Cloud Computing-based Parallel Deep Reinforcement Learning Energy Management Strategy for Connected PHEVs

Tong Sun, Chao Ma, Zechun Li and Kun Yang

Abstract—**This paper proposes a novel cloud computing-based parallel deep reinforcement learning (DRL) energy management strategy (EMS) for connected plug-in hybrid vehicles. First, a proximal policy optimization (PPO) algorithm is developed. Since the cloud computing can reduce the computational burden of the connected vehicles, the PPO is deployed in the proposed cloud computing-based EMS. In order to improve the strategy adaptation, a parallel mechanism is proposed to achieve the information interaction with multiple vehicles. Considering the real-time control requirements, thread pool is proposed and applied in the cloud computing based parallel EMS. The thread pool-based strategy provides an efficient real-time control ability and strategy improvement solution. To verify the PPO based EMS, the dynamic programming, deep Q-network and double deep Q-network strategies are developed for comparison. It is found that the PPO can achieve similar fuel efficiency improvement with the DP strategy among the three DRL algorithms. For parallel training of multiple connected vehicles, the cloud computing-based parallel EMS improves fuel economy by approximately 7.7%. Thread-pool based parallel real-time EMS reduces average time for computational interactions by 20% and further improves the fuel efficiency. The proposed strategy has the advantages of real-time control, adaptability and continuous learning for improved fuel efficiency.**

Index Terms—**Deep Reinforcement Learning, Cloud Computing, Connected Vehicle, Asynchronous Mechanism, Thread Pool, Energy Management Strategy.**

I. Introduction

WITH the increasing severity of energy issues and environmental pollution, the electrification of automobiles has emerged as an inexorable trend in the evolution of the automotive industry [1, 2]. Plug-in hybrid vehicles (PHEVs) represent a promising solution [3]. The EMS plays a pivotal role in the efficient operation of PHEVs [4]. However, due to the structural complexity of the powertrain and the uncertainty of driving scenarios, the design of an efficient and adaptive EMS is a challenging task [5]. There exist three primary categories of EMSs for PHEVs [6]. Rule-based methods are commonly utilized to accomplish real-time control, however, the efficacy is contingent upon the expertise of the engineers [7]. To minimize reliance on professionals, optimization-based methods have been developed [8]. The optimization-based methods include dynamic programming (DP) [9, 10], linear programming (LP) [11], model predictive control (MPC) [12–14], equivalent fuel consumption minimization strategy (ECMS) [15, 16], game theory [17], etc. However, the current well-established strategies entail heavy computational burdens and are less adaptable to complex driving conditions [18, 19].

DRL-based EMSs are highly capable of learning and adapting under complex driving cycles [20–22]. Wu et al. proposed a deep Q-network algorithm based EMS, The tables utilized in conventional Q-learning were substituted with deep neural networks (DNN), which can better handle the multidimensional state space in EMS problems [23]. Qi et al. implemented DQN with dueling networks to further accelerate the learning or convergence process [24]. Lian et al. embedded expert knowledge into deep deterministic policy gradient (DDPG) strategy and obtained better fuel economy [25]. Additionally, a PPO based EMS incorporating the thermal characteristics of batteries was proposed, achieving the optimal balance among multiple objectives through intelligent weight adjustments during the training phase [26]. While DRL-based EMSs have demonstrated exceptional theoretical performance, they encounter substantial challenges in real-world vehicle applications. Frequent real-time update policies require the controller to have strong computational power, which is not allowed in many scenarios [27].

The utilization of technologies related to internet of vehicles (IOV) and cloud computing can increase computing power [28]. Liu et al. proposed an event-triggered EMS based on vehicle cloud optimization to significantly improve the fuel economy of plug-in hybrid buses [29]. Hu et al. proposed a cloud-based schedule training framework that reduces the computational burden on edge devices [30]. Zhang et al. developed a two-layer EMS using cloud computing and IOV technologies for globally optimizing energy consumption and battery state of health [31].

Most current DRL-based research has focused on training and implementing EMS for individual vehicles, with limited exploration of the concurrent capabilities and benefits of cloud computing. Thus, this study innovatively develops a cloud computing based parallel DRL EMS. An asynchronous mechanism in the cloud is utilized to provide parallel control for multiple connected vehicles. Furthermore, the thread pool is employed to facilitate real-time strategy updates and improvements by utilizing data from multiple connected vehicles.

This paper is structured as follows: Section 2 introduces the PHEV powertrain model. The PPO based EMS is pro-

Manuscript received January 3, 2024; revised May 9, 2024. This work was supported by the National Natural Science Foundation of China under Grant 51605265.

Tong Sun is a postgraduate student of Shandong University of Technology, Zibo 255000, China (e-mail: i@suntong.cc)

Chao Ma is a professor of Shandong University of Technology, Zibo 255000, China (corresponding author to provide email: mc@sdut.edu.cn)

Zechun Li is a postgraduate student of Shandong University of Technology, Zibo 255000, China (e-mail: 18365896328@163.com)

Kun Yang is a professor of Shandong University of Technology, Zibo 255000, China (e-mail: yangkun@sdut.edu.cn)

posed in Section 3. The cloud computing-based parallel EMS is designed, which includes parallel EMS training framework and parallel real-time EMS in Section 4. Simulation results and test results analysis, are discussed in Section 5. Section 6 concludes the paper.

II. Dynamic Modeling and Control of PHEV

A. Configuration of PHEV

This study selects the power-split PHEV as the research subject. The powertrain configuration is illustrated in Figure 1, which includes traction motor (Motor1), generator (Motor2), engine, planetary gear set and battery. Specifically, the engine is linked to the planetary carrier, the sun gear is linked to Motor2, Motor1 is linked to the ring gear, and the speed of Motor1 is synchronized with the vehicle speed. The primary parameters of PHEV are illustrated in Table I.

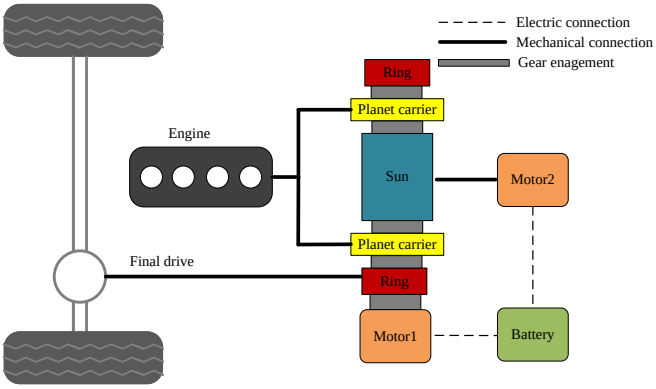

Fig. 1. Configuration of the PHEV powertrain.

TABLE I
Vehicle Parameters

Parameters	Value	Unit
Full mass	1801	kg
Motor1 rated/peak power	25/50	kW
Motor2 rated/peak power	15/30	kW
Maximum engine power	57	kW
Battery rated voltage	237	V
Battery capacity	39	Ah
Final gear ratio	4.05	-
Number of ring gear	78	-
Number of sun gear	30	-

B. The power demand Model

In this study, only the impacts of vehicle longitudinal dynamics are taken into account. The power required by the vehicle for a specific driving cycle is calculated as depicted in Equation 1.

$$\begin{cases} P_{dem} = (F_f + F_w + F_i + F_j)v \\ F_f = mg\cos\theta \\ F_w = \frac{1}{2}C_d A\rho v^2 \\ F_i = mg\sin\theta \\ F_j = \delta m\frac{dv}{dt} \end{cases} \tag{1}$$

where P_{dem} is demand power, F_f, F_i, F_w and F_j represent rolling resistance, slope resistance, air resistance and acceleration resistance respectively, f is rolling resistance coefficient, g is acceleration of gravity, ρ is air density, θ is road slope, a is longitudinal acceleration, C_d is the air resistance coefficient, δ is rotating mass conversion factor, m is vehicle mass, A is windward area.

The power-split powertrain system utilizes a planetary gear mechanism as the power coupling device. The torque-speed relationship of the planetary gear set can be expressed as:

$$\begin{cases} T_s : T_r : T_c = 1 : \alpha : -(1+\alpha) \\ \omega_s + \alpha\omega_r = (1+\alpha)\omega_c \end{cases} \tag{2}$$

where α represents the gear ratio of sun and ring gear, T_s, T_r and T_c represent their respective torques. ω_s, ω_r and ω_c represent the speed of sun gear, ring gear and planetary carrier.

C. Powertrain system model

Without any mechanical coupling between the engine and the wheels, the engine can be operated at the optimal efficiency point corresponding to any power demand. The fuel consumption of the engine can be calculated based on the real-time torque and speed. The fuel consumption is represented by Equation 3:

$$\begin{cases} \dot{m}_f = f(T_e, \omega_e) \\ Fuel = \int_0^T \dot{m}_f dt \end{cases} \tag{3}$$

where $\dot{m}_f$ represents the engine fuel consumption rate, T_e denotes the engine output torque, ω_e is the engine speed.

In the motor model, the numerical model of motor efficiency is a function of speed and torque, given by:

$$\eta_{mot} = f(T_{mot}, \omega_{mot}) \tag{4}$$

where η_{mot} is motor efficiency, T_{mot} is the motor torque, ω_{mot} is the motor speed.

In this study, the fluctuations in battery temperature and battery degradation are not taken into account. A simplified battery is developed and the corresponding equations are shown as follows:

$$\begin{cases} P_{batt}(t) = V_{oc}(t)\cdot I_{bat}(t) - R_0\cdot I^2(t) \\ I_{bat}(t) = \frac{V_{oc}(t) - \sqrt{V_{oc}^2(t) - 4\cdot R_0\cdot P_{batt}(t)}}{2\cdot R_0} \\ \dot{SOC} = -\frac{I_{bat}(t)}{Q_{bat}} \end{cases} \tag{5}$$

where I_{bat} is the battery current and Q_{bat} is the battery capacity. R_0 is the internal resistance, V_{oc} denotes the open-circuit voltage, P_{batt} is the the battery output power.

III. DRL based EMS for PHEV

A. The PPO algorithm

The PPO algorithm adopts actor-critic architecture which is a DRL algorithm based on policy gradient. The PPO algorithm is insensitive to changes in hyperparameters and has the advantages of training stability and strong robustness [32]. In contrast to other reinforcement learning algorithms, PPO exhibits faster convergence and reduced time costs [33]. Therefore, the PPO algorithm is applied to the EMS of the target PEHV.

Typically, PPO uses two policy networks to represent the old and new policies separately. To simplify the process of synchronizing the parameters of two policy networks, this study adopts an alternative scheme where PPO uses one policy network. The experience pool stores the logarithmic probability density of each action (the old strategy) in addition to the state, action, and reward. This single-policy network scheme makes the overall framework structure simpler and facilitates subsequent framework construction. The specific flow of algorithmic interactions and updates is shown in Algorithm 1.

Algorithm 1 PPO

Initialize policy network $\pi(s_t, a_t)$ with network parameters θ
Initialize iteration count M, action collection a, state collection s, discount factor γ, ϵ, step length α
Set experience pool ep with capacity N_e, batch size B
for e from 1 **to** M **do**
  State: s=s_0
  for t from 1 **to** T **do**
    Feed s_t into the policy network
    Select the action a_t and obtain a_{prob}
    Execute action a_t, the obtain s_{t+1} and reward r_t
    Store the experience data $(s_t, a_t, a_{prob}, r_t, s_{t+1})$ in ep
  end for
  for k from 1 **to** K_{epochs} **do**:
    Value network computes the value function
    Update network parameters of policy network by $L(\theta)$
  end for
end for
Output the final policy

The PPO algorithm uses the clip method to limit the range of policy updates to make the learning process smoother and more stable [34]. The specific loss function of PPO algorithm is constructed as follows:

$$L(\theta) = \hat{E}_t \left[\min \left(\frac{\pi_\theta(a_t, s_t)}{\pi_{\theta_{old}}(a_t, s_t)} \hat{A}_t, clip \left(\frac{\pi_\theta(a_t, s_t)}{\pi_{\theta_{old}}(a_t, s_t)}, 1-\varepsilon, 1+\varepsilon \right) \hat{A}_t \right) \right] \tag{6}$$

where π_θ and $\pi_{\theta_{old}}$ represent new and old policies. ε is a hyperparameter employed to regulate the magnitude of variances between the policy distribution before and after the update, ε is 0.2. $\hat{A}_t$ denotes the advantage value, which is calculated using the generalized advantage estimation (GAE) algorithm. The GAE algorithm is an improved advantage function estimation method can effectively reduce the variance of gradient estimation [35]. The specific algorithm is shown in Equation 7.

$$\begin{cases} \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \\ \hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \end{cases} \tag{7}$$

where γ is discount factor, V is the current value function, r_t is the reward at timestep t, λ is used to control the weighting of historical experience in the update process. δ_t is the advantage function at timestep t.

B. PPO-based EMS

This study develops a PPO-based EMS for PHEV. In order to effectively describe the EMS for the PHEV using the PPO algorithm, the fundamental components of the algorithm, namely state, action, and reward function, are initially defined.

The environment state is characterized by the SOC, vehicle speed (v), and acceleration (a). The state can be represented as:

$$S = \{SOC, v, a\} \tag{8}$$

The action is defined as the engine power output, as shown in Equation 9. To reduce the exploration space of actions, the operating point of the engine is chosen along the optimal operating line [36].

$$A = \{P_{eng}\} \tag{9}$$

where P_{eng} represents real-time engine power output.

In order to guide the agent to learn an effective energy management strategy, a suitable reward function should be designed to evaluate the effectiveness of the action. For the EMS of PHEV, both fuel consumption and electrical consumption should be considered. The reward function is formulated as:

$$\begin{aligned} Reward &= f\left(m_{ef}, SOC\right) \\ &= \begin{cases} -\dot{m}_{ef} & SOC_t < SOC < SOC_m \\ -\left(\alpha\dot{m}_{ef} + \beta\left(SOC_t - SOC\right)^2\right) - \lambda & SOC \le SOC_t \end{cases} \end{aligned} \tag{10}$$

where $\dot{m}_{ef}$ represents the instantaneous equivalent fuel consumption per time step, which is the sum of the actual fuel consumption and the equivalent fuel dissipation converted from electricity. SOC_m is the maximum SOC and SOC_t is expected terminal SOC. α is the weight of equivalent total fuel consumption, β is the penalty coefficient employed to constrain the significant fluctuation of SOC, λ is penalty compensation coefficient.

The PPO-based EMS framework is illustrated in Figure 2. The interactive environment is the process of driving the PHEV in a specified driving cycle. The policy network outputs a probability distribution (a_{prob}) over the action space and samples actions based on the current environment state (S_t): velocity, acceleration, and SOC. The sampled action (engine output power a_t) is used to interact with the environment to obtain the next state and reward. The interaction data is stored in an experience pool for sampling by the value networks and policy network. The policy loss function L_θ updates the network parameters θ based on the $\pi_\theta(a_t, s_t)$ generated by current policy and $\pi_{\theta_{old}}(a_t, s_t)$ (a_{prob}), as well as the advantage function $\hat{A}_t$ generated by the value network. The strategy is progressively optimized through multiple iterations such that the reward converges to a maximum value.

IV. Cloud computing-based parallel DRL EMS

In this section, the PPO-based EMS is deployed on a cloud server considering the limitation of the computing power of the connected vehicles. Then a cloud computing based parallel EMS training framework is proposed. The cloud computing-based EMS interacts with multiple connected

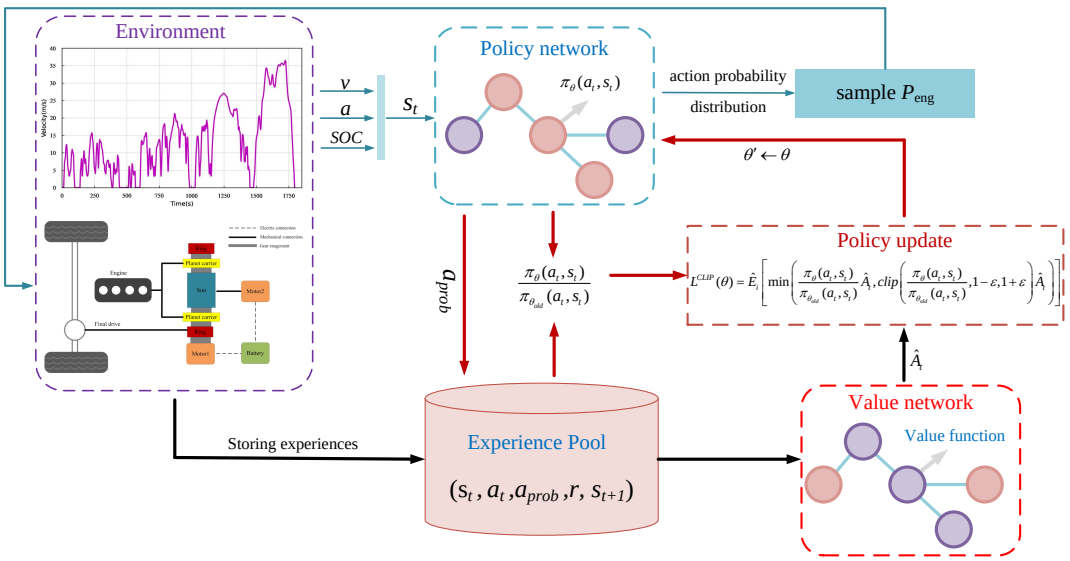

Fig. 2. PPO interactive process.

vehicles in parallel through the asynchronous mechanism, which can fully utilize the high-performance computing power of the cloud and the large amount of data resources of multiple connected vehicles. To better meet the requirements for real-time operation, a cloud computing-based parallel real-time EMS is developed, integrating the parallel EMS training framework with a thread pool mechanism.

A. Asynchronous mechanism

1) Event-driven asynchronous model: The cloud server operating system kernel provides asynchronous mechanisms such as asynchronous system calls and asynchronous event notifications. While waiting for the completion of the asynchronous operation, the main thread is capable of executing other tasks in parallel. This asynchronous mechanism can be concretely realized as an event-driven asynchronous model. The logical framework of the model is illustrated in Figure 3, where the architectural components of the event-driven asynchronous model can be divided into three parts: the event producer, the event broker, and the event consumer.

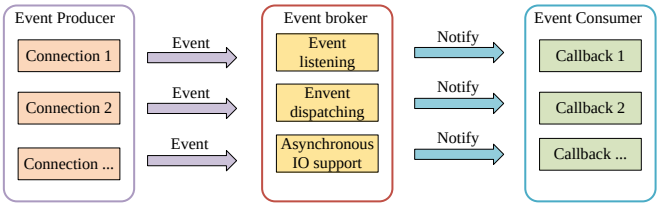

Fig. 3. Event-driven asynchronous model.

2) Application of asynchronous mechanisms in parallel EMS: In the cloud computing-based parallel EMS, multiple connected vehicles act as event producers, which generate a large number of network links and state data requests. The cloud computing-based EMS acts as an event consumer and triggers the control and update logic based on the state data requests from multiple connected vehicles. The cloud computing-based EMS handles the network connection requests of connectd vehicles and the network IO operations generated by control interactions, all of which occur asynchronously. The asynchronous logic is shown in Figure 4. The cloud computing-based EMS sends asynchronous requests to the cloud processor system kernel, which can then perform other operations unrelated to the previous call without waiting for a response. Thus a large number of parallel interactions can be performed without blocking.

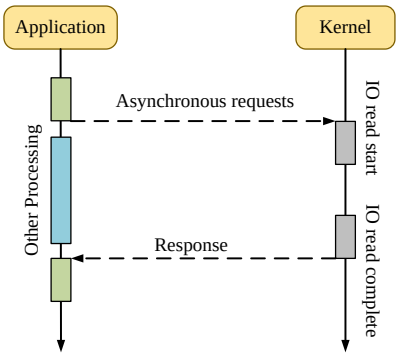

Fig. 4. Asynchronous principle.

B. Cloud computing-based parallel EMS training framework

If DRL-based EMSs are trained with a specific driving cycle, the trained strategies usually perform well only in the learned driving cycles and may not yield satisfactory outcomes in other unknown driving cycles. The adaptability to new driving conditions usually requires retraining on top of the existing strategies. In order to improve this problem, a more flexible strategy needs to be developed which can better handle different driving cycles and provide wider adaptability.

Therefore, a parallel EMS training framework based on an asynchronous mechanism is developed. The cloud computing based EMS can process the communication requests from connected vehicles running under different driving cycles in parallel and utilize the experience of different driving cycles to train the EMS. A schematic of the EMS training framework is shown in Figure 5. The global policy network interacts directly with multi-connected vehicles.

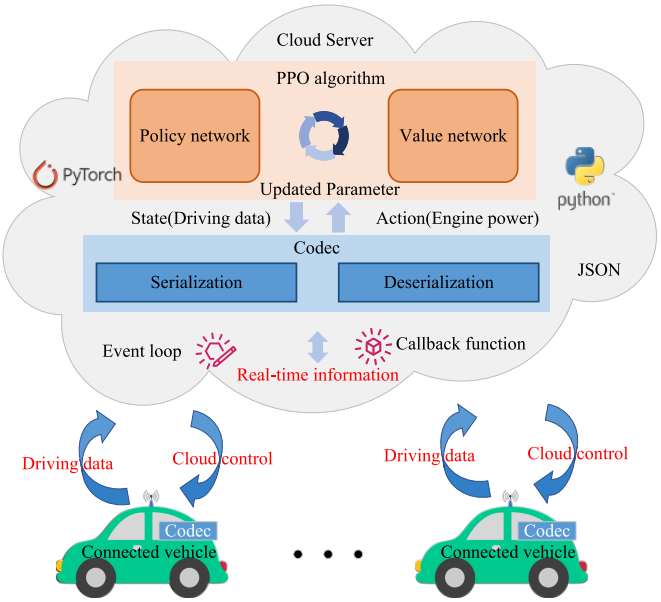

Fig. 5. Cloud computing based parallel EMS training framework

Connected vehicles continuously send real-time driving data to cloud servers during the driving process, including vehicle speed, SOC, and acceleration. After deserialization the driving data directly interacts directly with the PPO algorithm. The policy network generates action outputs (engine power) based on the state (driving data), which are serialized into JSON and returned to the connected vehicle in real time [37]. Simultaneously, the cloud collects and utilizes the driving data of the connected vehicles to continuously update the PPO algorithm. This cloud computing-based parallel EMS training framework is able to leverage the experience of multiple connected vehicles to accelerate policy learning and optimization.

This framework can be used as a generic base framework to adapt other DRL-based EMS. Data under different driving cycles can provide richer training samples. The overall training strategy is shown in Algorithm 2.

C. Cloud computing-based parallel real-time EMS

1) Improvements in parallel EMS training framework: The cloud computing-based parallerl EMS training framework can be trained in parallel interaction with multiple connected vehicles to improve the overall generalization capability. However, the framework may face practical problems during online real-time operation. The framework adopts an asynchronous mechanism to interact with multiple connected vehicles in parallel. In the cloud, the interaction between strategy and the environment, as well as strategy updates, are executed sequentially. The policy update consumes a relatively long time, which is unacceptable in real-time control. To improve cloud computing efficiency and enhance real-time performance, the strategy trained by the above framework is adopted as the initial strategy. A thread pool-based parallel real-time EMS is developed. The overall architecture is shown in Figure 6. In contrast to the parallel training framework, the real-time EMS maintains an actor network and a buffer for each remote connected vehicle, respectively. The actor network replicates the parameters of the policy network to interact with connected vehicles. A thread pool

Algorithm 2 Cloud Computing-Based Parallel EMS Training Framework.

Initialize PPO network
Initialize TCP server and bind to port
while server running **do**
  if connection request **then**
    Create echo object e, including a buffer b
    connection established
  end if
  if a connection event is triggered **then**
    Connected vehicles uploads state s
    Call back function of e with s
    Deserialize data to get s'
    Get control quantity $c = a(s')$
    Serialize c to get c'
    Send c' to client connected vehicle
    if $\exists e \in$ echo objects, e.buffer is full **then**
      Update global PPO network P using data in b
    end if
  end if
end while

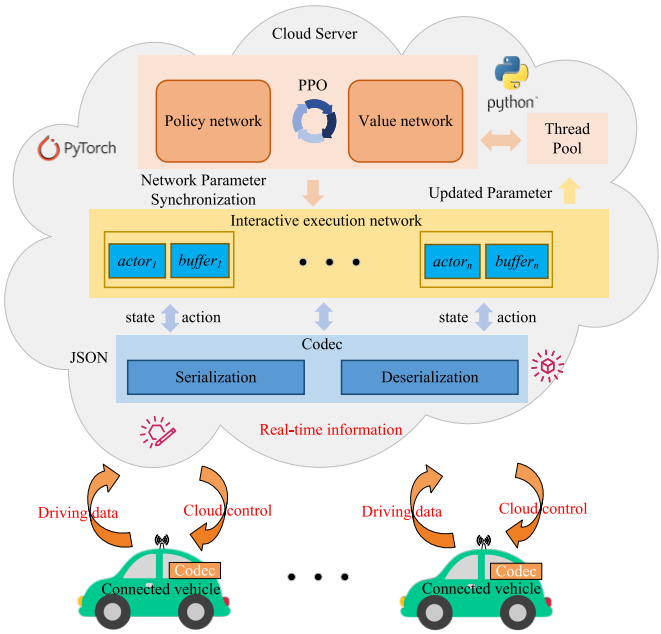

Fig. 6. Cloud computing based parallel real-time EMS

is created in the cloud for policy updates, which provides the basis for thread parallelism for subsequent algorithm updates and action execution.

2) Thread pool based real-time control: In order to parallelize the action execution and policy update in the cloud computing based EMS, this study uses the newly created threads for policy update. The interaction of states and actions, as well as access requests for newly connected vehicles, is still controlled by the asynchronous model of the main thread. Since cloud computing-based EMS can concurrently serve multiple connected vehicles, each vehicle may trigger a cloud policy update, which results in additional overhead from frequent thread creation. To solve the problem of frequent thread creation, a thread pool with a fixed number of threads is created during the initialization phase and

threads are dynamically assigned to perform policy updates. The specific algorithm is shown in Algorithm 3.

Algorithm 3 Cloud Computing-Based Parallel Real-Time EMS.

Initialize PPO network
Initialize Create ThreadPool with *maxthreads* $= n$
Initialize TCP server and bind to port
while server running **do**
  if connection request **then**
    Create echo object e, including $actor$ with parameters $\theta_{actor} \leftarrow \theta_{global}$, and *buffer*
    connection established
  end if
  if a connection event is triggered **then**
    Connected vehicles uploads state s
    Deserialize data to get s'
    Get control quantity $c = actor(s')$
    Serialize c to get c'
    Send c' to client
    if $buffer$ is full **then**
      $t \leftarrow$ FindAvailableThread($ThreadPool$)
      Assign the task to the selected thread
      acquireLock(*lock*)
      AssignTaskToThread(t, UPDATE_TASK)
      releaseLock(*lock*)
    end if
    if Thread finished **then**
      acquireLock(*lock*)
      Synchronize network parameters to the actor network
      releaseLock(*lock*)
    end if
  end if
end while
function UPDATE_TASK(*params*)
  Update global PPO network P using data in *buffer*
end function

Under the parallel real-time EMS, subsequent connected vehicles can access the entire cloud control at any time. In addition, subsequent accessed vehicles are able to directly utilize the experience of other previously accessed vehicles in similar states, which can result in improved fuel efficiency.

V. Validation results and disscussions

This section analyzes and evaluates the proposed strategy from multiple perspectives of performance through multiple comparative simulations. The effectiveness of the PPO algorithm based EMS is first verified. Then the generalization performance of the cloud computing-based EMS training framework is verified. Finally, the online validation of the parallel real-time EMS is performed. In this study, the deep reinforcement learning algorithm environment is built using Python 3.7 and Pytorch. The hyperparameters of PPO algorithm are adjusted by experience to ensure the stability and efficiency during training. The specific parameters are shown in Table II.

TABLE II
Hyperparameters

Parameters	value
Discount factor	0.99
Policy network learning rate	3e-4
Value network learning rate	3e-4
Hidden size	256
Clip factor	0.2

A. Validation of the PPO-based EMS

1) Single driving cycle simulation validation: To evaluate the performance of the PPO-based EMS, simulations are first executed under the WLTC driving cycle, commencing with initial SoC at 0.8 and 0.4. Under different initial SOC, the trajectory of SOC is shown in Figure 7. The cumulative reward is shown in Figure 8, as the episode count increases, the rewards for both scenarios tend to increase and converge at around 30 episodes. Due to the setting of the reward function, large negative rewards are frequently obtained when the SOC is lower than the set limit. Consequently, the reward curve exhibits frequent fluctuations when the initial SOC is 0.4, whereas it appears smoother with an initial SOC of 0.8.

The overall equivalent fuel consumption is illustrated in Figure 8, when the initial SOC is high, the agent tends to consume more electrical energy, resulting in lower equivalent fuel consumption. Conversely, when the initial SOC is low, the engine starts more frequently to avoid larger negative incentives, which causes an augmentation in fuel consumption. With the iterative updating of the PPO-based EMS, the equivalent fuel consumption for both cases stabilizes within a range, which shows the adaptability and robustness of the strategy.

2) Combined driving cycle simulation verification: For a more comprehensive evaluation of the performance of PPO-based EMS, combined driving cycles are further utilized for analysis. The DP-based EMS serves as the baseline strategy to compare with EMS based on DQN, DDQN, and PPO.

DQN and DDQN are classified as discrete DRL algorithms for solving problems with discrete, finite action sets. The calculation of the target Q-value by the DQN involves the utilization of the maximum action value selected by the current network, which can lead to overestimation and result in reduced algorithm performance[38]. In contrast, the DDQN algorithm separates the network for computing Q-values from the network for action selection, mitigating the issue of Q-value overestimation. As can be seen in Figure 9, DQN has a higher terminal SOC compared with the other algorithms. DDQN achieves a similar terminal SOC as DP with slightly better performance. Compared with DQN and DDQN, the PPO algorithm can handle a continuous action space and therefore provides finer and smoother control. The SOC trajectory of the PPO-based EMS is closer to that of the DP-based EMS, which confirms the performance advantage of the PPO algorithm. Table III shows the equivalent fuel consumption of the DP, PPO, DQN and DDQN strategies under the combined driving cycle. The PPO-based EMS fuel economy improves by 10.4% and 2.7% compared with the DQN and DDQN strategies, respectively.

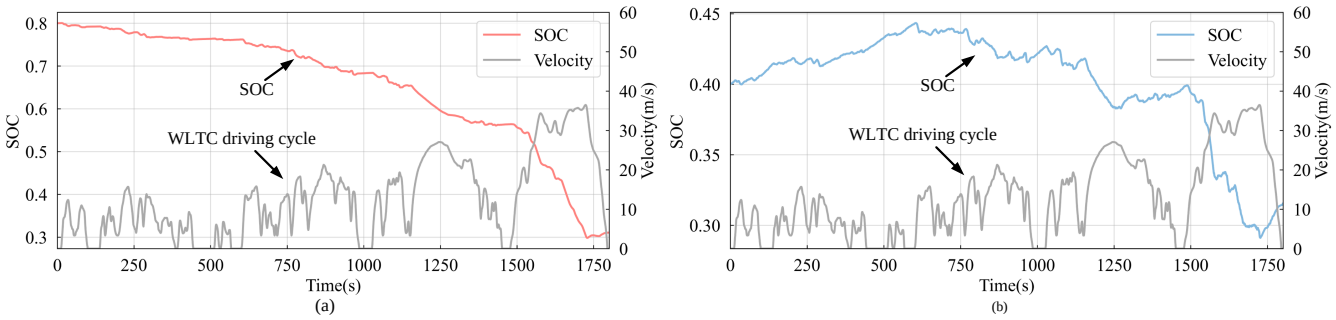

Fig. 7. SOC trajectories of PPO-based EMS at different initial SOCs under the WLTC cycle. (a) Initial SOC is 0.8. (b) Initial SOC is 0.4.

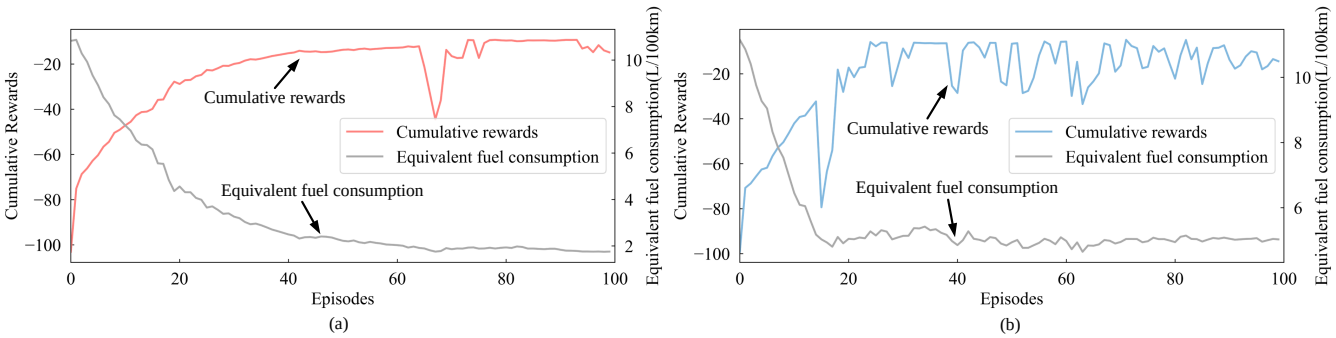

Fig. 8. Cumulative rewards and equivalent fuel consumption at different initial SOC. (a) Initial SOC is 0.8. (b) Initial SOC is 0.4.

TABLE III
COMPARISON OF ECONOMICS BETWEEN DIFFERENT EMSS.

Strategy	Initial SOC	Terminal SOC	Equivalent fuel(L/100km)	Fuel economy(%)
DQN	0.8	0.40	3.64	68.4
DDQN	0.8	0.32	3.27	76.1
PPO	0.8	0.31	3.16	78.8
DP	0.8	0.31	2.49	100

B. Validation of the cloud computing-based parallel EMS

In this section, four devices are used as shown in Figure 10. The cloud processor utilizes a computer with a 2.9GHz i5-10400 CPU core and 8GB of RAM, with PPO-based EMS deployed. The other three devices served as connected vehicle environments loaded with PHEV dynamics models and deployed with different training conditions, namely different driving cycles. Network communication between the connected vehicles and the cloud processor is conducted via the TCP protocol. The local PPO-based EMS is trained for 100 episodes each under different driving cycles, while the cloud computing-based EMS interacts with multiple connected vehicles operating under different driving cycles and trains for 100 episodes simultaneously.

1) Validation of the parallel EMS training framework: The two devices load the NEDC and UDDS driving cycles respectively. The cloud server receives and processes the state information of the two devices in parallel and updates the cloud policy. To assess the efficacy of the cloud computing parallel EMS training framework, the DP-based EMS is employed as a benchmark to compare the local PPO-based EMS single-driving cycle training (PPO_Local) with the cloud computing-based multi-driving cycle parallel training (PPO_Cloud) at initial SOC of 0.8 and 0.4, respectively.

When the initial SOC is 0.8, vehicles operating under both NEDC driving cycles and UDDS driving cycles only need to use the electric motor as much as possible to achieve lower equivalent fuel consumption. The PPO_Cloud strategy, which involves parallel interactive training in two environments, can easily learn close to optimal strategies. As depicted in Figure 11, the SOC trajectory of PPO_Cloud basically overlaps with the SOC trajectory of the DP algorithm, which is better than the single driving cycle training corresponding to the PPO algorithm.

When the initial SOC is 0.4, the connected vehicle is more likely to achieve a lower SOC under the NEDC driving cycle in the parallel EMS training framework, which leads to frequent engine involvement in the control and indirectly affects the overall strategy. Increase in equivalent fuel consumption for connected vehicles operating in the UDDS driving cycle. Comparative results of equivalent fuel consumption are shown in Figure 12. In all four scenarios, PPO_Cloud demonstrated an average fuel economy improvement of 7.7% compared to PPO_Local.

2) Verification of parallel real-time EMS: Finally, three connected vehicles with initial SOC of 0.4 are used. Two target connected vehicles are loaded with NEDC and UDDS driving cycles, respectively, while the other target connected

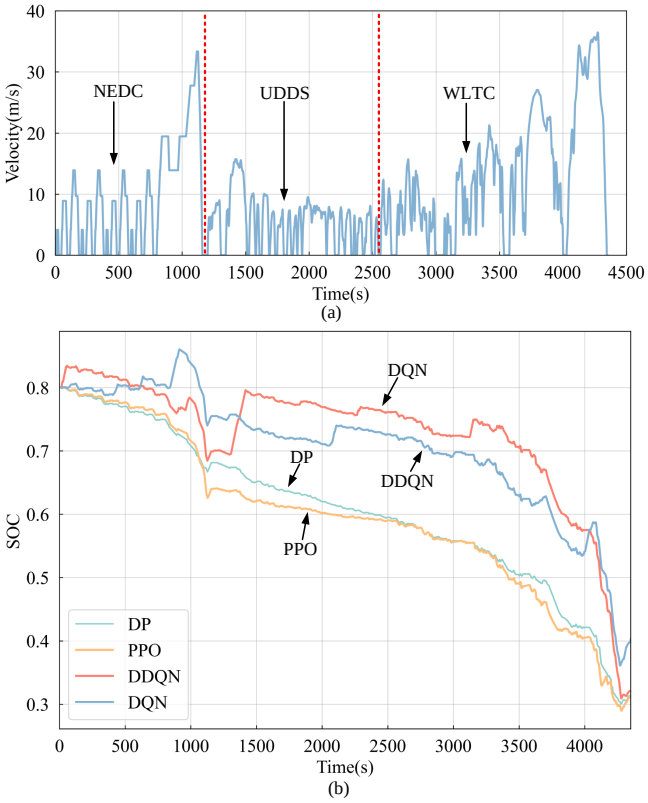

Fig. 9. Comparison of different algorithms. (a) Combined driving cycle. (b) SOC trajectories under different algorithms

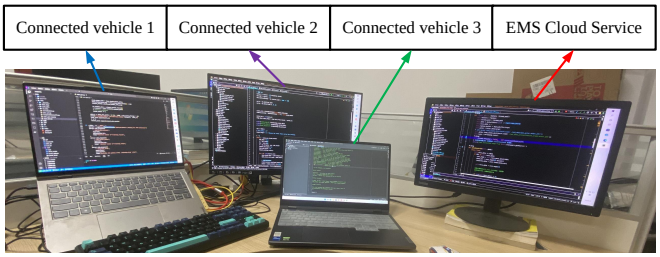

Fig. 10. Cloud strategies and multi-connected vehicle simulation testing.

vehicle is loaded with an untrained WLTC driving cycle. The parallel-trained policies are then deployed as initial policies in a parallel real-time EMS and run online for 10 episodes.

The strategy, as demonstrated by the connected vehicles running in parallel in real-time under the two trained driving cycles of NEDC and UDDS, exhibits stability, can even self-learn, and further optimize the equivalent fuel consumption, as illustrated in Figure 13.

For the connected vehicle operating under the untrained WLTC driving cycle, with relevant experience from other connected vehicles, this target vehicle can achieve 79.3% of the DP fuel economy after only 10 episodes of online learning. The specific results are shown in Table IV. The self-learning capability and applicability of the strategy are validated by the fuel economy being close to that of the local PPO-based EMS with 100 episodes of training. The SOC trajectory and power distribution of the two strategies are shown in Figure 14 and Figure 15.

To validate the real-time performance of the parallel real-time EMS, the average per-step control interaction duration of the three strategies is compared. Strategy 1 is the local

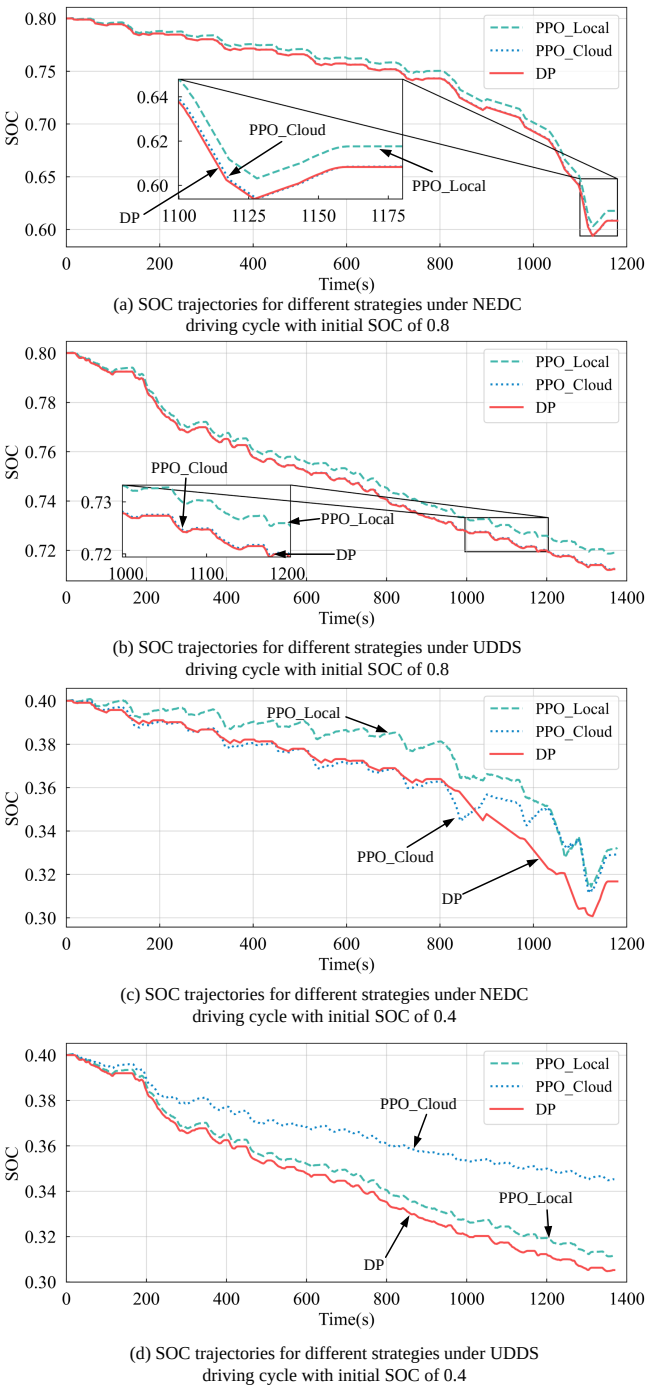

Fig. 11. Validation of parallel training results.

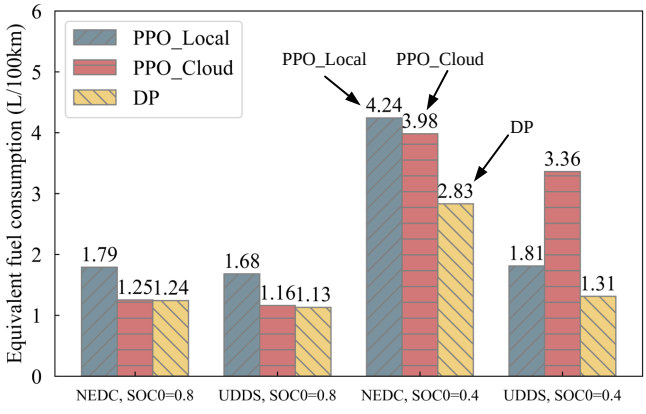

Fig. 12. Equivalent fuel consumption with different strategies.

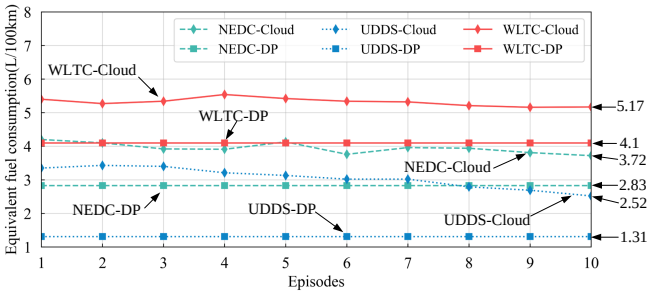

Fig. 13. Online deployment of strategies.

TABLE IV
COMPARATIVE OF EQUIVALENT FUEL ECONOMY IN WLTC.

Strategy	Equivalent fuel(L/100km)	Fuel economy(%)
PPO	5.03	81.5
Cloud	5.17	79.3
DP	4.1	100

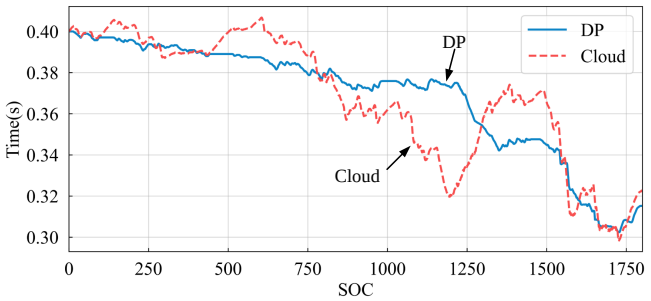

Fig. 14. SOC trajectories under different strategies.

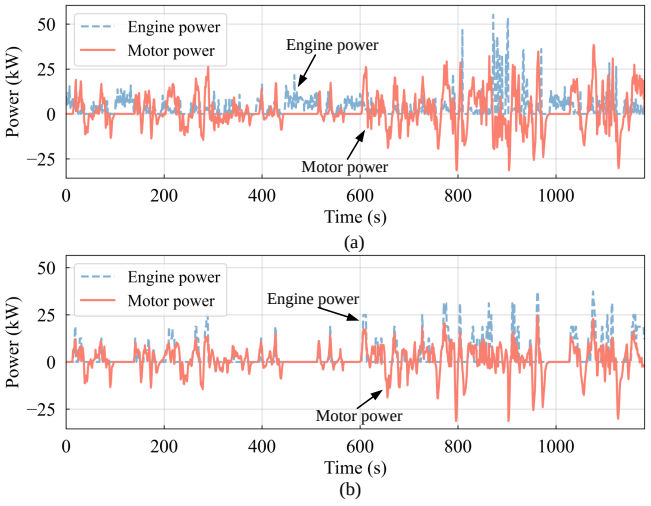

Fig. 15. Power distribution under different strategies. (a) Cloud. (b) DP.

PPO-based EMS deployed directly to the cloud server after training is completed. Strategy 2 is the cloud computing-based parallel EMS training framework. Strategy 3 is the thread pool-based parallel real-time EMS.

Figure 16 shows the average computational interaction duration per step between the connected vehicles running under different driving cycles and the cloud-based policy. For strategy 1, the average time per step is only related to the network condition, thus consuming the least amount of time. Strategy 2 contains the policy update time in addition to the control interaction as well as the network communication time. Although strategy 3 includes the time required for strategy updates, the thread pool allows the update and control to be executed concurrently in two individual threads. The thread pool-based parallel real-time EMS reduces time by 20% compared with parallel EMS training framework.

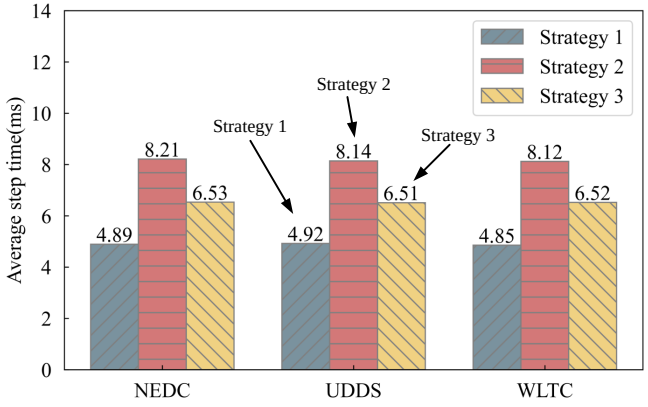

Fig. 16. Average interaction time of the three strategies.

It should be noted that this study is conducted in Python and Pytorch environments, since python has a global interpretation lock (GIL), which results in multithreading mechanism of python not effectively utilizing multiple cores of the CPU to improve performance. However, in the environment of multiple connected vehicles, there are multiple network IO requests at each time point. For this IO-intensive task, the multithreading mechanism of python still has some efficiency advantages. If the strategy is deployed industrially, the overall program can be developed in C++ and deployed to a Linux server, which makes full use of the server CPU multi-core performance.

VI. CONCLUSION

In this study, a novel cloud computing-based parallel DRL EMS for connected PHEV is proposed. The following conclusions can be drawn:

A PPO-based EMS is proposed. The convergence speed and robustness advantages of the PPO-based EMS under different initial SOC are verified through multiple driving cycle simulations. In addition, the PPO fuel economy under the combined driving cycle improve by 10.4% and 2.7% compared with the DQN and DDQN strategies, respectively.

A parallel EMS training framework based on asynchronous mechanism is proposed. By training multiple connected vehicles in parallel, the results demonstrate the effectiveness and adaptability of parallel training. In contrast to the PPO-based EMS that is trained utilizing a solitary driving

cycle, the cloud computing-based parallel EMS achieves an average improvement of about 7.7% in fuel economy under different conditions. The framework for parallel training can be used as a generalized framework to adapt to DRL-based EMS that can be empirically parallelized.

A thread pool based parallel real-time EMS is proposed based on the parallel EMS training framework. The EMS accomplishes training and computation in the cloud and provides control for multiple connected vehicles online in real time. The strategy remains stable and further optimizes the equivalent fuel consumption for connected vehicles operating under trained driving cycles. For a connected vehicle operating under untrained driving cycles, 79.3% of the fuel economy of the DP-based EMS can be achieved after only 10 episodes of online learning by utilizing similar experiences of other connected vehicles. In addition, the EMS with a thread pool has reduced the average time for computing interactions by 20%. The proposed strategy has the advantages of adaptability, continuous learning and real-time control for improved fuel efficiency.

REFERENCES

[1] M. F. M. Sabri, K. A. Danapalasingam, and M. F. Rahmat, "A review on hybrid electric vehicles architecture and energy management strategies," *Renewable and Sustainable Energy Reviews*, vol. 53, pp. 1433–1442, 2016.

[2] A. H. Ganesh and B. Xu, "A review of reinforcement learning based energy management systems for electrified powertrains: Progress, challenge, and potential solution," *Renewable and Sustainable Energy Reviews*, vol. 154, p. 111833, 2022.

[3] A. Biswas and A. Emadi, "Energy management systems for electrified powertrains: State-of-the-art review and future trends," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 6453–6467, 2019.

[4] F. Zhang, X. Hu, R. Langari, and D. Cao, "Energy management strategies of connected hevs and phevs: Recent progress and outlook," *Progress in Energy and Combustion Science*, vol. 73, pp. 235–256, 2019.

[5] D.-D. Tran, M. Vafaeipour, M. El Baghdadi, R. Barrero, J. Van Mierlo, and O. Hegazy, "Thorough state-of-the-art analysis of electric and hybrid vehicle powertrains: Topologies and integrated energy management strategies," *Renewable and Sustainable Energy Reviews*, vol. 119, p. 109596, 2020.

[6] X. Guo, T. Liu, B. Tang, X. Tang, J. Zhang, W. Tan, and S. Jin, "Transfer deep reinforcement learning-enabled energy management strategy for hybrid tracked vehicle," *IEEE Access*, vol. 8, pp. 165 837–165 848, 2020.

[7] H. Guo, X. Wang, and L. Li, "State-of-charge-constraint-based energy management strategy of plug-in hybrid electric vehicle with bus route," *Energy Conversion and Management*, vol. 199, p. 111972, 2019.

[8] T. Hofman, M. Steinbuch, R. Van Druten, and A. Serrarens, "Rule-based energy management strategies for hybrid vehicles," *International Journal of Electric and Hybrid Vehicles*, vol. 1, no. 1, pp. 71–94, 2007.

[9] Z. Chen, C. C. Mi, J. Xu, X. Gong, and C. You, "Energy management for a power-split plug-in hybrid electric vehicle based on dynamic programming and neural networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 4, pp. 1567–1580, 2014.

[10] X. Wang, H. He, F. Sun, and J. Zhang, "Application study on the dynamic programming algorithm for energy management of plug-in hybrid electric vehicles," *Energies*, vol. 8, no. 4, pp. 3225–3244, 2015.

[11] N. Robuschi, M. Salazar, N. Viscera, F. Braghin, and C. H. Onder, "Minimum-fuel energy management of a hybrid electric vehicle via iterative linear programming," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 575–14 587, 2020.

[12] C. Ma, Y. Shang, S. Jin, K. Yang, and Z. Li, "Research on real-time energy management strategy of dual motor coupled phev based on model predictive control," *IAENG International Journal of Applied Mathematics*, vol. 53, no. 1, pp. 76–85, 2023.

[13] Q. Zhou, C. Du, D. Wu, C. Huang, and F. Yan, "A tolerant sequential correction predictive energy management strategy of hybrid electric vehicles with adaptive mesh discretization," *Energy*, vol. 274, p. 127314, 2023.

[14] X. Tian, Y. Cai, X. Sun, Z. Zhu, Y. Wang, and Y. Xu, "Incorporating driving style recognition into mpc for energy management of plug-in hybrid electric buses," *IEEE Transactions on Transportation Electrification*, vol. 9, no. 1, pp. 169–181, 2022.

[15] A. Rezaei, J. B. Burl, and B. Zhou, "Estimation of the ecms equivalent factor bounds for hybrid electric vehicles," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 6, pp. 2198–2205, 2017.

[16] Q. Jiang, F. Ossart, and C. Marchand, "Comparative study of real-time hev energy management strategies," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10 875–10 888, 2017.

[17] S. Cheng, L. Li, X. Chen, S.-n. Fang, X.-y. Wang, X.-h. Wu, and W.-b. Li, "Longitudinal autonomous driving based on game theory for intelligent hybrid electric vehicles with connectivity," *Applied energy*, vol. 268, p. 115030, 2020.

[18] R. Johri and Z. Filipi, "Optimal energy management of a series hybrid vehicle with combined fuel economy and low-emission objectives," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 228, no. 12, pp. 1424–1439, 2014.

[19] X. Lin, K. Zhou, L. Mo, and H. Li, "Intelligent energy management strategy based on an improved reinforcement learning algorithm with exploration factor for a plug-in phev," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 8725–8735, 2021.

[20] D. Xu, C. Zheng, Y. Cui, S. Fu, N. Kim, and S. W. Cha, "Recent progress in learning algorithms applied in energy management of hybrid vehicles: a comprehensive review," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 10, no. 1, pp. 245–267, 2023.

[21] J. Zhou, S. Xue, Y. Xue, Y. Liao, J. Liu, and W. Zhao, "A novel energy management strategy of hybrid electric vehicle via an improved td3 deep reinforcement learning," *Energy*, vol. 224, p. 120118, 2021.

[22] R. Lian, H. Tan, J. Peng, Q. Li, and Y. Wu, "Cross-type transfer for deep reinforcement learning based hybrid electric vehicle energy management," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8367–8380, 2020.

[23] J. Wu, H. He, J. Peng, Y. Li, and Z. Li, "Continuous reinforcement learning of energy management with deep q network for a power split hybrid electric bus," *Applied energy*, vol. 222, pp. 799–811, 2018.

[24] X. Qi, Y. Luo, G. Wu, K. Boriboonsomsin, and M. Barth, "Deep reinforcement learning enabled self-learning control for energy efficient driving," *Transportation Research Part C: Emerging Technologies*, vol. 99, pp. 67–81, 2019.

[25] R. Lian, J. Peng, Y. Wu, H. Tan, and H. Zhang, "Rule-interposing deep reinforcement learning based energy management strategy for power-split hybrid electric vehicle," *Energy*, vol. 197, p. 117297, 2020.

[26] C. Zhang, T. Li, W. Cui, and N. Cui, "Proximal policy optimization based intelligent energy management for plug-in hybrid electric bus considering battery thermal characteristic," *World Electric Vehicle Journal*, vol. 14, p. 47, 2023.

[27] T. Matsushima, H. Furuta, Y. Matsuo, O. Nachum, and S. Gu, "Deployment-efficient reinforcement learning via model-based offline optimization," *arXiv preprint arXiv:2006.03647*, 2020.

[28] C. M. Martinez, X. Hu, D. Cao, E. Velenis, B. Gao, and M. Wellers, "Energy management in plug-in hybrid electric vehicles: Recent progress and a connected vehicles perspective," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, pp. 4534–4549, 2016.

[29] K. Liu, X. Jiao, C. Yang, W. Wang, C. Xiang, and W. Wang, "Event-triggered intelligent energy management strategy for plug-in hybrid electric buses based on vehicle cloud optimisation," *IET Intelligent Transport Systems*, vol. 14, no. 9, pp. 1153–1162, 2020.

[30] B. Hu and J. Li, "A deployment-efficient energy management strategy for connected hybrid electric vehicle based on offline reinforcement learning," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 9, pp. 9644–9654, 2021.

[31] Y. Zhang, H. Liu, Z. Zhang, Y. Luo, Q. Guo, and S. Liao, "Cloud computing-based real-time global optimization of battery aging and energy consumption for plug-in hybrid electric vehicles," *Journal of Power Sources*, vol. 479, p. 229069, 2020.

[32] Z. Zhang, T. Zhang, J. Hong, H. Zhang, and J. Yang, "Energy management strategy of a novel parallel electric-hydraulic hybrid electric vehicle based on deep reinforcement learning and entropy evaluation," *Journal of Cleaner Production*, vol. 403, p. 136800, 2023.

[33] H. Wang, Y. Ye, J. Zhang, and B. Xu, "A comparative study of 13 deep reinforcement learning based energy management methods for a hybrid electric vehicle," *Energy*, vol. 266, p. 126497, 2023.

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[35] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[36] D. Hu, H. Xie, K. Song, Y. Zhang, and L. Yan, "An apprenticeship-reinforcement learning scheme based on expert demonstrations for

energy management strategy of hybrid electric vehicles," *Applied Energy*, vol. 342, p. 121227, 2023.

[37] N. A. Alia, M. K. Yusof, and S. Safei, "An efficiency of native json+ for data integration." *IAENG International Journal of Computer Science*, vol. 51, no. 3, pp. 301–307, 2024.

[38] A. Mousa, "Extended-deep q-network: A functional reinforcement learning-based energy management strategy for plug-in hybrid electric vehicles," *Engineering Science and Technology, an International Journal*, vol. 43, p. 101434, 2023.