Hindawi
Journal of Advanced Transportation
Volume 2021, Article ID 6654254, 14 pages
https://doi.org/10.1155/2021/6654254

Research Article

Autonomous Bus Fleet Control Using Multiagent Reinforcement Learning

Sung-Jung Wang and S. K. Jason Chang

Department of Civil Engineering, National Taiwan University, No. 1, Sec. 4, Roosevelt Road, Taipei, Taiwan

Correspondence should be addressed to Sung-Jung Wang; d00521018@ntu.edu.tw

Received 6 October 2020; Revised 21 May 2021; Accepted 19 June 2021; Published 3 July 2021

Academic Editor: Saber Fallah

Autonomous buses are becoming increasingly popular and have been widely developed in many countries. However, autonomous buses must learn to navigate the city efficiently to be integrated into public transport systems. Efficient operation of these buses can be achieved by intelligent agents through reinforcement learning. In this study, we investigate the autonomous bus fleet control problem, which appears noisy to the agents owing to random arrivals and incomplete observation of the environment. We propose a multi-agent reinforcement learning method combined with an advanced policy gradient algorithm for this large-scale dynamic optimization problem. An agent-based simulation platform was developed to model the dynamic system of a fixed stop/station loop route, autonomous bus fleet, and passengers. This platform was also applied to assess the performance of the proposed algorithm. The experimental results indicate that the developed algorithm outperforms other reinforcement learning methods in the multi-agent domain. The simulation results also reveal the effectiveness of our proposed algorithm in outperforming the existing scheduled bus system in terms of the bus fleet size and passenger wait times for bus routes with comparatively lesser number of passengers.

1. Introduction

Autonomous vehicles (AVs) are bringing about a radical transformation in the public transportation sector. The fleet control problem associated with AVs has led to new challenges and topics of research. Fagnant and Kockelman [1] defined the current opportunities, barriers, and policy recommendations for AVs. Winter et al. [2] determined the optimal fleet size for a shuttle service of AVs considering the minimal total cost as an objective. Boesch et al. [3] explored the relationship between the served demand and the required AV fleet size. Yap et al. [4] studied the preferences of travelers in using AVs as a last-mile feeder system. Zhang et al. [5] analyzed the generalized cost for autonomous buses, which is one of the key elements for analyzing passenger preferences in using autonomous buses. Chen and Kockelman [6] explored the impact of pricing strategies on the market share of shared autonomous electric vehicles.

Scheltes and de Almeida Correia [7] compared the utility of AVs with that of other transportation modes as the last-mile connection. Montes et al. [8] developed an experimental platform for autonomous buses. Zhu and Kornhauser [9] developed strategies for the fleet management of a large-scale autonomous taxi system with the objective of a minimum fleet size. Hyland and Mahmassani [10] presented a taxonomy for AV fleets and developed a model for AV fleet management problems. Optimal assignment strategies for fully AVs were then explored with the objectives of the minimum fleet miles and minimum traveler wait time [11]. Shen et al. [12] simulated an integrated AV and public transportation system and proposed repurposing low-demand bus routes using shared AVs as an alternative. Salonen and Haavisto [13] presented passenger experiences, perceptions, and feelings when using a driverless shuttle bus in Finland. Abe [14] provided an overview of the impacts of autonomous buses and taxis by quantifying the costs of travel in Japan.

While the above studies and assessments cover diverse research topics, most of them treat the AV fleet control problem as a centralized architecture. Moreover, none of these previous studies have considered that AVs are naturally decentralized and self-governing intelligent agents, which learn to make decisions in an uncertain environment without external intervention.

Fleet control problems are widely considered using module-based dynamic programming to optimize a selected objective function with general heuristics by which agents abide. These mathematical programming models require fixed rules and assumptions to facilitate model convergence on a solution. This is a difficult task in a dynamic environment, as a solution may become outdated when the demand distribution changes. Reinforcement learning is a module-free method in which agents learn the action policy by interacting with a complex environment; this method has been shown to be effective in stochastic and high-dimensional situations. The number of autonomous shuttle trials has increased rapidly over the last few years, and these trials together with research on autonomous bus operation schemes and mobility services [15–17] have begun to attract the interest of various cities, universities, and private companies aiming at improving the safety in urban areas, reducing the cost for last-mile transportation, decreasing congestion, and improving the network connectivity for the user [18].

Thus, this study aims to develop an optimal fleet control scheme for autonomous buses using the reinforcement learning method. The challenge in our work is to teach these agents to make optimal decisions for satisfying passenger trip demand using highly dynamic and stochastic unstructured data and incomplete observations of the environment. In this study, we adopt the multi-agent reinforcement learning (MARL) model and a temporal-difference (TD) learning algorithm to provide an effective method for solving optimal fleet control problems. The major contributions of this study are as follows:

(1) An MARL algorithm was developed to solve the autonomous bus fleet control problem, proving to be a promising approach in such systems

(2) An agent-based simulation platform for autonomous bus fleet control was developed for training and evaluating the proposed algorithm

(3) Experimental results revealed that the proposed algorithm outperformed other MARL methods and could decrease the fleet size and wait times for low-frequency routes in comparison with existing scheduled bus systems

The remainder of this paper is organized as follows. In Section 2, related studies on reinforcement learning and multi-agent issues are reviewed. In Section 3, a new MARL algorithm is formulated and applied to an autonomous bus fleet control problem. In Section 4, the major components and a schematic for autonomous bus fleet control simulator are depicted. In Section 5, an approach to the problem based on the MADDPG algorithm is described, and the test scheme is described in detail, including the operation environment and simulation process for evaluating the performance of the proposed algorithm. In Section 6, the performance of the proposed algorithm is compared with that of the deep Q-network algorithm and the existing scheduled bus system. Finally, concluding remarks are presented in Section 7.

2. Related Works

2.1. Reinforcement Learning. Reinforcement learning (RL) has garnered increasing attention recently. It has a natural application to the case of autonomous agents, which receive sensations as inputs and take actions that affect their environments to achieve their own goals [19]. One of the most important breakthroughs in RL was the development of Q-learning (Watkins, 1989). Q-learning produces a Q-table, which is used by an agent to determine the best action to take based on a given state. However, Q-tables may be ineffective in large state-space environments. The Google DeepMind team created a neural network as an alternative to the Q-table, resulting in the deep Q-learning (DQN) method [20], which has proven to be successful in learning human-level performance for Atari games. To use Q-learning for control problem, first, the values of state–action pairs must be learned, and then these action values should be used directly to implement the policy and select greedy actions. All methods of this form can be referred to as value-based methods. A policy gradient (PG) is another reinforcement approach [21]. This method approximates a stochastic policy. The policy is represented by a neural network whose input is a representation of the state and whose output is the action selection probabilities. The weights in this neural network are represented by the policy parameters, thus producing a policy-based method. The merging of these two algorithmic families (Q-learning and PG) results in the actor-critic (AC) method, which has a separate memory structure to represent the policy explicitly independent of the value function. The policy structure is known as the actor because it is used to select actions, whereas the estimated value function is referred to as the critic because it criticizes the actions made by the actor [19].

Silver et al. [22] introduced an off-policy AC algorithm that learns a deterministic target policy from an exploratory behavior policy with continuous actions, referred to as the deterministic policy gradient (DPG) algorithm. The deep deterministic policy gradient (DDPG) algorithm is a variant of the DPG in which the policy, μ, and critic, Q^{μ}, are approximated with deep neural networks [23]. Schulman et al. [24] proposed and analyzed trust region methods for optimizing stochastic control policies, referred to as trust region policy optimization (TRPO). In TRPO, an objective function (the "surrogate" objective) is maximized subject to a constraint on the size of the policy update. The problem can be efficiently solved approximately using the conjugate gradient algorithm after making a linear approximation of the objective function and a quadratic approximation of the constraint. This algorithm is effective in optimizing large nonlinear policies, such as neural networks. However, TRPO is relatively complicated and is not compatible with architectures that include noise (such as dropout) or parameter

sharing (between the policy and value function, or with auxiliary tasks).

Schulman et al. [25] introduced an algorithm that can realize the data efficiency and reliable performance of TRPO using only first-order optimization with clipped probability ratios. This new method, proximal policy optimization (PPO), demonstrates superior performance in comparison with that of other online PG methods and has become the default RL algorithm at OpenAI. An overview of the RL methods described above is illustrated in Figure 1.

2.2. Multiagent Reinforcement Learning. In MARL methods, autonomous agents learn to solve dynamic tasks online using algorithms that originate in TD RL [26]. Challenges in applying MARL arise from the curse of dimensionality in the number of agents, which incurs high computational costs. However, MARL approaches are becoming increasingly popular. For example, Foerster et al. [27] demonstrated multiple agents sensing and acting in complex environments having partial observability with the goal of maximizing their shared utility. Leibo et al. [28] analyzed the dynamics of policies learned by multiple self-interested independent learning agents using the DQN method, and the results indicated the effect of the sequential nature of real-world social dilemmas on cooperation. MARL also has been applied on transportation system as well such as Sukhbaatar and Fergus [29] that introduced CommNet, which is a backpropagation method for MARL that can learn continuous communication between a dynamically changing set of agents; this method was applied to a four-way traffic junction. Nguyen et al. [30] explored an AC-RL method with a particular decomposition of the approximate action-value function applied to a real-world taxi fleet optimization problem. Lin et al. [31] proposed a contextual MARL framework (DQN) to achieve explicit coordination among a large number of agents for online ride-sharing platforms. However, the above assessments apply traditional RL methods, such as DQN or AC, which are poorly suited to multi-agent environments because the best policy of each agent is affected by changes in the policies of other agents, resulting in a nonstationary environment.

The OpenAI team proposed a counterfactual PG method by expanding DDPG for multi-agent domains in the MADDPG method, which uses the framework of centralized training with a decentralized execution. Using this method, agent populations can discover complex physical and communicative coordination strategies that can coordinate agents in mixed cooperative-competitive environments [32]. The MADDPG method introduces a training regimen utilizing an ensemble of policies for each agent, resulting in more robust multi-agent policies. MADDPG has been used in many applications such as Wang et al. [33] that proposed a data-driven multiagent power grid control scheme using MADDPG for the large-scale energy system with more control options and operating conditions. Zhu et al. [34] applied MADDPG to solve the flocking control problem of multi-robot systems in complex environments with dynamic obstacles. Lei et al. [35] introduced edge computing between terminals and the cloud using MADDPG to address the drawbacks of the traditional power cloud paradigm. In practice, MADDPG operates efficiently on a variety of cooperative and competitive multi-agent environments. However, MADDPG is deterministic policy gradient, intended for continuous action only. Therefore, we replace the DPG with a state-of-the-art stochastic policy gradient method, PPO, to satisfy the discrete action of introducing autonomous bus operation.

3. Methodology

To formulate the autonomous bus fleet control problem (using the notation), we consider a standard RL setting in which an agent (i.e., an autonomous bus) interacts with an environment over a number of discrete time steps. At each time step, t, the agent observes a state, s_t, and selects an action, a_t. The goal of the agent is to maximize the expected reward, r, from each state, s_t, for policy π. This is also called a value-based RL method. One example of such an algorithm is Q-learning, which can be defined as follows:

$$Q^*(s,a) = r + \gamma \times \max Q(s'a'). \tag{1}$$

When the action-value function is represented using a neural network with parameters θ, the DQN algorithm is obtained. DQN learns the action-value function, Q^*, corresponding to the optimal policy by minimizing the loss as follows:

$$L(\theta) = \mathbb{E}_{s,a,r,s'}\left[\left(Q^*(s,a|\theta) - \left(r + \gamma \times \max \overline{Q}^*(s',a')\right)\right)^2\right]. \tag{2}$$

DQN sets a target function, $\overline{Q}$, whose parameters are periodically updated with the most recent θ. The learning process can be stabilized by including the use of an experience replay buffer, D, containing tuples (s,a,s',a').

In contrast to value-based methods, policy-based methods directly adjust parameter θ of policy π to maximize the objective, $J(\theta)$, by taking steps in a direction through gradient ascent, $\nabla_\theta J(\theta)$. This results in the PG algorithm, which can be defined as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim p^\pi, a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)\right]. \tag{3}$$

If the PG works as an actor learning an approximation of the true action-value function, $Q^\pi(s,a)$, through TD or Monte Carlo predictions, this $Q^\pi(s,a)$ is called the critic and leads to a variety of AC algorithms.

It is possible to extend the PG framework to deterministic policies, μ_θ: $S \mapsto A$ with continuous actions, resulting in the DPG algorithm. Under certain conditions, we can formulate the objective function as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim D}\left[\nabla_\theta \mu_\theta(a|s) \nabla_a Q^\mu(s,a)|_{a=\mu\theta(s)}\right]. \tag{4}$$

When the policy, μ_θ, and critic, Q^μ, are approximated with deep neural networks, the DDPG algorithm is obtained.

The above algorithm families generally act on a single agent. For an environment with N agents and the set of

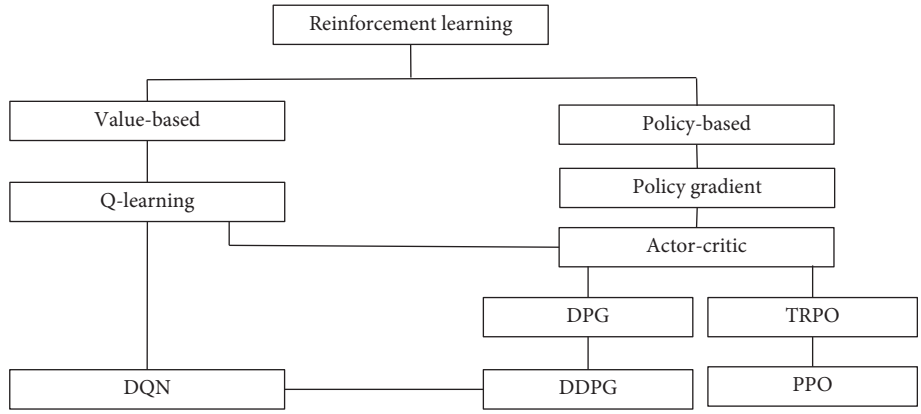

Figure 1: Overview of reinforcement learning methods.

deterministic policies $\mu = \{\mu\theta_1, \ldots, \mu_{\theta N}\}$, we can formulate the gradient of the expected return for agent i as

$$\nabla_{\theta_i} J(\mu\theta_i) = \mathbb{E}_{s,a\sim D}\left[\nabla_{\theta_i}\mu\theta_i(a_i|o_i)\nabla_{a_i} Q_i^{\mu\theta}(s, a_1, \ldots, a_N)|_{a_i=\mu\theta_i(o_i)}\right], \tag{5}$$

where $s = (o_1, \ldots, o_N)$ and consists of the observations of all agents, i.e., the MADDPG algorithm. The framework of the MADDPG method is represented by centralized training with decentralized execution, enabling the policies to use additional information to ease the training. The centralized action-value function, $Q_i^{\mu\theta}$, can be updated as

$$\begin{aligned} L(\theta) &= \mathbb{E}_{s,a,r,s'}\left[\left(Q_i^{u\theta}(s, a_1, \ldots, a_N) - y\right)^2\right], \\ y &= r_i + \gamma Q_i^{\mu\theta'}(s', a_1', \ldots, a_N')|_{a_j'=\mu\theta_j'(o_j)}. \end{aligned} \tag{6}$$

In order to satisfy the discrete action of introducing autonomous bus operation, we replace the DPG with a state-of-the-art stochastic SPG method, PPO. PPO is based on trust region methods (TRPO), and the "surrogate" objective is maximized subject to a constraint on the size of the policy update. The theory governing TRPO is justified by using a penalty instead of a constraint, as follows:

$$\text{maximize}\widehat{\mathbb{E}}\left[\frac{\pi\theta(a_t|s_t)}{\pi\theta_{\text{old}}(a_t|s_t)}\widehat{A}_t - \beta\text{KL}\left[\pi\theta_{\text{old}}(\cdot|s_t).\pi\theta(\cdot|s_t)\right]\right]. \tag{7}$$

Because it is difficult to select a single value of β, an additional modification is proposed as follows:

$$L_t(\theta) = \min\left(r_t(\theta)\widehat{A}_t, \text{clip}\left(r_t(\theta)\right), 1-\varepsilon, 1+\varepsilon\right)\widehat{A}_t, \tag{8}$$

where $r_t(\theta)$ denotes the probability ratio, $r_t(\theta) = \pi\theta(a_t|s_t)/\pi\theta_{\text{old}}(a_t|s_t)$; $\widehat{A}_t$ is an estimator of the advantage function [36] at time step t; and ε is a clip parameter that moves $r_t(\theta)$ away from one to avoid excessively large policy updates. The modified MADDPG algorithm for autonomous bus fleet control is presented in Algorithm 1.

4. Simulator for Autonomous Bus Fleet Control

Unlike mathematical programming problems in which the data are stationary relative to the algorithm and can be evaluated by paradigms, RL applies naturally to the case of autonomous agents, which introduce complex difficulties in training and evaluation. One solution in traffic studies is to create a simulator that is representative of the real world to define a specific problem domain. In this study, we generated large amounts of simulated experience to accelerate the learning process beyond what would be possible using actual experience. The simulator was coded in Python 3.6.1 within an IDE of PyCharm. For the autonomous bus fleet control system, the simulated events were considered as discrete variables (e.g., agents make decisions); therefore, we used SimPy under the MIT License as the simulation framework. SimPy is a discrete-event simulation library. The behavior of the active components (such as vehicles, passengers, or messages) was modeled using processes. All processes live in an environment, which follows the structure of an OpenAI Gym, enabling the application of RL algorithms. For scientific computing, NumPy, a fundamental package, was used, and Matplotlib was applied to plot the training and validation results.

In this study, we built a simulator as an environment for autonomous bus fleet control where the autonomous bus is the only "intelligent" agent. The major components and a schematic of the environment are depicted in Figure 2 and discussed as follows:

Route: a fixed stop/station loop route.

Autonomous buses (V): each agent $i \in V$; there are m agents ($i = 1 \ldots m$). The numerical digit along with agent i represents the number of passengers in the car. Here, "v" indicates that agent i is traveling in the current direction, whereas "-" indicates that agent i is idle and waiting for a policy update. An agent can occupy the same space as another agent without collision. Agents can only travel along the defined route and can perform a U-turn at any stop/station.

Stop/station (S): each stop/station $o \in S$; there are n stops/stations ($o = 1 \ldots n$) where passengers are

for episode = 1 to M **do**
  for $t = 1$ to max-episode-length **do**
    for each agent i
    Execute action $a = \{a_1, \ldots, a_N\}$ and observe reward r and new state s'
    Store (s, a, r, s') in replay buffer D
    $s \leftarrow s'$
    for agent $i = 1$ to N **do**
      Sample a random mini batch of S samples (s^j, a^j, r^j, s'^j) from D
      Set $y' = r_i^j + \gamma Q_i^{\pi'}(s'^j, a_1', \ldots, a_N')$
      Update critic by minimizing the loss
      $\mathscr{L}(\theta) = \mathbb{E}_{s,a,r,s}[(Q_i^{\pi}(s^j, a_1^j, \ldots, a_N^j) - y')^2]$
      Update actor using PPO stochastic PG
      $L_t(\theta) = \min(r_t(\theta)\widehat{A}_t, \text{clip}(r_t(\theta)), 1-\varepsilon, 1+\varepsilon)\widehat{A}_t$
    end for
    Update target network D parameters for each agent i
    $\theta_{\text{old}} \leftarrow \theta$
  end for
end for

Algorithm 1: Modified MADDPG for autonomous bus fleet control.

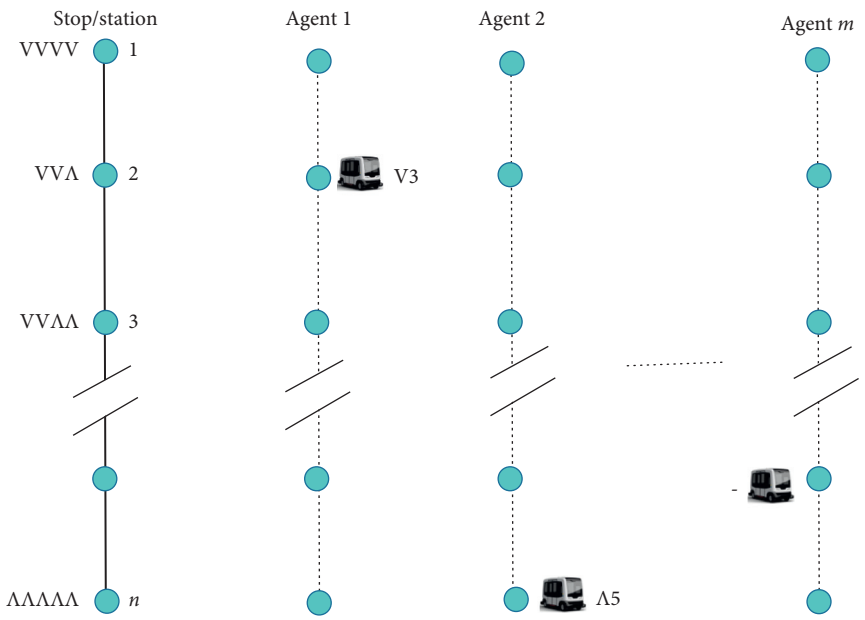

Figure 2: Schematic of the simulation environment.

allowed to enter or leave an agent. Stations 1 and n are terminal stations.

Passenger (P): each passenger $k \in P$; the sign "v" represents the direction in which the passenger plans to travel.

The system dynamics are approximated by the following parameters.

(i) Car capacity (C): 12 seats with a maximum of 6 standees

(ii) Maximum car speed ($V_{\max}$): 20 km/h

(iii) Distance between stops/stations (D): 200–450 m

(iv) Stop time (st): time required for vehicle deceleration (12 s), and acceleration (12 s)

(v) Load time (lt_k): time required for passenger k to enter or exit a car, which is a random variable from a truncated normal distribution with a range of 0.6–6.0 s and a mean of 1.75 s

(vi) Dwell time (dt): time spent in opening and closing bus doors, plus the time spent for passenger flow onto and off the bus

(vii) Passenger arrivals at each stop/station are assumed to be a Poisson distribution, with arrival rates (λ) in a test period

5. Transformation of the Problem

By interacting with the environment, for each agent $i \in V$ at each time step t, the agent i receives the environment's state $s_t \in$ STATU, where STATU is the set of possible states, and selects an action $a_t \in$ ACTS, where ACTS is the set of actions available in state s_t; the goal of the agent is to maximize the expected reward r, for policy π. One time step later, in part as a consequence of its action, agent i finds itself in a new state s_{t+1}. In the following, we provide the state space, action space, and rewards that are required by the algorithm.

(1) State space (STATU): the state space of each agent is composed of two dimensions of situational elements. Specifically, it includes information about the observation space and car states. Observation space constitutes the car position, number of passengers in the car, stop/station call status, and in-car requests. For car states, the space includes moving, idling, intent to station $o = 1$, and intent to station $o = n$.

(2) Action space (ACTS): each agent i chooses the action to execute from its state space in the current environment by using policy π: $S \mapsto A$. We define it as the agent's awareness of eight situational elements in two dimensions. For a cluster of agents, the action space in time step t is represented as follows:

0:_move_move: nonstop,
1:_move_idle: stop at the next stop/station,
2:_idle_ds_move: idling, direction to station $o = 1$; nonstop,
3:_idle_ds_idle: idling, direction to station $o = 1$; stop at the next stop/station,
4:_idle_os_move: idling, direction to station $o = n$; nonstop,
5:_idle_os_idle: idling, direction to station $o = n$; stop at the next stop/station,
6:_idle_idle: idling, continue idling,
7:_idle_intend_os: idling, intend to station $o = 1$,
8:_idle_intend_ds: idling, intend to station $o = n$,
9:_idle_intend_idle: idling, intend to idle.

(3) Rewards: the objective is to minimize the average squared passenger wait time, as this objective tends to maintain low wait times while encouraging fair service for all passengers. The reward function is expressed as

$$-\left(\frac{wt_k}{\gamma}\right)^2. \tag{9}$$

Note that wt_k is the wait time of passenger k, and γ is the reward discount based on Bellman Error.

5.1. Simulation Process. The analyst first defines a scenario, including the fleet size (L), number of stops/stations (n), spatial-temporal demand rate of passenger requests (λ), length of an episode (τ, initial = 1h), and maximum number of episodes (L, initial = 300). Based on the scenario data, the simulation creates a dictionary for the ACTS, and state STATU. The STATU and ACTS are references for building the neural network. The simulation is process-driven; at each process step (Δs), the simulation updates the position and state of the agents and passengers into STATU, and the agent takes action accordingly. A critic value reward of the action is assigned in lieu of a reward function, followed by an actor update action policy for the agent. The simulation then checks whether the Timer of the simulation hour is > 1; then $\tau = \tau + 1$, Timer = 0; this process is repeated until $\tau = L$.

A random generator (class PASSENGER) is employed to generate the passenger origin stop/station (o_k), destination stop/station (d_k), and created time (ct_k) for each passenger $k \in P$.

The class BUSES defines the operating behavior for each agent $i \in V$ and the legal actions $a \in$ ACTS by which agent i abides. The agent i makes decisions to select actions with the aim of minimizing passenger wait times. A decision epoch occurs upon agent arrival or completion of loading. Agent arrival is triggered when an agent reaches a stop/station or on completion of the previous idling event. The loading completion event is triggered when the loading process is complete and the agent is ready to take the next action. Agent i cannot pass a stop/station if a passenger wants to disembark at that location and cannot perform a U-turn until all the passenger requirements in the present direction have been serviced. If agent i is moving, it must decide whether it will stop at the next stop/station. If agent i is idling, it must select an intended movement direction. If agent i is approaching the terminal station, it must stop at the next stop/station. Upon agent i making a decision, the simulation updates the reward for agent i. When agent i is called by passenger k, the system returns "False" if agent i is full; otherwise passenger k enters the list of passengers carried by agent i, and the simulation records the pick-up time (pt_k). When passenger k reaches the destination, they are removed from the list of passengers carried by agent i.

The wait time of passenger k is calculated as ($\mathrm{wt}_k = \mathrm{pt}_k - \mathrm{ct}_k$). Here, ct_k is the time at which passenger k appears at the stop/station, which is given by the random generator (class PASSENGER), and pt_k is the time at which passenger k is picked up. Upon completion of a simulation episode τ, the simulator accounts for the total passenger waiting time $\mathrm{PW}(\tau)$ and the number of passengers served $\mathrm{PS}(\tau)$. The average wait time $\mathrm{AW}(\tau)$of episode τ is then calculated as follows:

$$\mathrm{AW}(\tau) = \frac{\mathrm{PW}(\tau)}{\mathrm{PS}(\tau)}. \tag{10}$$

Figure 3 depicts flowchart of the simulation process. The process marked in pink represents scenario data that the analysis plans to test, whereas the processes marked in blue represent the interfaces with the MARL algorithm.

To train each agent's actor, we used a three-layer TensorFlow neural network. The units of the input layer were the same as the observation space: 100 units in the first hidden layer, 50 units in the second hidden layer, and 10 units of action space in the output layer. For each agent's critic, we used a two-layer

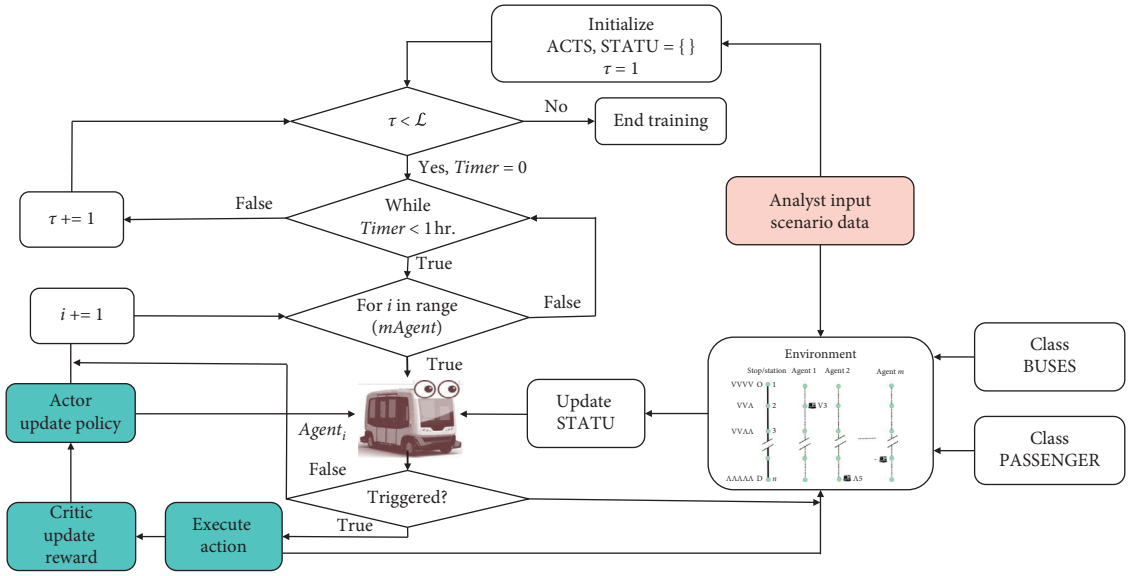

Figure 3: Flowchart of the simulation process.

neural network with the same units of the observation space as those in the input layer, 50 units in the first hidden layer, and one unit in the output layer. The network was trained using the Adam optimizer for faster training. We set each episode equal to one simulation hour, with a maximum of 300 episodes for implementing the training. The actor network's learning rate (a) was set as 1e-4, while the critic's ($ß$) was set as $1e{-}3$. This allowed the critic to learn slightly faster than the actor as the learning of the actor network relies on the critic network.

6. Experiments

Experiments were conducted using the simulator introduced in Section 4. Considering the importance of spatial and temporal resource dynamics, the passenger arrival times and their destinations were set as random seeds. To test the generalization performance of our proposed algorithms, two comparisons were performed: (1) in the RL method comparison, we compared our proposed policy-based algorithm with the valued-based Q-learning algorithm; (2) in the fleet control method comparison, we compared the proposed algorithm with the existing scheduled bus system. Furthermore, the effect of the dispatched fleet size on passenger wait times was also compared for the proposed algorithm and scheduled bus system.

The simulations were run on a standard 64-bit desktop computer with 8 GB of RAM and a 2.40 GHz processor. A single simulation experiment (1 h) for scheduled bus replication takes 2–10 min for completion. Owing to the TensorFlow neural network technology and PPO algorithm, MARL only takes 0.5–4 s to complete a training episode. Experiments with larger fleet sizes, stops, and passenger volumes take longer to run.

6.1. Algorithm Comparison Results. An evaluation of the performance of proposed algorithm was conducted to compare with DQN for single-agent and compare to MADQN for multiagent environments. First, a single agent was used to test the effectiveness of the algorithms and fine-tune the parameters before further building on the simulation. The single-agent environment considered consisted of five stops/stations with a 15-minute headway. Two travel patterns with passenger request rates of 90 and 180 passenger requests per hour were tested. The performance was measured based on the average passenger wait time gained by the platform over 300 episodes. The results indicate that both the MADDPG and DQN algorithms can learn the correct behavior in a single-agent environment. Moreover, MADDPG performs better than DQN, as shown in Figure 4.

The multi-agent environment consisted of 5 buses and 15 stops/stations with a 5-minute headway. Two travel demand patterns of 810 and 1620 passenger requests per hour were tested. The results indicate that the MADDPG algorithm functions effectively; however, the MADQN fails to learn the correct behavior, as shown in Figure 5.

6.2. Comparison of Fleet Control Methods. In this study, we compared the proposed algorithm with a scheduled bus system in different scenarios, namely, a university campus, an industrial zone, and a downtown street. To closely approximate real-world conditions, each scenario was designed based on the bus planning guidelines [37] using the procedure shown in Figure 6.

The passenger service volume figures were adjusted to reflect that the autonomous bus has fewer seats (12) than those of a standard bus (43). In addition, we used the reasonable peak-hour factor of 0.75 suggested by the guidelines instead of the value of 1.0 in the original settings.

In a scheduled conventional bus system, the required fleet size is defined as follows:

$$F = \frac{R}{h}, \tag{11}$$

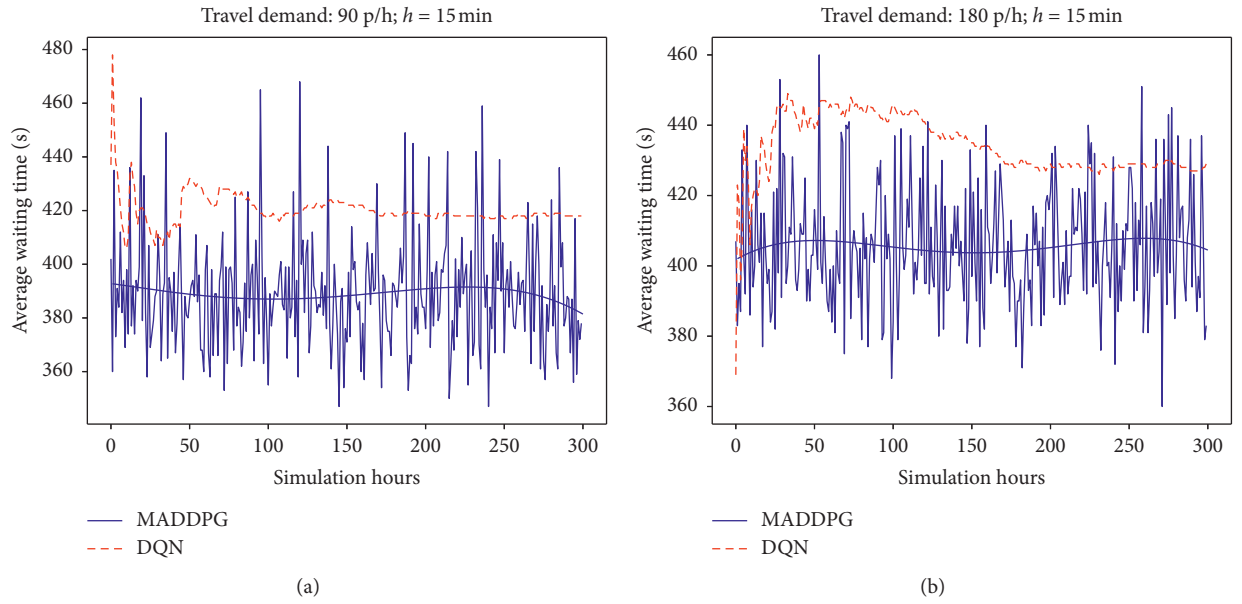

FIGURE 4: Performance comparison between the MADDPG and DQN algorithms for a travel demand of 90 and 180 passengers/*h* in a single-agent environment.

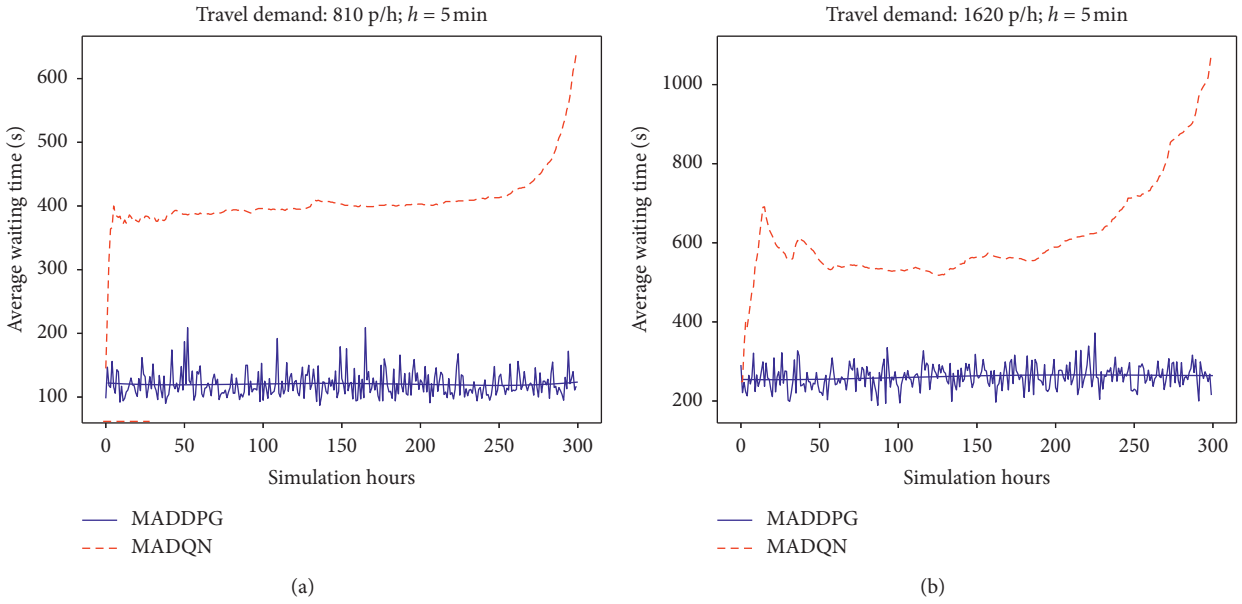

FIGURE 5: Performance comparison between the MADDPG and MADQN algorithms for travel demand patterns of 810 and 1620 passengers/*h* in a multiagent environment.

where R is the round-trip travel time. Vehicles stop at all of the n stops/stations such that

$$R = 2t_o(n-1), \tag{12}$$

where t_o is the average travel time between stops/stations. With these formulations, we can obtain the proper route distance and number of stops/stations. The parameter values and passenger service volume per hour for the three scenarios are summarized in Table 1.

Two metrics were evaluated in this study: the average passenger wait time and vehicle-kilometers per hour.

Average waiting time of passenger: this metric represents the customer service quality. The simulations calculate the results for the MADDPG algorithm and the scheduled bus model using equation (9).

Vehicle-kilometers per hour (VKH): this metric represents the operational cost of the fleet. The simulations

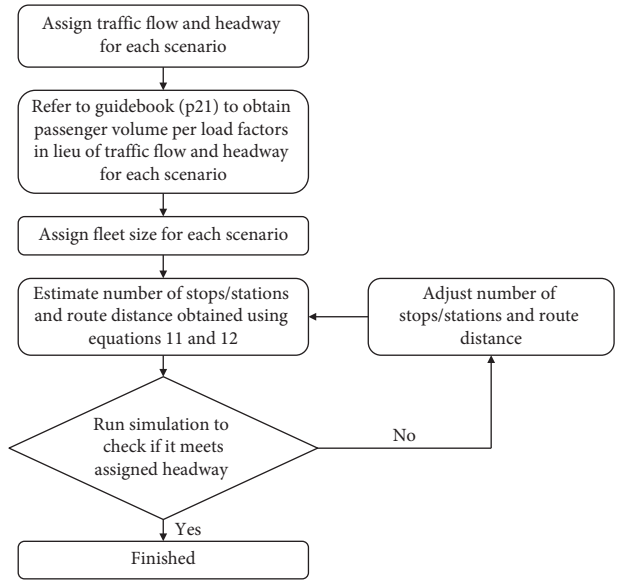

Figure 6: Procedure for design scenarios.

Table 1: Parameter and passenger volumes per hour for three scenarios.

		Scenario 1	Scenario 2	Scenario 3
Simulation environment		University campus	Industrial zone	Downtown street
Traffic flow		Free flow	Stable flow; unconstrained	Stable flow; interference
Fleet size		1	2	5
Number of stops		5	10	15
Headway (min)		15	10	5
Average stop spacing (m)		450	250	200
Load factors	0.00–0.50	90	270	810
	0.51–0.75	135	405	1215
	0.76–1.00	180	540	1620
	1.01–1.25	225	675	2025
	1.26–1.50	270	810	2430

calculate the results for the MADDPG algorithm and the scheduled bus model using the following equation:

$$\text{VKH} = (\text{total moving steps of agents in a episode}) \times D, \tag{13}$$

where D is the average distance between stops/stations.

The performance of the MADDPG was evaluated based on the learning process over 300 episodes. The results of the scheduled bus model were obtained from a 1 h simulation replicated for an average of 10 times.

The comparison between the performance of the MADDPG and that of the scheduled bus methods for three scenarios is presented as follows.

Scenario 1. This scenario was set as a university campus with a short route (3.6 km) and low passenger demand (free flow). The low frequency was assigned as a 15-minute headway with 5 stops/stations and was operated by one bus, similar to the scale of recent autonomous bus pilot projects [18] The MADDPG method was significantly more efficient than the scheduled bus method in terms of both the passenger wait time and VKH in this scenario. This is because MADDPG has a more flexible dispatching strategy; for example, MADDPG allows a bus to idle at a stop/station when no passenger demand exists, or assigns a bus a U-turn to pick up the nearest passengers. A comparison between the results of the two methods is depicted in Figure 7.

Scenario 2. This scenario was set as an industrial zone with a medium-range route (4.5 km) and general passenger demand (stable flow; unconstrained). The headway was 10 min with 10 stops/stations and was operated by two buses. The MADDPG method outperformed the scheduled bus method in terms of the passenger wait time but performed worse than the scheduled bus method in terms of the VKH in this scenario. This is because the objective of MADDPG is to minimize the average passenger wait time, which tends to waste more vehicle kilometers in picking up passengers as early as possible. A comparison between the results of the two methods is depicted in Figure 8.

Scenario 3. This scenario was set as a downtown street with a longer route (5.6 km) and higher passenger demand (stable

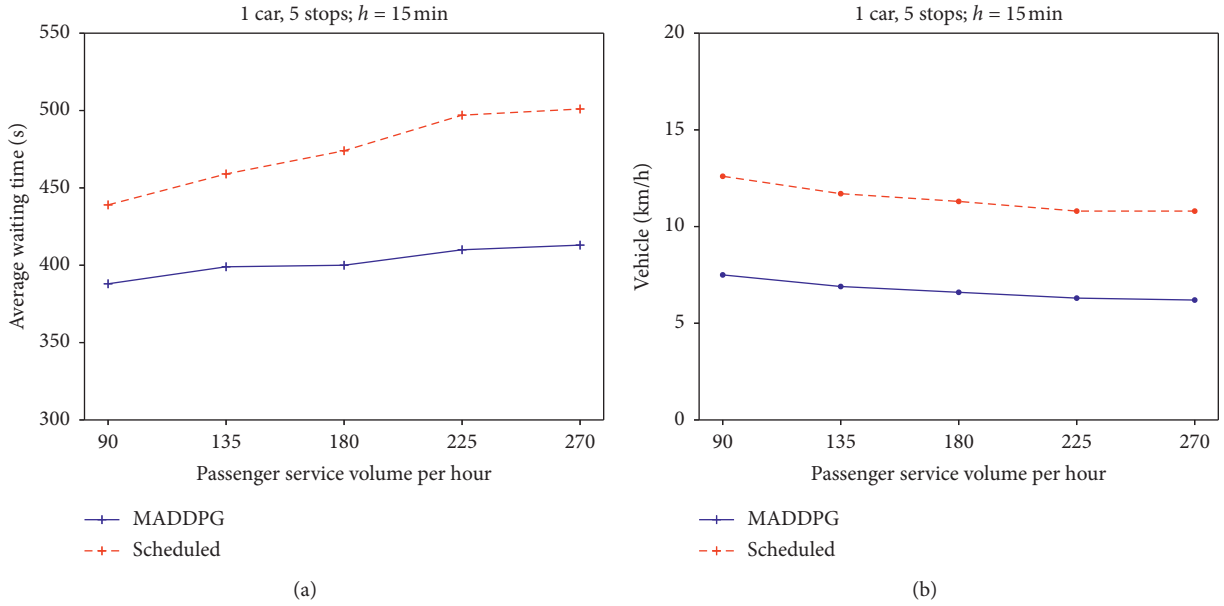

Figure 7: Performance comparison between the MADDPG and SCHEDULED methods in Scenario 1 based on the average passenger wait time and VKH.

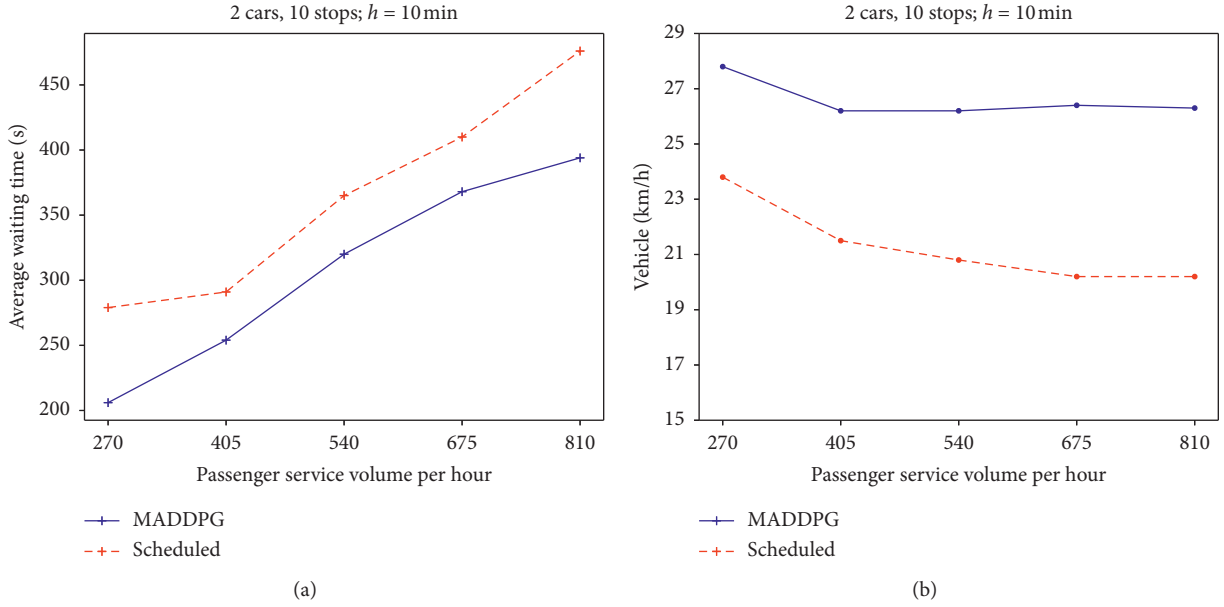

Figure 8: Performance comparison between the MADDPG and SCHEDULED methods in Scenario 2 based on the average passenger wait time and VKH.

flow; interference). The high frequency was assigned as a 5-minute headway with 15 stops/stations, and was operated by five buses. The performance of the MADDPG method was similar to that of the scheduled bus method in terms of the passenger wait time but was worse than that of the scheduled bus method in terms of the VKH in this scenario. This is because the MADDPG free-route advantage for reducing passenger waiting time is constrained by the high-frequency operation. A comparison between the results of the two methods is depicted in Figure 9.

The results indicate that the performance of the MADDPG algorithm is strongly related to the service frequency. The MADDPG method outperformed the scheduled bus system in terms of passenger wait times and VKH when the service frequency was comparatively low, as in Scenario 1. However, for higher service frequencies, such as in

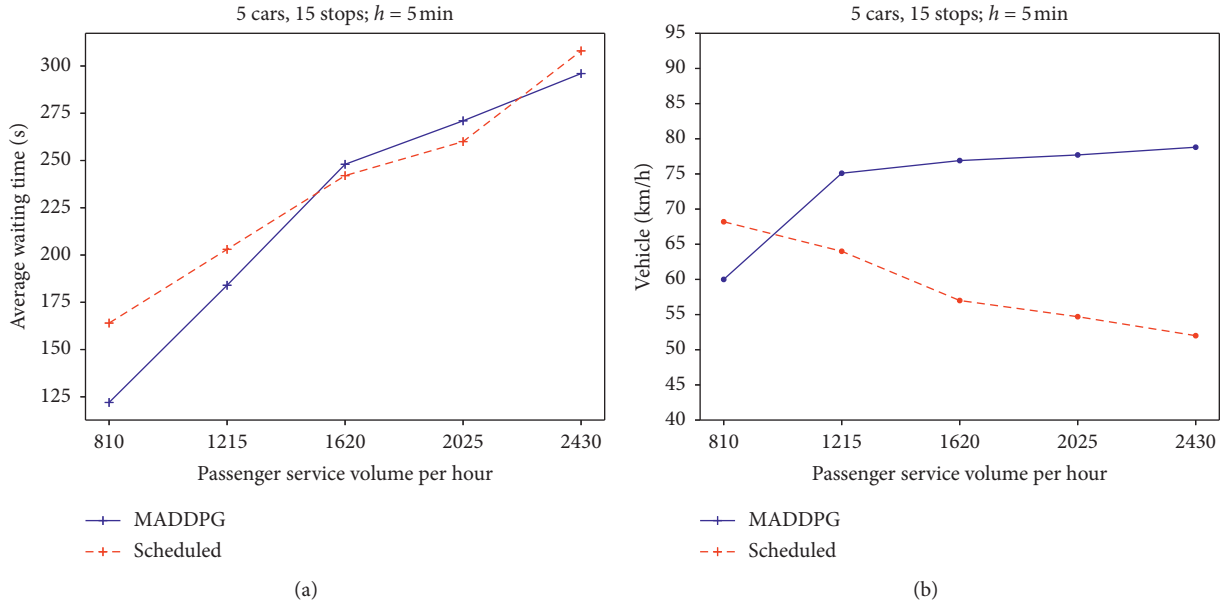

Figure 9: Performance comparison between the MADDPG and SCHEDULED methods in Scenario 3 based on the average passenger wait time and VKH.

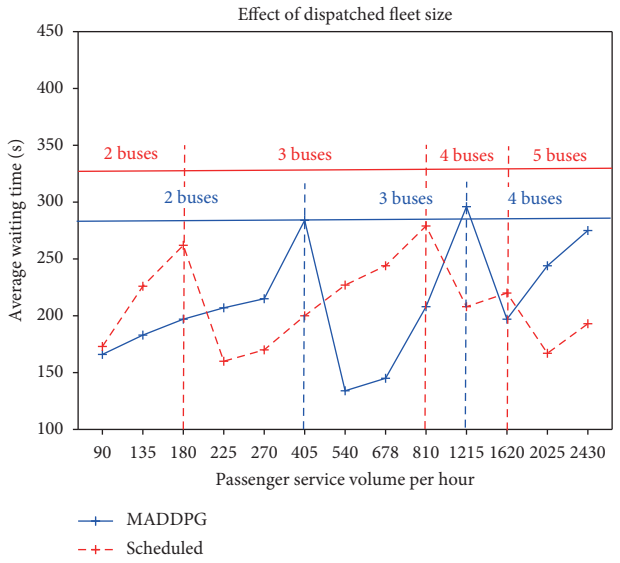

Figure 10: Effect of the fleet size on the passenger wait time with the MADDPG and SCHEDULED methods.

Scenarios 2 and 3, the VKH generated by the MADDPG method increased significantly to minimize the passenger wait time.

6.3. Effect of the Dispatched Fleet Size. In comparison with the scheduled bus headway, which cannot be adjusted exactly according to the demand, the demand responsiveness of the MADDPG method allows agents to operate when there exists a requested service demand. A comparative analysis of the effect of the dispatched fleet size on the level of service was conducted by comparing the mean bus headway between the MADDPG and scheduled bus methods. In an online station system, the average passenger wait time was half the headway. We used a target service level of a 5 min average passenger wait time as the main policy evaluation criterion for a 10 stop/station loop route with a route distance of 4.5 km. Upon an average passenger wait time of over 5 min, an additional bus joined the service. Various travel request rates were tested, as shown in Figure 10.

As expected, increasing the fleet size resulted in a decrease in passenger wait times. The results indicated that the MADDPG method was significantly better than the scheduled bus system in using a smaller fleet size to serve a greater travel demand. For example, in the MADDPG method, two buses could serve up to 405 passengers per hour; however, the scheduled bus system could serve only 180 passengers per hour. The scheduled bus system required five buses to serve the passenger volume of 2430 passengers during the peak hour, whereas the MADDPG method required only four buses for the same passenger volume.

7. Conclusion

7.1. Summary. Fully-autonomous buses promise to increase the competitiveness of public mobility services via eliminating the costs and performance limitations of human drivers and significantly reduce traffic accidents, hence allowing to potentially revolutionize existing public transportation systems. Multi-agent control systems are important owing to the nature of spatial-temporal dynamic environments and in environments where centralized information is unavailable, requiring agents to collaborate with other agents as they may not contain all the data or resources required to achieve the objective. The study findings demonstrated that the proposed algorithm could successfully solve the multi-agent problem of autonomous bus fleet control.

Another important finding presented in this paper is that the proposed method performed better than other RL approaches, such as DQN and MADQN and also outperformed the existing scheduled bus method in reducing fleet sizes and decreasing passenger wait times for lower-frequency routes.

7.2. Contribution and Practical Implications. This paper makes several contributions to the transportation research literature including proposing an MARL algorithm and applied it to obtain the optimal dispatching method for an autonomous bus fleet under several operational rules and constraints, developing a discrete-event simulation platform for autonomous bus fleet operation to simulate real-world traffic activities, enabling the training and evaluation of the MARL algorithm as well as the simulation of scheduled bus methods, and revealing the performance of the proposed algorithm in comparison with the scheduled bus method in three scenario, namely, a university campus, an industrial zone, and a downtown street to closely approximate real-world conditions.

The dispatching model presented in this paper can be employed by autonomous bus fleet operators that plan to offer an on demand service. Relative to scheduled bus method, the more-efficient operation presented in this paper allows fleet manager to either (1) keep their current fleet size and reduce traveler wait times (i.e., improve customer service quality) or (2) decrease their fleet size (i.e., reduce capital costs) but keep traveler wait times at satisfying level. In addition to the above cost and service quality advantages of the proposed method, this study provides a robust solution algorithm to allow AVs to make decisions rather than to provide decision support to vehicle dispatchers, in which a brand new view on fleet control problem is illustrated.

7.3. Future Research. The proposed method is a demand-responsive model that has the potential to fill gaps in the service provision for low-density areas and can tackle the "last-mile" problem, which makes it worth further exploration. Furthermore, the domain of self-driving ride-hailing services (e.g., Waymo and DIDI) suggests an interesting unscheduled, free-size, and free-route environment that may be useful for MARL learning in the future.

Abbreviations

Sets

P: Set of passengers, index by $k \in P$
V: Set of agents, index by $i \in V$
S: Set of stops/stations, index by $o \in S$

Simulation parameters

L: Maximum number of episodes
τ: Episode, initial as 1 h
Timer: Current simulation time
Δs: Simulation process step
F: Fleet size
λ: Spatiotemporal demand rate of passenger requests
C: Capacity of an agent
D: Distance between stop/station
$\mathscr{R}$: Round-trip travel time
h: Headway of successive buses
t_o: Average trip travel time between stop/station
VKH: Vehicle-kilometers per hour
st: Stop time required for an agent at stop/station to decelerate and accelerate again
lt_k: Time required for passenger k to enter or exit a car
dt: Time spent in opening and closing bus doors, plus time spent for passenger flow onto and off the bus
$V_{\max}$: Maximum speed of agent
o_k: Origin stop/station of passenger k
d_k: Destination stop/station of passenger k
ct_k: Created time for passenger k
pt_k: Pick-up time for passenger k
wt_k: Waiting time of passenger k
$\mathrm{PS}(\tau)$: Total number of passengers served in episode τ
$\mathrm{PW}(\tau)$: Total passenger wait time in episode τ
$\mathrm{AW}(\tau)$: Average passenger wait time in episode τ

Reinforcement learning parameters

ACTS: Dictionary of action space
STATU: Dictionary of environment state
D: Experience replay buffer
s: Current state of system $\in$ STATU
a: Legal action $\in$ ACTS
r: Current reward
s': Updated state$\in$ STATU
a': Updated legal action $\in$ ACTS
π: Policy
θ: Parameter of policy
γ: Reward discount
ϵ: Clip parameter of PPO
$\widehat{A}_t$: Estimator of advantage function at time step t
α: Learning rate of actor network
β: Learning rate of critic network
$r_t(\theta)$: Ratio of new/old policy probability.

Data Availability

The data used to support the results of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper exist.

Acknowledgments

This article was subsidized by the Ministry of Science and Technology and National Taiwan University, Taiwan, under grant F49620–93–1–0269 to Sung-Jung Wang. The authors thank Hubert Chen, Chair of Taiwan TURING Drive Corp., for providing insightful comments regarding the practical aspects of autonomous bus operation.

References

[1] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 167–181, 2015.

[2] K. Winter, O. Cats, G. H. d. A. Correia, and B. van Arem, "Designing an automated demand-responsive transport system: fleet size and performance analysis for a campus-train station service," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2542, no. 1, pp. 75–83, 2016.

[3] P. M. Boesch, F. Ciari, and K. W. Axhausen, "Autonomous vehicle fleet sizes required to serve different levels of demand," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2542, no. 1, pp. 111–119, 2016.

[4] M. D. Yap, G. Correia, and B. Van Arem, "Preferences of travellers for using automated vehicles as last mile public transport of multimodal train trips," *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 1–16, 2016.

[5] W. Zhang, E. Jenelius, and H. Badia, "Efficiency of semi-autonomous and fully autonomous bus services in trunk-and-branches networks," *Journal of Advanced Transportation*, vol. 2019, pp. 1–17, 2019.

[6] T. D. Chen and K. M. Kockelman, "Management of a shared autonomous electric vehicle fleet: implications of pricing schemes," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2572, no. 1, pp. 37–46, 2016.

[7] A. Scheltes and G. H. de Almeida Correia, "Exploring the use of automated vehicles as last mile connection of train trips through an agent-based simulation model: an application to Delft, Netherlands," *International Journal of Transportation Science and Technology*, vol. 6, no. 1, pp. 28–41, 2017.

[8] H. Montes, C. Salinas, R. Fernández, and M. Armada, "An experimental platform for autonomous bus development," *Applied Sciences*, vol. 7, no. 11, p. 1131, 2017.

[9] S. Zhu and A. Kornhauser, "The interplay between fleet size, level-of-service and empty vehicle repositioning strategies in large-scale, shared-ride autonomous taxi mobility-on-demand scenarios," in *Proceedings of the Transportation Research Board 96th Annual Meeting*, pp. 17–59, Washington, DC, USA, January 2017.

[10] M. F. Hyland and H. S. Mahmassani, "Taxonomy of shared autonomous vehicle fleet management problems to inform future transportation mobility," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2653, no. 1, pp. 26–34, 2017.

[11] M. Hyland and H. S. Mahmassani, "Dynamic autonomous vehicle fleet operations: optimization-based strategies to assign AVs to immediate traveler demand requests," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 278–297, 2018.

[12] Y. Shen, H. Zhang, and J. Zhao, "Integrating shared autonomous vehicle in public transportation system: a supply-side simulation of the first-mile service in Singapore," *Transportation Research Part A: Policy and Practice*, vol. 113, pp. 125–136, 2018.

[13] A. Salonen and N. Haavisto, "Towards autonomous transportation. passengers' experiences, perceptions and feelings in a driverless shuttle bus in Finland," *Sustainability*, vol. 11, no. 3, p. 588, 2019.

[14] R. Abe, "Introducing autonomous buses and taxis: quantifying the potential benefits in Japanese transportation systems," *Transportation Research Part A: Policy and Practice*, vol. 126, pp. 94–113, 2019.

[15] M. Azad, N. Hoseinzadeh, C. Brakewood, C. R. Cherry, and L. D. Han, "Fully autonomous buses: a literature review and future research directions," *Journal of Advanced Transportation*, vol. 2019, p. 16, Article ID 4603548, 2019.

[16] S. K. J. Chang, D. W. Shen, C.-C. Kung, and Y.-H. Hung, "The trial experience and future prospect of autonomous bus made in Taiwan," *Journal of the Chinese Institute of Civil & Hydraulic Engineering*, vol. 46, no. 2, pp. 26–32, 2019.

[17] C. Iclodean, N. Cordos, and B. O. Varga, "Autonomous shuttle bus for public transportation: a review," *Energies*, vol. 13, no. 11, p. 2917, 2020.

[18] J. Ainsalu, V. Arffman, M. Bellone et al., "State of the art of automated buses," *Sustainability*, vol. 10, no. 9, p. 3118, 2018.

[19] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, USA, 1998.

[20] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[21] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, S. A. Solla, T. K. Leen, and K. Müller, Eds., pp. 1057–1063, NIPS, Barcelona, Spain, 2000.

[22] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the International Conference on Machine Learning*, Beijing, China, June 2014.

[23] T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," 2015, https://arxiv.org/abs/1509.02971.

[24] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the International Conference on Machine Learning*, pp. 1889–1897, Lille, France, July 2015.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, https://arxiv.org/abs/1707.06347.

[26] L. Busoniu, R. Babuska, and B. de Schutter, "Multi-agent reinforcement learning: an overview," in *Innovations in Multi-Agent Systems and Applications*, D. Srinivasan and L. C. Jain, Eds., pp. 183–221, Springer, Berlin, Germany, 2010.

[27] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, D. D. Lee, M. Sugiyama, U. V. Luxburg, L. Guyon, and R. Garnett, Eds., NIPS, Barcelona, Spain, 2016.

[28] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," in *Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems*, pp. 464–473, International Foundation for Autonomous Agents and Multiagent Systems, São Paulo, Brazil, May 2017.

[29] S. Sukhbaatar and R. Fergus, "Learning multiagent communication with backpropagation," in *Advances in Neural Information Processing Systems*, D. D. Lee, M. Sugiyama, U. V. Luxburg, L. Guyon, and R. Garnett, Eds., pp. 2244–2252, NIPS, Barcelona, Spain, 2016.

[30] D. T. Nguyen, A. Kumar, H. C. Lau et al., "Policy gradient with value function approximation for collective multiagent planning," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pp. 4319–4329, NIPS, Barcelona, Spain, 2017.

[31] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1774–1783, ACM, London, UK, August 2018.

[32] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, I. Mordatch et al., "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pp. 6379–6390, NIPS, Barcelona, Spain, 2017.

[33] S. Wang, J. Duan, D. Shi et al., "A data-driven multi-agent autonomous voltage control framework using deep reinforcement learning," *IEEE Transactions on Power Systems*, vol. 35, no. 6, pp. 4644–4654, 2020.

[34] P. Zhu, W. Dai, W. Yao, J. Ma, Z. Zeng, and H. Lu, "Multi-robot flocking control based on deep reinforcement learning," *IEEE Access*, vol. 8, pp. 150397–150406, 2020.

[35] W. Lei, H. Wen, J. Wu, and W. Hou, "MADDPG-based security situational awareness for smart grid with intelligent edge," *Applied Sciences*, vol. 11, no. 7, p. 3101, 2021.

[36] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, https://arxiv.org/abs/1506.02438.

[37] KFH Group, *Transit Capacity and Quality of Service Manual*, TRB, Washington, DC, USA, 3rd edition, 2013.