commonly defined adversary model in privacy-preserving computation [30]. Some literature also refer it as honest-but-curious [31]. In this model, the participants are honest in following the protocol merely, but they are curious about extracting any data and information. For example, in our case, the participants will not perform a data poisoning attack [32] to cheat the system, but they may infer information using membership inference attacks [7] with the obtained model from an individual passenger. Therefore, any unpremeditated information leakage can be prevented by achieving this level of security.

B. Problem Formulation

In a MaaS, there are two main components. The first component is the mobility coordinator, which is responsible for coordinating resources and planning the journey. The coordinator plays a critical role in arranging various transportation services to ensure efficient mobility for passengers. There exist multiple mobility providers that offer diverse transport services, including those provided by bus companies and railway operators. These mobility providers function as service providers, handling the transportation needs of passengers within the MaaS ecosystem. Together, these components work together to facilitate seamless mobility solutions and enhance the overall MaaS experience. We consider the scenario in MaaS where passengers query their origin and destination to the MaaS coordinator for obtaining an optimal planned journey based on their individual behaviors and characteristics. We will introduce the multi-objective journey planning problem and passenger satisfaction problem. Then, we will present the integrated bi-level problem.

1) Multi-objective Journey Planning Problem:

Consider a MaaS transport network modeled by a directed graph $G(\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ and $\mathcal{A}$ is the set of nodes and links in the network, respectively. One can interpret the node and link as the station and road, respectively, for a bus service for example. We denote the set of mobility providers by $\mathcal{F}$ and each mobility provider $f \in \mathcal{F}$ provides transport services over their transport network $\mathcal{A}_f \in \mathcal{A}$. A transport service from i to j is provided on link (i,j) and the time cost, discomfort index, price, and operation cost of each service provided by f are represented by β_{ij}^f, δ_{ij}^f, ρ_{ij}^f, and μ_{ij}^f, respectively. Based on the service utility cost, passenger's origin o^k and d^k, and passenger behavior, the MaaS scheduler computes an optimal journey to be offered to the passenger. Noted that the journey may be a combination of services from more than one mobility provider in MaaS. The optimization model MaaS problem can be referenced in the literature [33].

Two binary decision variables, x_{ij}^{kf} and y_{ij}^f, are defined in the formulation of the problem. x_{ij}^{kf} are used to represent the mobility service to be offered to the passenger:

$$x_{ij}^{kf} = \begin{cases} 1 & \text{if link } (i,j) \text{operated by } f \text{ offers to } k, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

y_{ij}^f represent the transport service operation status:

$$y_{ij}^f = \begin{cases} 1 & \text{if link } (i,j) \text{ is operated by } f, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The objective of the problem is to determine the optimal x_{ij}^{kf} and y_{ij}^f such that the total utility cost of passengers and mobility providers are minimized:

$$\sum_{(i,j)\in\mathcal{A}_f, k\in\mathcal{K}, f\in\mathcal{F}} (w_\beta^k \beta_{ij}^f + w_\delta^k \delta_{ij}^f + w_\rho^k \rho_{ij}^f) x_{ij}^{kf} + w_\mu^f \mu_{ij}^f y_{ij}^f, \tag{3}$$

where β_{ij}^f, δ_{ij}^f, ρ_{ij}^f, and μ_{ij}^f are the travel time, discomfort index, price, and operating cost of link $(i,j) \in \mathcal{A}_f$, respectively, and w_β^k, w_δ^k, w_ρ^k, w_μ^f are the weighting of the corresponding utility terms.

Let $\mathcal{N}^-(i)$ and $\mathcal{N}^+(i)$ be the sets of outgoing and incoming locations of i, respectively, such that $\mathcal{N}^-(i) = \{j \in \mathcal{N} | (i,j) \in \mathcal{A}\}$ and $\mathcal{N}^+(i) = \{j \in \mathcal{N} | (j,i) \in \mathcal{A}\}$. To ensure the flow conservation in the transport network, the following equation requires to be imposed in the problem.

$$\sum_{j\in\mathcal{N}^-(i)} x_{ij}^{kf} - \sum_{j\in\mathcal{N}^+(i)} x_{ji}^{kf} = \begin{cases} 1 & \text{if } i = o^k, \\ -1 & \text{if } i = d^k, \\ 0 & \text{otherwise,} \end{cases} \quad \forall i \in \mathcal{N}, k \in \mathcal{K}, f \in \mathcal{F}, \tag{4}$$

TABLE II
SAMPLE RELATIONSHIP BETWEEN SATISFACTION LEVEL AND ADMISSION RATE.

Satisfaction level, H^k	**Admission rate,** $P(H^k)$
5	1
4	0.75
3	0.5
2	0.25
1	0

Since the capacity of the transport service is limited, an equation is included to restrict the total number of passengers in the transport service:

$$\sum_{k\in\mathcal{K}} x_{ij}^{kf} \leq C_{ij}^f y_{ij}^f, \quad \forall (i,j) \in \mathcal{A}_f, f \in \mathcal{F} \tag{5}$$

where C_{ij}^f is the capacity of transport service from i to j that operated by $f \in \mathcal{F}$.

As a whole, the multi-objective journey planning problem is given as follows:

Problem 1 (Multi-objective Journey Planning Problem):

$$\begin{aligned} \min_{x_{ij}^{kf}, y_{ij}^f} \quad & (3) \\ \text{s.t.} \quad & (4)\text{–}(5). \end{aligned}$$

To solve this problem which is formulated as an integer linear program, a standard solver can be used given that the weight of the objectives, $W^k = [w_\beta^k; w_\delta^k; w_\rho^k; w_\mu^f]$, are set. However, the utility weight W^k that represents the preference for different objectives is hard to be defined, and it is the difference for each passenger who has different behaviors, experiences, and preferences. Therefore, another problem, namely, the passenger satisfaction problem, for determining the utility weight is formulated.

2) Passenger Satisfaction Problem:

The passenger satisfaction problem aims to determine the utility weight based on passenger characteristics so that the passenger satisfaction and admission rate to the transport service is increased, thus increasing the MaaS profit.

We model the passenger admission process to the MaaS as a 4-tuple Markov decision process $\langle \mathcal{S}, \mathcal{A}, P, R \rangle$, where $\mathcal{S}$, $\mathcal{A}$, P, and R are the set of states and actions, state transition and reward function, respectively. The state $s_t^k \in \mathcal{S}$ is the concatenation of passenger satisfaction and the passenger characteristics. The action $a_t^k \in \mathcal{A}$ is the utility weight introduced in Problem 1, i.e., $a_t^k := \left[w_\beta^k; w_\delta^k; w_\rho^k; w_\mu^f\right]$ for time t and passenger k. $P(s_{t+1}^k | s_t^k, a_t^k)$ represents the satisfaction variation from states $s_t^k \in \mathcal{S}$ to $s_{t+1}^k \in \mathcal{S}$ for acting action $a_t^k \in \mathcal{A}$. $R(s_t^k, a_t^k, s_{t+1}^k)$ represents the total profit due to the transition from s_t^k to s_{t+1}^k after acting a_t^k.

The passenger satisfaction level in the state can be represented by a N-level integer value, which also indicates the admission rate proportionally. A sample relationship between the satisfaction with $N = 5$ and admission rate is shown in Table II.

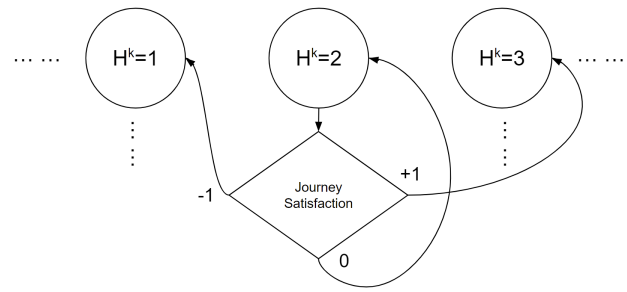


Fig. 1. Satisfaction transition representation.

Let H^k be the satisfaction level of passenger k. It is a variable depending on the transport service offered by the MaaS. In general, the satisfaction increases if the passenger is satisfied with the offered journey and decreases otherwise. The passenger satisfaction change is a function of the expectation difference on each utility:

$$H^k := \begin{cases} H^k + n & \text{if } E^k \geq \overline{E}^k, \\ H^k - n & \text{if } E^k \leq \underline{E}^k, \\ H^k & \text{otherwise}, \end{cases} \quad \forall k \in \mathcal{K}, \tag{6}$$

where $\overline{E}^k$ and $\underline{E}^k$ is the upper and lower expectation difference threshold, n is the step size of the satisfaction level, and the expectation difference is defined as

$$\begin{aligned} E^k = \tilde{w}_\beta^k(\tilde{\beta}_{o^k d^k}^k - \beta_{o^k d^k}^k) + \tilde{w}_\delta^k(\tilde{\delta}_{o^k d^k}^k - \delta_{o^k d^k}^k) + \\ \tilde{w}_\rho^k(\tilde{\rho}_{o^k d^k}^k - \rho_{o^k d^k}^k), \quad \forall k \in \mathcal{K}. \end{aligned} \tag{7}$$

where $\tilde{W}^k = [\tilde{w}_\beta^k; \tilde{w}_\delta^k; \tilde{w}_\rho^k]$ is the utility weight of passenger k, $\tilde{\beta}_{o^k d^k}^k, \tilde{\delta}_{o^k d^k}^k, \tilde{\rho}_{o^k d^k}^k$ are the utility expected implicitly by the passenger, and o^k and d^k are the origin and destination of passenger k, respectively. Expected utility is the utility of an ideal best journey for the passenger. It can be determined by solving the planning problem as if the passenger is traveling alone in the system. The actual utility is the utility of the journey planned by MaaS, which may be deviated from the expected utility with an incompetent planner and limited capacity. Therefore, an incompetent planner may have a negatively large expectation difference on average.

A sample state transition representation of $n = 1$ is given in Fig. 1. The passenger experience of each journey may lead to satisfaction increases, decreases or unchanged, which affect the status and admission of the next journey. Similar consumer satisfaction models can be found in other field of studies such as supply chain [34] and product management [35].

Therefore, the passenger satisfaction problem can be formulated as:

Problem 2 (Passenger Satisfaction Problem):

$$\begin{aligned} \max_{a_t^k} \quad & \sum_{k,t} R(s_t^k, a_t^k, s_{t+1}^k) \\ \text{s.t.} \quad & (6)\text{–}(7). \end{aligned}$$

With the Markov decision process model, this problem can be solved by a reinforcement learning-based method that learns the transition and optimal action for the passengers.

3) Bi-level problem:

Problems 1 and 2 are connected via the utility weights since the actions in Problem 2 are defined as utility weights, i.e., $a_t^k := \left[w_\beta^k; w_\delta^k; w_\rho^k; w_\mu^f\right]$. We can integrate the problems into a bi-level problem, where problems 1 and 2 is the upper-level and lower-level optimization task, respectively. As a whole, the bi-level problem is formulated as:

Problem 3 (Bi-level Multi-objective Journal Planning and Passenger Satisfaction Problem):

$$\begin{aligned} \max_{x_{ij}^{kf}, y_{ij}^f} \quad & (3) \\ \text{s.t.} \quad & (4)\text{–}(5), \\ & a_t^k \in \arg\max\{\sum_{k,t} R(s_t^k, a_t^k, s_{t+1}^k) : (6)\text{–}(7)\}. \end{aligned}$$

We can determine the action (utility weights) first by solving Problem 2 with a reinforcement learning-based method, and then solve Problem 1 with integer optimization method.

IV. Privacy-Preserving Federated Deep Deterministic Policy Gradient

In this section, we present the privacy-preserving FDDPG algorithm and the components of the system.

A. System Components

1) Environment: The environment represents the state transition of the set of passengers and the MaaS platform. Given the journey x_{ij}^{kf} offered by the MaaS coordinator, the environment proceeds to a new state s_{t+1}^k and returns a reward r_t^k to the client agent at each time t, which is also represented by transition probability $P(s_{t+1}|s_t, a_t)$.

2) State: The state s_t represents the static and dynamic information of the passenger, including the passenger's characteristics and satisfaction level, respectively. The characteristics are composed of the passenger's personal information, such as income and age. The satisfaction level of the transport service is also sensitive. Therefore, we consider the state as the major private information that must stay in the passenger's property, such as mobile applications, and cannot be transmitted to other applications.

3) Action: The action a_t^k represents the utility weight $\left[w_\beta^k; w_\delta^k; w_\rho^k; w_\mu^f\right]$ which indicates the weight of the terms of objective function: time, discomfort, price, and operation cost in Eq. (3). The weight affects the optimal journey as defined in Problem 2.

4) Reward: The reward r_t^k represents the profit of the mobility service of passenger k at time t, which is equivalent to the price minus the operating cost, i.e,

$$r_t^k = \sum_{i,j,k,f} (\rho_{ij}^f - \mu_{ij}^f) x_{ij}^{kf}. \tag{8}$$

The MaaS coordinator aims to maximize profit by offering journeys that maximize passenger satisfaction and admission rate.

5) Agent: The agent represents the component for determining the action based on a given state. The agent aims to maximize the reward with an optimal action. Since the action is the utility weight, different actions affect the offered journey and thus the passenger satisfaction. Hence, the agent should learn to output utility weights that can increase the long-term reward. There are two types of networks in an agent: actor and critic. The actor network is used to output action based on the input state. A critic network is used to evaluate the performance of the state and action pairs.

6) Actor and Critic Networks: Neural networks are used to learn and perform the transition functions of states, actions, and rewards. To preserve the generality of the proposed approach, we use deep fully-connected neural networks as the neural network structure in this paper. The reason is that the input and output of the neural network are numeric features and values where deep fully-connected neural networks might be able to model the relationship already, unlike other data types such as images and time series data that usually use convolutional neural networks and recurrent neural network, respectively. The deep neural networks have to be designed to match the problem nature. For example, the action is an array ranging from 0 to 1 and thus we use the $sigmoid$ as the activation function.

B. Federated Deep Deterministic Policy Gradient

As discussed in Section III, the FDDPG algorithm is mainly used to solve Problem 2, which is equivalent to the lower-level optimization task of Problem 3. The privacy-preserving MaaS focuses on preventing personal information leakage during the model training and inference. The MaaS scheduler is responsible for solving the Problem 2. For the centralized architecture, the MaaS scheduler would require the passenger to submit their characteristics and satisfaction level as the state $s_t^k \in \mathcal{S}$ for the computation. This information may leak to semi-honest participants in MaaS or hackers who perform eavesdropping attacks. Hence, we present the privacy-preserving FDDPG algorithm to avoid information leakage under the semi-honest threat model assumption.

To avoid information leakage, the architecture is designed to be a federated architecture, as shown in Fig. 2 where the information and experience are stored locally within the passenger's device during the model training. An example sequence of the information flow is as follows:

1. The client agents transmit the weights to the MaaS coordinator based on the current models.
2. The MaaS coordinator plans the journey and offers it to the passengers.
3. Passengers decide whether to accept the journey offered, which obtains the experience, new state, and rewards.
4. The experience is transmitted to the client agent for storing in the replay buffer and performing model updates.
5. The client agents transmit the updates to the server.
6. The MaaS coordinator broadcasts the server models to clients.

The FDDPG algorithm is based on the deep deterministic policy gradient [11] which is a model-free off-policy algorithm

for determining the continuous utility weight. In the federated architecture, each passenger owns a client agent responsible for the individual utility weight. The agents in client and server have the same structure as presented in Section IV-A. The conventional terminology of "server" and "client" is adopted in this paper to denote the hierarchical difference between the involved parties. The designation of "server" can be associated with the MaaS coordinator, while the term "client" can be associated with the passenger.

In each iteration, a different set of client agents may participate in the model inference and model update. As any passenger may go offline at any moment, we do not restrict the participation of clients. However, the server will sample the experience based on the buffer indices on behalf of clients to ensure unbiased sampling. The algorithm is designed to be robust on random dropouts, given that there are at least K_0 clients throughout the algorithm. The algorithm aborts when the number of participating clients is smaller than K_0. To simplify the discussion and presentation, we assume at least K_0 clients participating in each iteration.

Each agent maintains an experience replay buffer ER^k that stores the transition tuple $(s_t^k, a_t^k, r_t^k, s_{t+1}^k)$ for updating the actors and critics. The reason for using an experience replay buffer to store transition for mini-batch sampling is to ensure the samples are independently and identically distributed. If the transitions are used to update the models immediately after being recorded, it would be sequential transitions, and thus the training is unstable. Noted that the mini-batch refers to the number of training samples that are processed together in each iteration of the algorithm. Having a mini-batch rather than training the sample one by one could speed up the training and stabilize the convergence in general. In our federated case, the training samples are sampled from individual replay buffer, which belongs to the same passenger. If it is in the centralized case, the replay buffer is mixed with training samples from different passengers, and thus the mini-batch could belong to different passengers.

We use the epsilon-greedy action selection method instead of adding Ornstein-Uhlenbeck (OU) noise for exploration. The reason is that the action space in our problem is normalized to between 0 and 1, so the OU noise may not guarantee that the actions are within the range. Uniformly random actions are selected by the probability ϵ. ϵ gradually decreases from the beginning episode with a decay factor $\bar{\epsilon}$ smaller than one.

The following update rules of the four models, namely, actor local, actor target, critic local, and critic target network, are applied to update the actor and critic based on a mini-batch sample. The parameters of critic local θ^Q are updated based on the loss, L:

$$L = \frac{1}{B}\sum_i (r_i + \gamma Q'(s_{i+1}, \pi'(s_{i+1}|\theta^{\pi'})|\theta^{Q'}) - Q(s_i, a_i|\theta^Q))^2 \tag{9}$$

where B is the mini-batch size, i is the index of sample in the mini-batch, and γ is the discount factor. The parameters of actor local θ^π are updated by computing the policy gradient:

$$\nabla_{\theta\pi} J \approx \frac{1}{B}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\pi(s_i)} \nabla_{\theta\pi}\pi(s|\theta^\pi)|_{s_i}. \tag{10}$$

Critic target $\theta^{Q'}$ and actor target $\theta^{\pi'}$ are updated softly by assigning the parameters from the corresponding local networks with a factor of τ:

$$\theta^{Q'} := \tau\theta^Q + (1-\tau)\theta^{Q'}, \tag{11}$$

and

$$\theta^{\pi'} := \tau\theta^\pi + (1-\tau)\theta^{\pi'}. \tag{12}$$

Each participating client computes the gradients of actor and critic based on the update rules in each iteration. The mini-batch sample of each participating client is randomly selected by the server on behalf of the clients to enhance unbiased sampling. Each client selects samples according to the random indices instructed by the server and computes gradients from their client experience replay buffer only to preserve privacy. To further ensure no information leakage from the gradient, the gradients are aggregated in the server using the secure aggregation protocol to be present in Section IV-C which guarantees the server cannot retrieve the gradient of an individual client except the aggregated gradient. The aggregated gradient is then used to update the actor and critic of the agent in the server. Notice that in some federated learning algorithms, the server aggregates the model parameters instead of the gradient. However, aggregating the model parameters is not applicable to our problem as we use Adam [36] as the optimizer, which contains a momentum term during the model update. Aggregating the model parameters updated by Adam using client experience makes the training unstable.

The detailed algorithm is shown in Algorithm 1. Lines 1 – 6 are parameter initialization. In line 9, the server samples a set of passengers. From lines 10 – 17, each passenger (client) decides their corresponding action. After executing the actions in line 18, the passengers store the experience in their replay buffer in lines 19 – 21, which occurs on the client side. Lines 22 – 24 occur on the server side to ensure that the sampling is randomly sampled. Based on the sampling instruction, the clients sample and compute the gradients in lines 25 – 28. Line 30 is the secure aggregation, which is shown in Algorithm 2. Finally, the server models are updated accordingly on the server side in lines 31 – 34 and broadcast the updated models to clients in line 35. There are two major communications between server and client in Algorithm 1, which are the model broadcast from the server to clients in lines 6 and 35, and the secure aggregation for the gradients from clients to the server in general if the key and mask communications in Algorithm 2 are excluded for simplicity.

C. Secure Aggregation

Although the states are not shared during the model training due to the federation, semi-honest users can still perform inference attacks to the shared gradient to get information about the training data. Moreover, agents in MaaS are highly dynamic and they may disconnected from the server at any

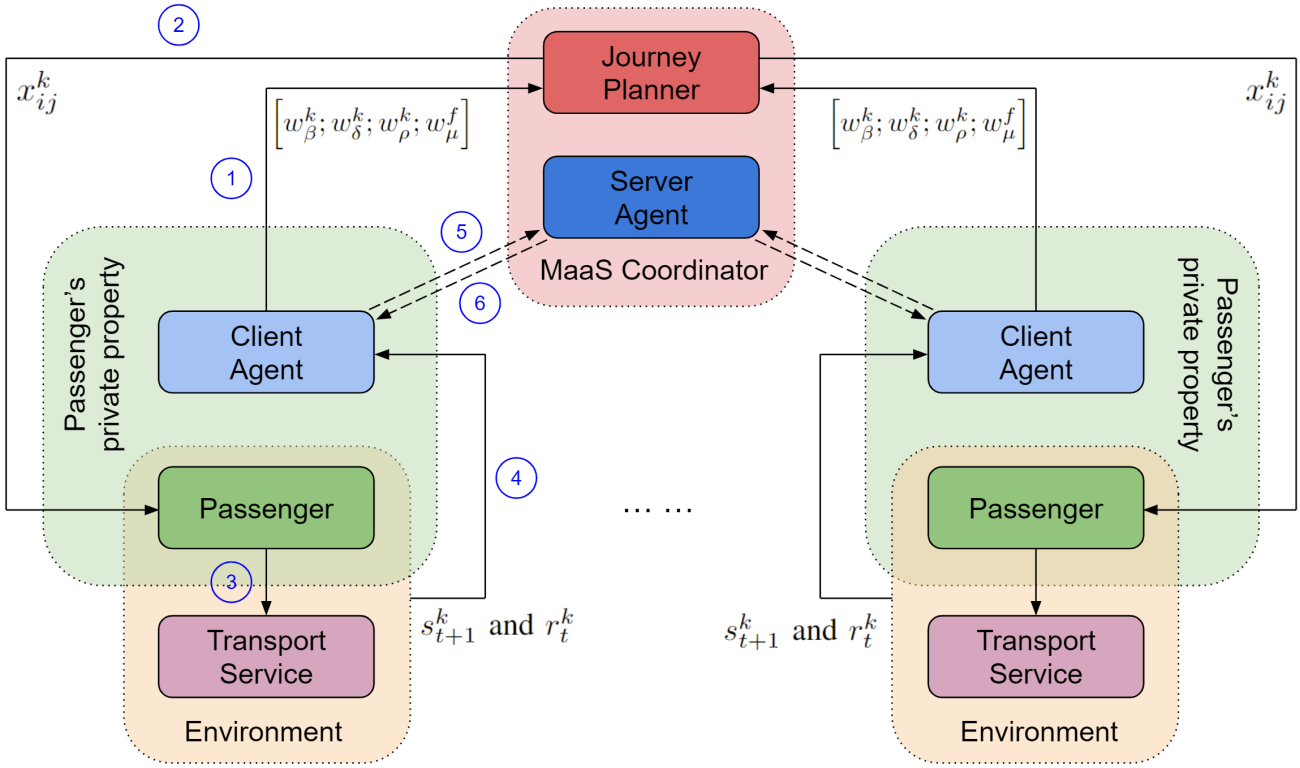


Fig. 2. Interaction between the environment, passengers and the MaaS coordinator.

moment. However, FDDPG requires contribution of gradient for training. Therefore, the system must be robust to the random dropout. A secure aggregation protocol that allow dynamic dropout from the system is required for gradient sharing to prevent information leakage and instability. The secure aggregation [24] used in this paper is based on secret sharing [37] and Diffie-Hellman (DH) key agreement [38]. The core of this protocol is to enable learning from aggregated input rather than individual input. The protocol involves five rounds of operations between the server and the clients:

- First, each client generates DH key pairs (c_{PK}^k, c_{SK}^k) and (s_{PK}^k, s_{SK}^k), and a signature ω^k based on a signing key d_{SK}^k and the public keys c_{PK}^k and s_{PK}^k. The keys are sent to the server as an acknowledgment of participation. After the server collected at least K_0 messages from the $|\mathcal{K}_t^{up}|$ participating clients, the server broadcasts the keys to the clients $k \in \mathcal{K}_t^{up}$. This round is mainly for key generation and collection. Only keys are shared in this round.
- Second, the clients validate the signatures and generate and send encrypted secret shares of a random element b^k and s_{SK}^k to the server. Specifically, the b^k and s_{SK}^k is for active and dropped client for later secret reconstruction.
- Third, the server broadcasts the secret shares to the clients. The clients use the secret shares to compute a masked input y^k and send it to the server. Since the masked input y^k is computed based on the secret shares, the aggregated masked input should be equal to the original aggregated input as if without the mask. In other words, the masks will be canceled out with each other. Therefore, the information is secure with the mask.
- Fourth, the clients check if the client list contains at least K_0 clients and return a signature ω_k'. This is to ensure that the secret reconstruction process can be carried out for the k-out-of-n redundant condition.
- Finally, the server broadcasts the signature to the clients for validation. The clients send a list of shares consisting of b^k for active client k or s_{SK}^k for dropped client k to the server for secret reconstruction. This round is for handling the dropped clients. Without the masks of dropped clients, the aggregated masked input cannot be reconstructed to the original aggregated input since the masks are not canceling out each other. Therefore, for dropped clients, the server will recover the key s_{SK}^k for dropped client k so that the masks are computed and included in the aggregation to cancel out the masks of other active clients.

During the process, the server maintains at least K_0 active clients. Otherwise, the process will be aborted. The aggregation can still proceed when clients dropout as the masks of that clients can be reconstructed by the secret shares in other clients. As in our problem, the aggregation of gradients is not a simple summation but a weighted sum with the number of transitions. Therefore, each client is required to multiply the gradient with the number of transitions and send it to the server for aggregation. The server will then divide the aggregated gradient by the total number of transitions for the model parameters update. A high-level pseudo-protocol of the secure aggregation is given in Algorithm 2. This protocol can be integrated into line 30 of Algorithm 1 for the set of

Algorithm 1 Privacy-Preserving Federated Deep Deterministic Policy Gradient

Initialize server actor local $\pi(s|\theta^{\pi})$ and critic local networks $Q(s,a|\theta^{Q})$ with parameters θ^{π} and θ^{Q}
Initialize parameters of server actor target $\pi'(s|\theta^{\pi'})$ and critic target networks $Q'(s,a|\theta^{Q'})$ with parameters $\theta^{\pi'} \leftarrow \theta^{\pi}$ and $\theta^{Q'} \leftarrow \theta^{Q}$
for $k = 1$ to $|\mathcal{K}|$ **do**
    Initialize passenger k parameters
end for
Broadcast θ^{π}, θ^{Q}, $\theta^{\pi'}$, and $\theta^{Q'}$ to clients
for episode $= 1$ to M **do**
    for iteration $t = 1$ to T **do**
        Admit a set of participating passengers $\mathcal{K}_t^{in} \subseteq \mathcal{K}$ participating model inference
        for passenger $k \in \mathcal{K}_t^{in}$ **do**
            $j^k \leftarrow$ random number between 0 and 1
            if $j^k < \epsilon$ **then**
                $a_t^k \leftarrow$ random vector between 0 to 1
            else
                $a_t^k \leftarrow \pi(s_t^k|\theta^{\pi})$
            end if
        end for
        Execute action $a_t = [a_t^1, \ldots, a_t^{|\mathcal{K}_t^{in}|}]$ to obtain new state $s_{t+1} = [s_{t+1}^1, \ldots, s_{t+1}^{|\mathcal{K}_t^{in}|}]$ and reward r_t
        for passenger $k \in \mathcal{K}_t^{in}$ **do**
            Store $(s_{i:i+N}^k, a_{i:i+N}^k, r_{i:i+N}^k, s_{i:i+N+1}^k)$ to replay buffer ER^k
        end for
        Admit a set of passengers $\mathcal{K}_t^{up} \subseteq \mathcal{K}$ participating model update
        if transitions in $\bigcup_{k \in \mathcal{K}_t^{up}} ER^k \geq$ mini-batch size B **then**
            Random select a mini-batch transitions with size B from $\bigcup_{k \in \mathcal{K}_t^{up}} ER^k$
            for passenger $k \in \mathcal{K}_t^{up}$ **do**
                Compute critic gradient based on Eq. (9) for transitions in ER^k
                Compute actor gradient based on Eq. (10) for transitions in ER^k
            end for
        end if
        Perform secure aggregation for the gradients according to Algorithm 2
        Update critic local based on the aggregated critic gradient
        Update actor local based on the aggregated actor gradient
        Update critic target based on Eq. (11)
        Update actor target based on Eq. (12)
        Broadcast θ^{π}, θ^{Q}, $\theta^{\pi'}$, and $\theta^{Q'}$ to clients
    end for
    $\epsilon := \epsilon\bar{\epsilon}$
end for

passengers $\mathcal{K}_t^{up}$.

Algorithm 2 Secure Aggregation Protocol

Round 0:
Clients:
Generate DH key pairs (c_{PK}^k, c_{SK}^k) and (s_{PK}^k, s_{SK}^k), and signature ω^k
Send signed public keys $c_{PK}^k, s_{PK}^k, \omega^k$ to server
Server:
Broadcast received keys to all clients
Round 1:
Clients:
Validate signatures
Generate secret shares of b^k and s_{SK}^k to server
Send secret shares of b^k and s_{SK}^k to server
Round 2:
Server:
Broadcast secret shares to client
Clients:
Compute masked input $\tilde{x}^k$
Send $\tilde{x}^k$ to server
Round 3:
Server:
Broadcast the list of active clients
Clients:
Check if the list contains at least K_0 clients
Send signature ω_k' to server
Round 4:
Server:
Broadcast signature to clients
Clients:
Send the list of shares consisting b^k for active client k or s_{SK}^k for dropped client k to the server
Server:
Reconstruct secrets

D. Discussion and Analysis

1) Equivalent solution quality:

Proposition 1: Consider the bi-level multi-objective journal planning and passenger satisfaction problem of the MaaS platform (Problem 3). The quality of the agent trained by Algorithm 1 is statistically equal to that of the agent trained by centralized training, given that the set of passengers is the same.

Proof 1: The actors in both federated and centralized algorithms are updated based on the policy gradient $\nabla_{\theta^{\pi}} J$ defined in Eq. (10) except the policy gradient in Algorithm 1 combines the individual policy gradient of client agents while the policy gradient in centralized algorithm is obtained from the mini-batch samples of the experience replay buffer. Since we ensure that the distribution of samples in Algorithm 1 is the same as that of the centralized algorithm, i.e. $D_{KL}(P(\bigcup_k ER^k)|P(ER)) = 0$ where $P(\bigcup_k ER^k)$ and $P(ER)$ is the probability distribution of the union of client

experience replay buffer and server buffer, respectively. One has

$$\begin{aligned}&\frac{1}{B}\sum_k B^k \nabla_{\theta\pi} J^k\\ &\approx \frac{1}{B}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_i,a=\pi(s_i)} \nabla_{\theta\pi}\pi(s|\theta^\pi)|_{s_i}\\ &\approx \nabla_{\theta\pi} J,\end{aligned}$$

where B^k is the sampled size of k. Therefore, the actor parameters have the same distribution if the initialization processes are the same.

2) Convergence:

Proposition 2: Consider the bi-level multi-objective journal planning and passenger satisfaction problem of the MaaS platform (Problem 3). The policy obtained in Algorithm 1 will be converged, given that the critic and actor networks are updated based on Eqs. (9) – (12).

Proof 2: For the critic network, it updates each value function according to the supervision signal (reward) in Eq. (9). Let the optimal value function be $Q^*(s,a|\theta^{Q^*}) = \sum_0^\infty \gamma^t r_{t+1}$. In the training, minimizing the loss L focuses the value function $Q(s,a||\theta^Q)$ tend to the target (left-hand side in the loss, $r_i + \gamma Q'(s_{i+1}, \pi'(s_{i+1}|\theta^{\pi'})|\theta^{Q'})$), i.e.,

$$Q(s,a||\theta^Q) \to r_i + \gamma Q'(s_{i+1}, pi'(s_{i+1}|\theta^{\pi'})|\theta^{Q'}).$$

According to Robbins-Monro algorithm [39], as the reward is added to the next step value function Q', the value function will converge to the true value function Q^* when $t \to \infty$. So that

$$|Q^*(s,a|\theta^{Q^*}) - Q(s,a||\theta^Q)| \le e,$$

where Q^* is the optimal value function and e is a bounded error value when $t \to \infty$. For the actor network, the policy objective function is defined as the critic network $Q(s,a)$ as in Algorithm 1. By optimizing the policy along the policy gradient of value function $\nabla_a Q(s,a|\theta^Q)$, the policy improves by training iteration. One has

$$\theta^\pi_{t+1} = \theta^\pi_t + \alpha \nabla_{\theta\pi} J,$$

where α is a positive learning rate. During the training, the gradient ascent-based equation ensures the increase of $J(\theta^\pi)$ in general to an local optimal point, which can be expressed as

$$J(\theta^\pi_t) \le J(\theta^\pi_{t+1}).$$

Therefore, the actor network will converge to $\pi(s|\theta^{\pi^*})$ according to the gradient ascent-based direction of the loss function, where π^* is the optimal policy function.

3) Privacy:

We analyze the privacy of the proposed approach in an honest but curious setting. In Algorithm 1, the user $k \in \mathcal{K}^{up}_t$ participating model update requires to upload their gradient to the server for aggregation. As the gradient still contains the client's information implicitly, attack methods such as membership inference attacks [7], model inversion attacks [8], and property inference attacks [9] can be used to infer that information[1]. Both server and clients are assumed to be honest but curious, and we must ensure the information is not leaked to any participants. The secure aggregation protocol guarantees that both the server and clients learn no more information during the aggregation process.

Regarding the privacy against clients, the pairwise masks added to the information of clients in Round 2 make the resulting values appear to be uniformly random if the sum of masked information is equal to the sum of information. Based on this condition, the secure aggregation protocol can guarantee that the clients learn no more information than their own gradient. Any alliance formed by honest but curious clients can only learn the information within the group.

4) Cost:

Implementing the secure aggregation protocol induces additional costs to the clients. Let K be the number of clients and M be the data size of each client. Each client has to compute the key agreements, secret shares of b^k and s^k_{SK}, and the pairwise mask, with computation cost $O(K^2+MK)$. The communication of keys exchanging, secret shares of b^k and s^k_{SK}, masked input $\tilde{x}^k$, and secret shares of other clients, cost it $O(K+M)$. Also, each client must store the keys, secret shares, and data vector for masking of size $O(K+M)$.

For privacy against the honest but curious server, it may unite with other honest but curious clients to form a group. If the number of clients in the group is smaller than K_0, the group cannot learn additional information from the clients outside the group. The server can only learn the aggregated gradient with the inputs of the remaining clients.

Although the proposed approach guarantee the system privacy, there are trade off to the server. Specifically, extra computation, communication, and storage costs are added to the server. Reconstruction of the secrets for each client and generation and removal of the pairwise mask take a computation cost of $O(MK^2)$ to the server. The mediation of all pairwise communication between clients and the receipt of masked input $\tilde{x}^k$ brings a communication cost of $O(K^2+MK)$ to the server. Also, the server has to store the secret shares and a data buffer for calculating the aggregation with a storage cost of $O(K^2+M)$. Table III summarized the cost of the protocol.

5) Scalability:

Regarding the time complexity and scalability of the algorithms employed, it is worth noting that these algorithms are executed during the pre-deployment phase. Consequently, both the server and clients can train their models offline without affecting the real-time processes. After model training, the process becomes a feedforward calculation using train models, which can be done in real-time. Furthermore, due to the federated architecture, each client is responsible for solving their sub-problem in parallel. As a result, the time complexity does not scale with the number of clients, thereby mitigating potential scalability issues.

[1]Various inference attack mechanisms are studied in the literature. Detailed information can be referred to those references.

TABLE III
COSTS OF THE SECURE AGGREGATION PROTOCOL.

	Server	Client
Computation cost	$O(MK^2)$	$O(K^2+MK)$
Communication cost	$O(K^2+MK)$	$O(K+M)$
Storage cost	$O(K^2+M)$	$O(K+M)$

V. EXPERIMENTS

A. Experiment Setup

We perform experiments on two scenarios, namely New York City (NYC) and synthetic scenario, using the proposed approach to evaluate the performance.

1) New York City Scenario: NYC scenario is constructed based on the taxi zone maps[2] in the Manhattan region of NYC. We follow the terminology and description from the dataset, wherein the zone and edge represents the node and link, respectively, as introduced in Section III-B. The map of the taxi zone structure of the NYC scenario is shown in Fig. 3. There is an edge between the two zones if the zones are neighbors on the map. Any isolated zones are ignored from the network and the corresponding data are filtered out. From the dataset and map, there are 63 zones that form a network with an irregular graph structure. There are three different types of vehicles in the dataset, so we consider there are 3 mobility providers to provide mobility services on each edge and we assume the three mobility providers are providing transport services in all zones. After constructing the graph based on the taxi zone structure, we counted the number of edges and found that there are 963 edges in total. Each edge is associated with four utilities: time, discomfort, price, and operation cost. Their values are uniformly generated between 0 and 1, except the price become the upper limited when generating the cost.

The set of passengers $\mathcal{K}$ is simulated based on the NYC datasets. Traffic query and passengers' charateristics processed in the experiments are sampled from the NYC Taxi and Limousine Commission Trip Record Data[3] and Citywide Mobility Survey[4], respectively. In other words, we used the dataset to model passenger behavior and request transport service. Noted that the taxi driver in the dataset is responsible for driving the passenger from the origin to the destination, not requesting the journey. The expected utility weight vector is calculated based on the characteristics for Eq. (7) only, which is unknown to the intelligent scheduler throughout the experiments. Initial passenger satisfaction levels are at level 3. Problem 3 is solved by a standard optimizer in CVXPY [40] after the utility weights are obtained from the agent.

2) Synthetic Scenario: Beside the NYC scenario, a synthetic scenario is also built to provide comprehensive experiments. A square grid with 36 nodes are used to represent the transport network. 6 mobility providers are simulated to offer transport service for any two neighboring nodes, and thus there are 360 edges in the network. The network utility of each edge is generated the same as in the NYC scenario.

[2]https://data.cityofnewyork.us/Transportation/NYC-Taxi-Zones/d3c5-ddgc
[3]https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
[4]https://www1.nyc.gov/html/dot/html/about/citywide-mobility-survey.shtml

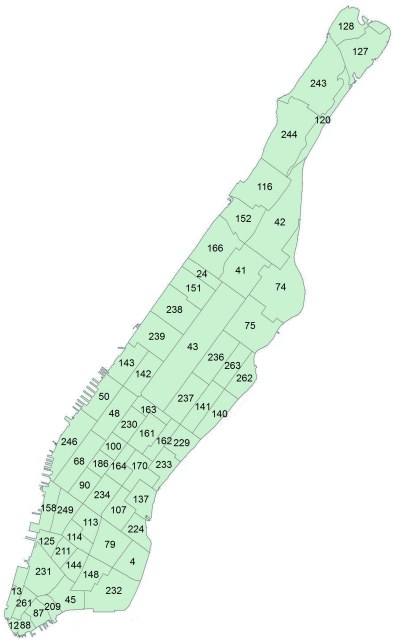


Fig. 3. The map and structure of the NYC taxi zone.

The set of passengers $\mathcal{K}$ in the synthetic scenario is generated with random origin, destination, and characteristics. The calculation of the expected utility weights, initial passenger satisfaction levels, and problem-solving process of Problem 3 are the same as the NYC scenario. Table IV shows the parameters setting of the experiments.

B. Experiment Results

1) Solution quality: To test the effectiveness of the proposed DRL-based journey planning, two baseline methods were evaluated with the FDDPG approach, namely, "random" and "fixed" approaches, which set the utility weights by uniformly random values between 0 and 1, and constant values of ones, respectively. The evaluation metric is based on the profit, which is the reward returned by the environment as discussed in Section IV-A4. We run the experiments for 2000 episodes, and each episode contains 100 iterations of transport queries of a different set of random passengers based on the two scenarios. The average rewards of the approaches of NYC and synthetic scenarios are given in Tables V and VI, respectively. Both scenarios show a similar observation. Among the three compared approaches, the agent approach has the highest average profit on average. Both "fixed" and "random" approaches are much worse than the FDDPG approach. To view the trend during the training in the episodes, we plotted the moving average of rewards of those approaches in Figs. 4 and 5 for NYC and synthetic scenario, respectively. We can see that the reward of the FDDPG approach in both scenarios increases along with the training episodes. This indicates that the FDDPG approach can successfully learn from the experience of the initial random policy as the initial ϵ is high, and most of the actions taken in the initial episodes

TABLE IV
PARAMETER SETTINGS.

Parameter	Definition	Value
NYC transportation network and datasets		
$\|\mathcal{N}\|$	Number of nodes	63
$\|\mathcal{A}\|$	Number of links	963
$\|\mathcal{F}\|$	Number of mobility providers	3
Synthetic transportation network and datasets		
$\|\mathcal{N}\|$	Number of nodes	36
$\|\mathcal{A}\|$	Number of links	360
$\|\mathcal{F}\|$	Number of mobility providers	6
$\|\mathcal{K}^m\|$	Number of passengers per episode	10
C_{ij}^f	Capacity	3
$\|\mathcal{R}\|$	Replay buffer size	10^6
B	Minibatch size	128
γ	Discount factor	0.99
τ	Target network soft update rate	0.001
-	Actor learning rate	0.0001
-	Critic learning rate	0.0003
-	Neural network optimizer	Adam
ϵ_0	Initial random explore rate	1
$\bar{\epsilon}$	Explore rate decay per episode	0.995
T	Number of iteration per episode	100
M	Number of episodes	2000
-	Number of neural network layers	3
-	Number of neurons of each layer	256
-	Range of satisfaction level	1 to 5
$\overline{E}^k$	upper expectation threshold	0.0
$\underline{E}^k$	lower expectation threshold	-0.1

TABLE V
AVERAGE REWARD OF DIFFERENT APPROACHES IN THE NYC SCENARIO.

Approach	Average reward (profit)
FDDPG	358.28
Fixed utility weights	236.77
Random utility weights	174.82

TABLE VI
AVERAGE REWARD OF DIFFERENT APPROACHES IN THE SYNTHETIC SCENARIO.

Approach	Average reward (profit)
FDDPG	410.19
Fixed utility weights	166.80
Random utility weights	102.78

are random. In the later episodes, ϵ decays to a small value, and thus the FDDPG policy with good performance dominates the actions to be taken, further improving the policy for exploitation. For "random" and "fixed", they remain in a low reward level which indicates the incapability of adapting to passengers with different preferences.

2) Passenger Satisfaction: Passenger satisfaction is an important metric to evaluate how well the MaaS coordinator works in addition to the profit. Figs. 6 and 7 show the average

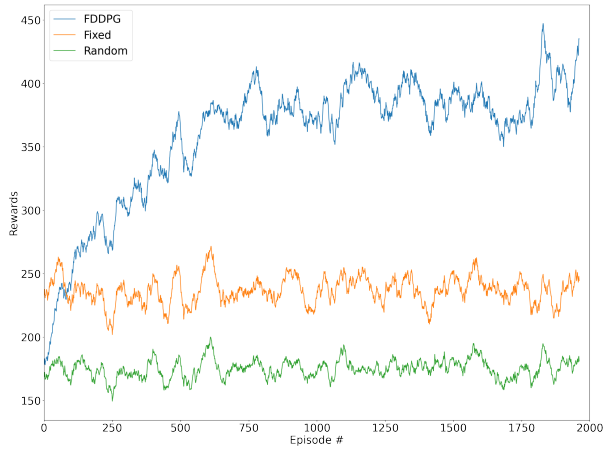


Fig. 4. Moving average of rewards of different approaches in the NYC scenario. The time window of the moving average is equal to 40.

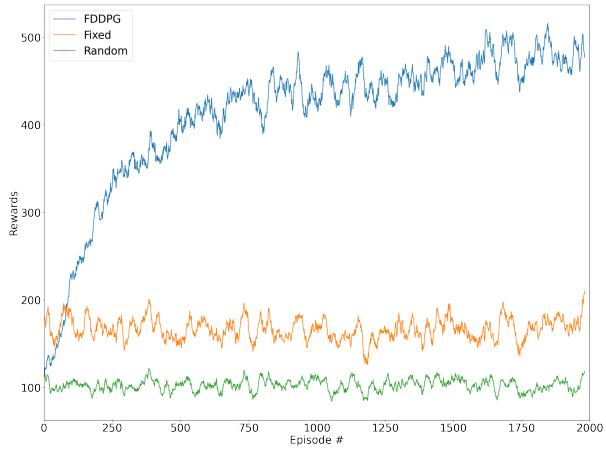


Fig. 5. Moving average of rewards of different approaches in the synthetic scenario. The time window of the moving average is equal to 20.

satisfaction level of each episode of NYC and synthetic scenario, respectively. The color indicates the satisfaction level, as shown in the color bar on the right of the figures. In general, we can see that the trend of satisfaction level of the two scenarios are similar. The FDDPG in both scenarios show a clear increasing trend at the beginning episode and then converge to around 4.5, which suggests the policies improve along with the episode. "fixed" and "random" remain in a lower satisfaction level as similar to the rewards. Therefore, a higher satisfaction level can increase the admission rate and thus increase the profit in the two simulated MaaS scenario.

3) Sampling Mechanism: As discussed in Section IV, an inappropriate sampling mechanism may significantly diminish the performance. To evaluate the effectiveness of the sampling mechanism in FDDPG, we compare the training with a biased baseline method. At each iteration, the baseline samples a mini-batch of experience from an agent instead of multiple agents. Figs. 8 and 9 show the reward along training episode of the FDDPG and the baseline of NYC and synthetic scenario, respectively. From the results, we can see that the FDDPG performs better than the baseline in both scenarios, which in-

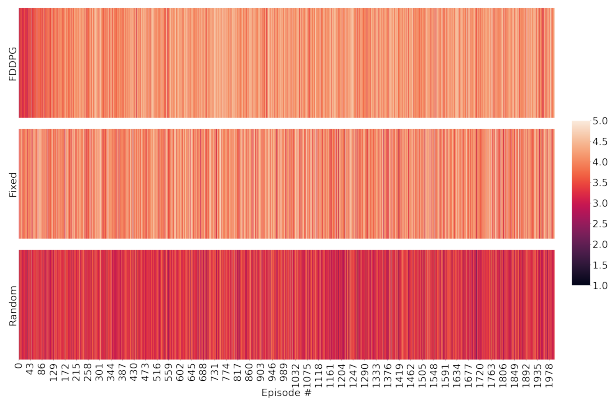


Fig. 6. Average satisfaction level of each episode in the NYC scenario.

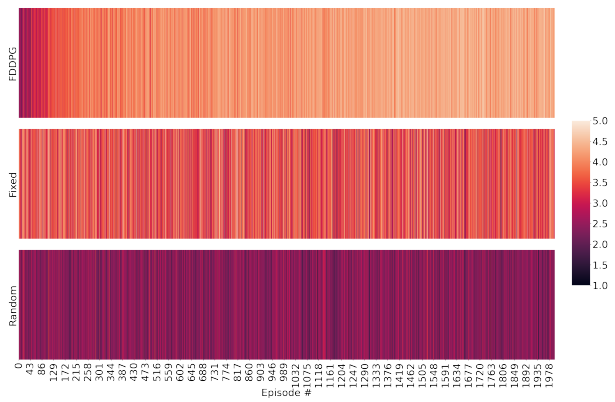


Fig. 7. Average satisfaction level of each episode in the synthetic scenario.

dicates the importance of an appropriate sampling mechanism. Indeed, sampling bias easily occurs in federated training if a group of agents frequently participate in the training than the rest of the agents. Therefore, ensuring the sampling diversity as in the sampling mechanism of the proposed FDDPG may enhance the training performance.

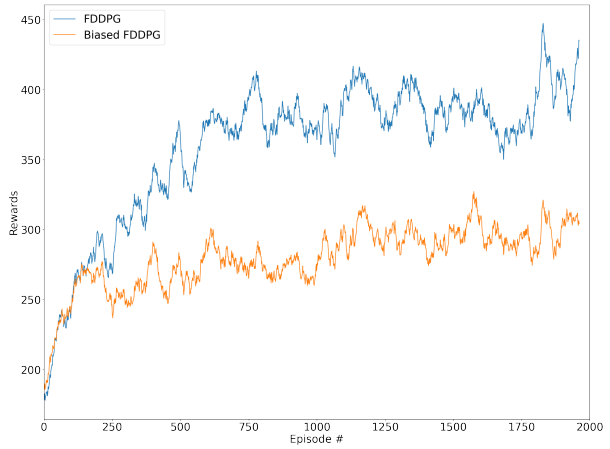


Fig. 8. Moving average of rewards of FDDPG approach and biased baseline in the NYC scenario. The time window of the moving average is equal to 40.

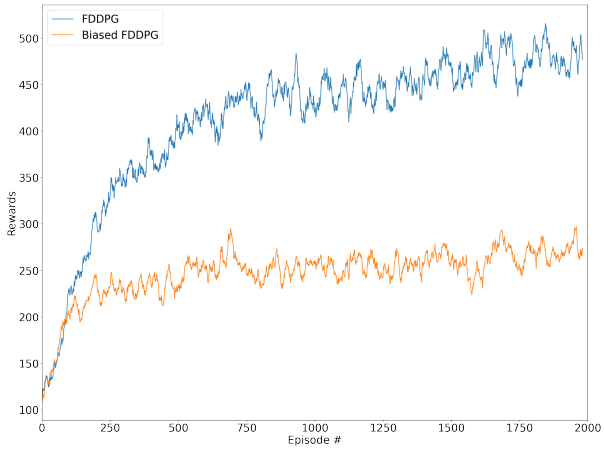


Fig. 9. Moving average of rewards of FDDPG approach and biased baseline in the synthetic scenario. The time window of the moving average is equal to 20.

4) Dropout: In the highly dynamic transportation system, client agents are not dedicated to the training process, and thus they may dropout at any time. We simulate this situation by assigning a dropout probability to each client in each iteration. The agent will not participate in the training according to the dropout probability. Figs. 10 and 11 show the FDDPG approach with different dropout probabilities. In general, the FDDPG with 0.25 and 0.5 dropout probabilities perform similarly to the FDDPG without dropout for both scenarios. On the other hand, the one with a 0.75 dropout probability has a much worse performance compared to others in the synthetic scenario. A slightly worse performance of 0.75 dropout case can be observed from the NYC scenario. This may be due to the decrease in training samples and diversity, which can be expected with such a high dropout probability. However, considering the probability against performance, the performance reduction is insignificant with a high dropout probability of 0.5. So the results in both scenarios suggest that the algorithm is relatively robust to the passenger dropout as long as the number of passengers is larger than a certain threshold, which can be guaranteed by the value of K_0 in the secure aggregation protocol.

VI. Conclusion

The privacy risk in the MaaS platform using the DRL-based approach with passenger behavior and characteristics is an unresolved problem. Curious participants and eavesdroppers may acquire those sensitive information through the privacy loophole of the system. To prevent information leakage, we proposed a privacy-preserving FDDPG architecture that guarantees the information is only accessible by its owner. With this architecture, passengers interested in MaaS can enjoy the offered transport service without being concerned about privacy. The experiment results also show that the MaaS profit and passenger satisfaction improve by about 90% and 15%, respectively, while preserving privacy and maintaining stable training against agent dropout. The trust and participation of

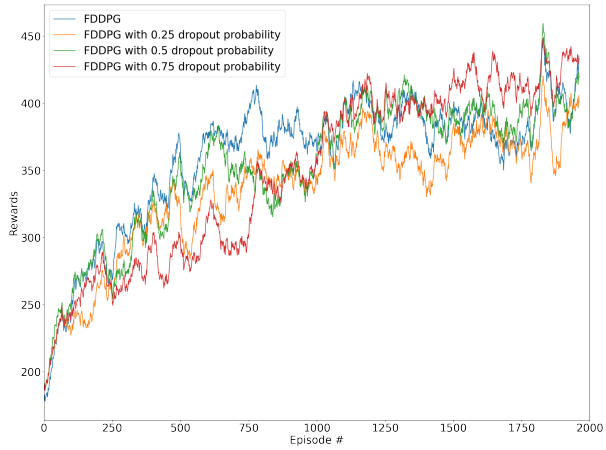


Fig. 10. Moving average of rewards of FDDPG approach with different dropout probabilities in the NYC scenario. The time window of the moving average is equal to 40.

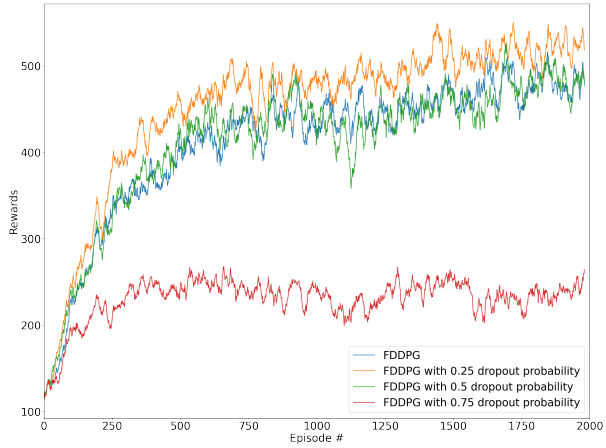


Fig. 11. Moving average of rewards of FDDPG approach with different dropout probabilities in the synthetic scenario. The time window of the moving average is equal to 20.

MaaS and other data-driven transportation systems could be enhanced by the proposed privacy-preserving architecture.

In the future, we may include additional cyber-security measures for the MaaS platform to ensure information privacy in the client's device. We may also extend the privacy-preserving architecture to other DRL-based approaches in transportation, such as the one in [41].

References

[1] K.-F. Chu, A. Y. Lam, and V. O. Li, "Deep multi-scale convolutional lstm network for travel demand and origin-destination predictions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3219–3232, 2019.

[2] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Optimization for dynamic ride-sharing: A review," *European Journal of Operational Research*, vol. 223, no. 2, pp. 295–303, 2012.

[3] K.-F. Chu, A. Y. Lam, and V. O. Li, "Joint rebalancing and vehicle-to-grid coordination for autonomous vehicle public transportation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7156–7169, 2022.

[4] A. Nikitas, K. Michalakopoulou, E. T. Njoya, and D. Karampatzakis, "Artificial intelligence, transport and the smart city: Definitions and dimensions of a new mobility era," *Sustainability*, vol. 12, no. 7, p. 2789, 2020.

[5] K.-F. Chu and W. Guo, "Deep reinforcement learning of passenger behavior in multimodal journey planning with proportional fairness," *Neural Computing and Applications*, vol. 35, pp. 20 221–20 240, 2023.

[6] F. Callegati, S. Giallorenzo, A. Melis, and M. Prandini, "Cloud-of-things meets mobility-as-a-service: An insider threat perspective," *Computers & Security*, vol. 74, pp. 277–295, 2018.

[7] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[8] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[9] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proceedings of the ACM SIGSAC conference on computer and communications security*, 2018, pp. 619–633.

[10] C. D. Cottrill, "Maas surveillance: Privacy considerations in mobility as a service," *Transportation Research Part A: Policy and Practice*, vol. 131, pp. 50–57, 2020.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proceedings of the International Conference on Learning Representations*, 2016.

[12] K.-F. Chu and W. Guo, "Federated reinforcement learning for consumers privacy protection in mobility-as-a-service," in *2023 IEEE 26th International Conference on Intelligent Transportation Systems*. IEEE, 2023, pp. 1–7.

[13] D. Sever, L. Zhao, N. Dellaert, E. Demir, T. Van Woensel, and T. De Kok, "The dynamic shortest path problem with time-dependent stochastic disruptions," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 42–57, 2018.

[14] M. D. M. Molina, B. D. M. Molina, D. C. Pérez, and V. S. Campos, "Connecting passenger loyalty to preferences in the urban passenger transport: Trends from an empirical study of taxi vs. vtc services in spain," *Research in Transportation Business & Management*, vol. 41, p. 100661, 2021.

[15] Y. Wang, X. Lin, F. He, and M. Li, "Designing transit-oriented multimodal transportation systems considering travelers' choices," *Transportation Research Part B: Methodological*, vol. 162, pp. 292–327, 2022.

[16] K.-F. Chu and W. Guo, "Passenger spoofing attack for artificial intelligence-based mobility-as-a-service," in *2023 IEEE 26th International Conference on Intelligent Transportation Systems*. IEEE, 2023, pp. 1–7.

[17] Y. Li, D. Yang, and X. Hu, "A differential privacy-based privacy-preserving data publishing algorithm for transit smart card data," *Transportation Research Part C: Emerging Technologies*, vol. 115, p. 102634, 2020.

[18] T. H. Nguyen, J. Partala, and S. Pirttikangas, "Blockchain-based mobility-as-a-service," in *2019 28th international conference on computer communication and networks (icccn)*. IEEE, 2019, pp. 1–6.

[19] R. Xu, N. Baracaldo, and J. Joshi, "Privacy-preserving machine learning: Methods, challenges and directions," *arXiv preprint arXiv:2108.04417*, 2021.

[20] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[21] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[22] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Proceedings of the NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[23] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[24] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[25] B. Liu, L. Wang, and M. Liu, "Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4555–4562, 2019.

[26] H. Liu and W. Wu, "Federated reinforcement learning for decentralized voltage control in distribution networks," *IEEE Transactions on Smart Grid*, 2022.

[27] M. Xu, J. Peng, B. Gupta, J. Kang, Z. Xiong, Z. Li, and A. A. Abd El-Latif, "Multi-agent federated reinforcement learning for secure incentive mechanism in intelligent cyber-physical systems," *IEEE Internet of Things Journal*, 2021.

[28] Y. Nie, J. Zhao, F. Gao, and F. R. Yu, "Semi-distributed resource management in uav-aided mec systems: A multi-agent federated reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 13 162–13 173, 2021.

[29] M. K. Abdel-Aziz, C. Perfecto, S. Samarakoon, M. Bennis, and W. Saad, "Vehicular cooperative perception through action branching and federated reinforcement learning," *IEEE Transactions on Communications*, 2021.

[30] J. Brickell and V. Shmatikov, "Privacy-preserving graph algorithms in the semi-honest model," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2005, pp. 236–252.

[31] Y. Aono, T. Hayashi, L. Wang, S. Moriai *et al.*, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[32] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[33] G. Musolino, C. Rindone, and A. Vitetta, "Models for supporting mobility as a service (maas) design," *Smart Cities*, vol. 5, no. 1, pp. 206–222, 2022.

[34] C. Lam and W. Ip, "A customer satisfaction inventory model for supply chain integration," *Expert Systems with Applications*, vol. 38, no. 1, pp. 875–883, 2011.

[35] D. Martín-Consuegra, A. Molina, and Á. Esteban, "An integrated model of price, satisfaction and loyalty: an empirical analysis in the service sector," *Journal of Product & Brand Management*, 2007.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

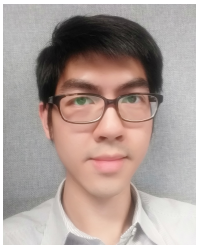
[37] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[38] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

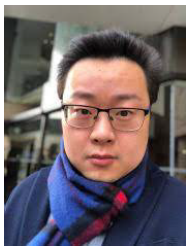
[39] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[40] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[41] K.-F. Chu, A. Y. Lam, and V. O. Li, "Traffic signal control using end-to-end off-policy deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7184–7195, 2022.



Kai-Fung Chu (S'17–M'20) received the Ph.D. degree in Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong, in 2020, and the M.Sc. and B.Eng. (First Class Honors) degrees both in Electronic and Information Engineering from The Hong Kong Polytechnic University, Hong Kong, in 2016 and 2013, respectively. He is a Marie Skłodowska-Curie Fellow at the University of Cambridge. He was a Research Assistant Professor at the Hong Kong Polytechnic University and a research fellow at Cranfield University. He also worked in the industry as an engineer for several years. He is an Area Editor of EAI Transactions on Energy Web. His research interests include artificial intelligence, optimization, autonomous vehicles, and intelligent transportation systems.



Weisi Guo (M'11–SM'16) received his PhD degree from the University of Cambridge, UK in 2011. He was an Associate Professor at the University of Warwick (2012-19) and Turing Fellow (2017-19). He is currently a full Professor and head of the Human Machine Intelligence Group at Cranfield University, UK. He was a winner of the IET Innovation Award. His research interests focus on networks, data science, and autonomy.

2023-10-03

Privacy-preserving federated deep reinforcement learning for mobility-as-a-service

Chu, Kai-Fung

IEEE

Chu K-F, Guo W. (2024) Privacy-preserving federated deep reinforcement learning for mobility-as-a-service, IEEE Transactions on Intelligent Transportation Systems, Volume 25, Issue 2, February 2024, pp. 1882-1896

https://doi.org/10.1109/TITS.2023.3317358