Optimal Re-Sequencing of Connected and Autonomous Electric Vehicles in Battery SOC-Aware Platooning

Shaopan Guo[1], Xiangyu Meng[1]

***Abstract*—In previous studies, vehicle platooning has been mainly formulated as an energy optimization problem. With the advent of electric vehicles, such formulations have not taken into account practical issues involved in electric vehicles, such as battery health and charging time. One of the major concerns is the imbalanced energy consumption resulting from fixed positions of vehicles in a platoon. This imbalance in energy consumption can be mitigated by changing fleet formations during a trip. This raises the question of determining the optimal sequences of vehicles at a few fixed re-sequencing locations. To solve this problem, five approaches for a deterministic environment and one reinforcement learning approach for a stochastic environment are proposed. Experimental results validate the effectiveness of our proposed approaches on real-world transportation networks in deterministic and stochastic environments, respectively.**

I. Introduction

Increasing efficiency in intelligent transport systems combined with vehicle automation technologies has made vehicle platooning strategies possible, in which several connected and autonomous vehicles (CAVs) line up and drive together at a close distance in order to achieve energy savings [1]. Researchers have investigated multiple methods for improving the energy efficiency of electric vehicle platoons. These include minimizing energy consumption by planning the speed or acceleration profile of the leading vehicle [2], [3], optimizing front-rear torque distribution [4], reducing communication energy consumption [5], and adjusting the distance between vehicles [6]. It should be noted that all the platoons discussed in these works are fixed. There are very few references to non-fixed platooning. In [7], the idea of non-fixed platooning is explored, and two heuristic algorithms, namely, "Fuel Amount Heuristic" and "Traveling Time Heuristic", are proposed to extend the driving range of a truck platoon by changing formations.

The rapid development of learning-based artificial intelligence technologies has led to the exploration of combining deep Q learning (DQL) technologies with CAVs. DQL is a subfield of machine learning that combines deep learning techniques with reinforcement learning algorithms [8]. It involves training artificial intelligence agents to learn from experience in a trial-and-error manner by interacting with an environment to maximize a reward signal [9]. Deep neural networks are used to represent the agent's policy and value function, enabling it to learn and make decisions in complex and dynamic environments [10]. DQN has several advantages over traditional reinforcement learning (RL) algorithms, including:

(i) Overcoming the limitations of high-dimensional state and action spaces by using deep neural networks to approximate the Q-value function.
(ii) Being able to learn directly from raw sensory input, allowing for more efficient and effective learning.
(iii) Handling environments with delayed rewards by utilizing experience replay and target network techniques.
(iv) Being able to handle environments with continuous state and action spaces, unlike traditional RL algorithms that are often limited to discrete spaces.

As a result of these advantages, DQN offers a promising solution to improve the efficiency, safety, and sustainability of intelligent transportation systems with CAVs. Some applications of DQL in CAVs includes: traffic signal control under mixed traffic conditions [11], jamming and eavesdropping defense [12], and driving decision-making [13].

Compared with vehicles without platooning, the vehicle fleet using the platoon control technology reduces aerodynamic drag by decreasing the safe inter-vehicle space, leading to energy saving [14], [15]. Road tests in [16], [17], [18] and fluid dynamic simulations in [14], [19], [20] have shown that platooning may save up to 10% of energy, depending on vehicle model, driving cycle, traffic speed, and platoon size [18]. There is a common thread in the above experiments: vehicles in different positions in a fleet enjoy different energy-saving benefits from platooning control. In most experiments, the energy saving of trailing vehicles is higher than that of the lead one. In an experiment carried out in [21] on a three-vehicle platoon at a 6-meter gap, the first vehicle in the platoon is able to save 4.3% of its normal fuel consumption when cruising at 85 km/h, while the other two vehicles save 10% to 14%. As the age of EVs is dawning, the need to operate EVs efficiently has turned into an important concern.

Despite the fact that platooning can reduce the energy consumption of vehicles, some EV-related issues are not taken into consideration in the traditional platooning scheme. Range anxiety has long been one of the fundamental roadblocks preventing EVs from mass deployment. The total mileage that an EV fleet can cover depends on the vehicle with the least amount of energy. In other words, the distinct energy consumption reduction rates attributed to fixed vehicle positions

[1]Shaopan Guo and Xiangyu Meng are with the Division of Electrical and Computer Engineering, Louisiana State University, Baton Rouge, LA 70803, USA xmeng5@lsu.edu, gshaop1@lsu.edu

in the traditional platoon control methods limit the driving range of an EV fleet. Another important issue caused by the distinct energy reduction rates is the imbalance of the final remaining energy of vehicles at the destination since the fleet has to wait for the vehicle with the least remaining energy to fully charge before starting a new journey. This issue is more critical for EVs than internal combustion engine vehicles due to the long recharging time [22]. For instance, it takes at least 6 hours and 9.5 hours for a Tesla Model S and Model X, respectively, to be fully charged on a Level 2 charging station, which operates on 240 volts AC [23]. Thus, the impact of waiting time at charging stations cannot be ignored. Moreover, the rapid decline in the battery charge level of the lead EV in a platoon is likely to negatively affect its battery state of health (SOH) since EV batteries degrade as they undergo more charge and discharge cycles [24].

In light of the foregoing discussions, we propose a new platooning scheme called non-fixed platooning, which allows vehicles in a fleet to change positions during a trip. This idea was explored in [7], where two heuristic algorithms, called "Fuel Amount Heuristic" and "Traveling Time Heuristic", were proposed for extending the driving range of a truck platoon by changing formations. Different from [7], we aim at balancing the final state-of-charge (SOC) of EVs by changing the formations of an EV platoon at several fixed locations during a trip, where the performance is measured by the standard deviation of EVs' final SOCs. Mathematically, this is an NP-hard combinatorial optimization problem. As far as we know, there is no algorithm that can find the optimal solution in polynomial time. Certain combinatorial optimization problems can be solved using suboptimal polynomial-time algorithms, such as dynamic programming [25], [26], [27], Hungarian method [28], and branch-and-cut algorithm [29]. However, the above mentioned algorithms are inapplicable to non-fixed platooning problems as the objective function cannot be decomposed into a sum of costs at multiple stages and the standard deviation of final SOCs can be calculated only when the trip is finished.

In this paper, the optimal re-sequencing (ORS) problem is formulated with the objective of minimizing the standard deviation of final SOCs. Platoon re-sequencing strategies are designed to solve the ORS problem in deterministic and stochastic environments, respectively. In the deterministic environment, a brutal force algorithm is presented first, which can find the optimal solution to the ORS problem with small numbers of vehicles and re-sequencing locations. To reduce the time complexity of the brutal force algorithm, several algorithms, including SOC ranking algorithm and max-min swap algorithm, are proposed. It is proved that the optimal re-sequencing strategy at the last stage should follow the SOC order, regardless of previous formations at earlier stages. As a special case, the optimal formation of a fixed platoon should follow the SOC order. In the stochastic environment, a re-sequencing algorithm is trained using deep Q networks. Compared with the Monte Carlo method used in [30], the deep Q network approach can cope with continuous observation space with SOCs of all EVs being the state variable. To evaluate the proposed algorithms, a real-world transportation route is considered.

Overall, the contributions of this paper are summarized in four aspects:

1) It is proved theoretically that the SOC-order-based formation is the optimal formation for the fixed platoon and for the vehicle fleet at the last re-sequencing location.
2) It is shown numerically that the SOC ranking algorithm may not always be the optimal solution to the ORS problem by providing a counterexample.
3) The max-min swap algorithm can obtain a near-optimal solution.

II. Problem Formulation

We begin by formulating the EV platoon re-sequencing problem. Consider a platoon with N EVs having identical dynamics, and denote vehicle i's displacement as x_i. The total distance of the trip is d, and there are M locations that allow platoon re-sequencing with $d_0, d_1, \ldots, d_{M-1}$ distance from the departure point. Particularly, the first location that allows platoon re-sequencing is at the departure point, which means that $d_0 = 0$. The time instants at which the lead vehicle of the platoon passes the re-sequencing locations are denoted as $t_0, t_1, \ldots, t_{M-1}$, respectively. The time instant at which the fleet reaches its destination is denoted as t_f. The orders of N EVs during M phases are denoted by a position change matrix $P \in \mathbb{R}^{N\times M}$, where $P_{ij} \in \{1, \ldots, N\}$ is the order of EV i at the jth platooning phase for $i = 1, \ldots, N$, and $j = 1, \ldots, M$. An example is shown in Fig. 1 with three electric vehicles - yellow EV (EV 1), purple EV (EV 2), and blue EV (EV 3). There are 3 re-sequencing phases at d_0, d_1, and d_2, respectively, and 3 platooning phases. A formation-changing policy is applied to the EVs at the re-sequencing phase, and then they keep the formation in the following platooning phase. In this example, formations change from 2-1-3 to 3-1-2 to 1-3-2, which indicates that the position change matrix is

$$P = \begin{bmatrix} 2 & 3 & 1 \\ 1 & 1 & 3 \\ 3 & 2 & 2 \end{bmatrix}.$$

Electric vehicles utilize SOC to indicate the charge level of the battery in comparison to its capacity, similar to a fuel gauge. Let δ denote the average rate of electricity usage per unit distance, Q denote the maximum battery capacity, and $\eta^j_{P_{ij}}$ denote the reduction rate of electricity usage at position P_{ij} during the jth platooning phase, where $i = 1, \ldots, N$, and $j = 1, \ldots, M$. To calculate the energy consumption of vehicle i during the jth platooning phase, we use the following equation:

$$\Delta^j_{P_{ij}} = \frac{\delta}{Q}(1 - \eta^j_{P_{ij}})[x_i(t_j) - x_i(t_{j-1})],$$

where $\Delta^j_{P_{ij}}$ represents the energy consumption of EV i at position P_{ij} during the jth platooning phase. Since the re-sequencing phase is considerably shorter than the platooning phase, and there are only a few re-sequencing phases throughout a trip, we disregard the energy cost during the re-sequencing phase. Therefore, the SOC change of EV i

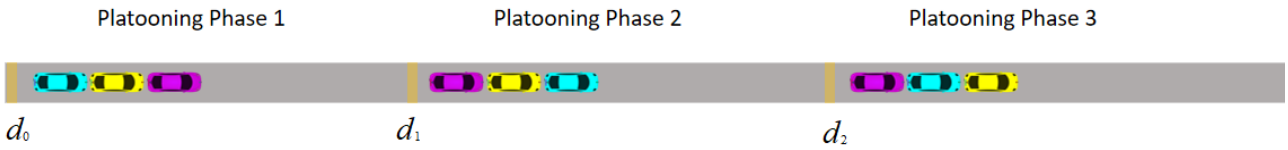

Fig. 1. Re-sequencing and platooning scheme

at position P_{ij} during the jth platooning phase is given by $\Delta^j Pij$. We can calculate the SOC of EV i at time instant t_j, denoted by $s_i(t_j)$, using the following equation:

$$s_i(t_j) = s_i(t_{j-1}) - \Delta^j_{P_{ij}}, \tag{1}$$

where $s(t_j) = \left[s_1(t_j), \ldots, s_N(t_j)\right]^T$ is a vector representing the SOC of all the EVs in the platoon at time instant t_j. Throughout this paper, we assume that the inequality $\eta^j_{P_{i_2j}} \geq \eta^j_{P_{i_1j}}$ holds for any two positions P_{i_1j} and P_{i_2j} satisfying $P_{i_2j} > P_{i_1j}$. Recall that $P_{i_2j} > P_{i_1j}$ indicates that EV i_1 is ahead of EV i_2 during the jth platooning phase. In this paper, we use standard deviation to quantify the dispersion of vehicles' final SOCs. Let $a = [a_1, \ldots, a_N]^T$ be an N-element vector. We define the mean of its elements as

$$\mu(a) = \frac{1}{N}\sum_{i=1}^{N} a_i,$$

and the variance of its elements as

$$\sigma^2(a) = \frac{1}{N}\sum_{i=1}^{N}(a_i - \mu(a))^2.$$

The root of $\sigma^2(a)$, denoted as $\sigma(a)$, is the standard deviation of the elements of vector a.

Remark 1: A well-known formula [31, Equation (1.4)] can be used to speed up the calculation of the variance:

$$\sigma^2(a) = \left[\frac{1}{N}\sum_{i=1}^{N} a_i^2\right] - \mu^2(a).$$

This formula allows for the calculation of both the mean and the variance by iterating through a given vector a only once.

The above-defined notations are summarized in Table I for future reference. The optimal re-sequencing (ORS) problem is formulated as the following optimization problem

$$\begin{aligned} \min_{P}\ & \sigma^2(s(t_f)) \\ \text{s.t. } & (1), \\ & s_i(t_f) \geq 0, \\ & P_{ij} \in \{1, \ldots, N\}, \\ & P_{i_1j} \neq P_{i_2j} \text{ for } i_1 \neq i_2, i_1, i_2 \in \{1, \ldots, N\}, \\ & i = 1, \ldots, N, j = 1, \ldots, M. \end{aligned} \tag{2}$$

III. Re-sequencing Algorithms for Deterministic Environmental Conditions

Utilizing the historic data, we can use the statistical knowledge to get the electricity usage of an EV at position i during the jth platooning phase, Δ^j_i. Then we can build an electricity usage matrix $\Delta \triangleq [\Delta^j_i]$, where the element at its ith row and jth column is Δ^j_i.

Notation	Definition ($i = 1, \ldots, N, j = 1, \ldots, M$)
$x_i(t)$	Vehicle i's displacement at time t
d_{j-1}	The distance from the jth re-sequencing place to the departure point
t_{j-1}	The time instant at which the lead vehicle of the platoon passes the jth re-sequencing place
t_f	The time instant at which the fleet reaches its destination
P_{ij}	Position of EV i during the jth platooning phase
Q	The battery maximum capacity
δ	The average rate of electricity usage per unit distance
$s_i(t)$	SOC of EV i at time t
η^j_i	The electricity usage reduction rate at position i during the jth platooning phase
Δ^j_i	The SOC change of an EV at position i during jth platooning phase
$\sigma(a)$	The standard deviation of the elements in the vector a.

TABLE I
Notations

Before proceeding, let us introduce a lemma, which is crucial in developing the first result in this paper.

Lemma 1: [32, Fact 2.11.37] Let $y \triangleq [y_1, \ldots, y_N]$ be an N-dimensional real vector. We rearrange $y_1, \ldots, y_N$ as $y_{[1]}, \ldots, y_{[N]}$ such that $y_{[1]} \geq \cdots \geq y_{[N]}$. The following inequalities hold

$$\begin{aligned} \frac{\max\left\{y_{[1]} - \mu(y), \mu(y) - y_{[N]}\right\}}{\sqrt{N-1}} &\leq \sigma(y) \\ &\leq \sqrt{N-1}\min\left\{y_{[1]} - \mu(y), \mu(y) - y_{[N]}\right\}, \end{aligned} \tag{3}$$

where

$$\mu(y) \triangleq \frac{1}{N}\sum_{i=1}^{N} y_i.$$

By using Lemma 1, we can determine the upper and lower bounds of the standard deviation of SOCs in deterministic environments.

Theorem 1: The standard derivation of the final SOCs of

all EVs can be bounded as

$$0 \leq \sigma(s(t_f)) \leq \sqrt{N-1}\min\{\lambda_1, \lambda_2\},$$

where

$$\lambda_1 = s_{[1]}(t_0) - \sum_{j=1}^{M} \Delta_N^j - \overline{s}(t_f),$$

$$\lambda_2 = \overline{s}(t_f) - s_{[N]}(t_0) + \sum_{j=1}^{M} \Delta_1^j,$$

and the average value of final SOCs

$$\overline{s}(t_f) = \frac{1}{N}\sum_{i=1}^{N} s_i(t_0) - \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} \Delta_i^j. \tag{4}$$

Proof: Given the initial SOCs $s(t_0)$ and the electricity usage matrix $\Delta = [\Delta_i^j]$, the average value of the final SOC $\overline{s}(t_f)$ is a constant. The maximum possible final SOC of any EV can be obtained by always placing the EV with the highest initial SOC at the position with the lowest energy consumption, that is, position N of the platoon. Similarly, the minimum possible final SOC of any EV can be obtained by always placing the EV with the lowest initial SOC at the position with the highest energy consumption, that is, position 1 of the platoon. Therefore, we have

$$s_{[1]}(t_f) = s_{[1]}(t_0) - \sum_{j=1}^{M} \Delta_N^j$$

and

$$s_{[N]}(t_f) = s_{[N]}(t_0) - \sum_{j=1}^{M} \Delta_1^j.$$

In light of Lemma 1, the upper bound of the standard deviation of the final SOCs can be obtained.

The perfect case is that all EVs have the same SOC at the destination, which indicates that the lower bound of $\sigma(s(t_f))$ is 0. ■

Here we present five re-sequencing algorithms for minimizing the standard deviation of the final SOCs in deterministic environments.

1) Brutal force algorithm (A1): A naive algorithm for the ORS problem is to enumerate all possible position matrices and find the one with the smallest standard deviation of the final SOCs. For an ORS problem with M re-sequencing phases and N EVs, there are $N!$ different formations at each re-sequencing phase, and the total number of possible formation series is $N!^M$. Algorithm A1 is capable of finding the optimal solution for the ORS problem of any N and M in theory. The disadvantage of algorithm A1 is that it must traverse all possible solutions. When N and M are large, the number of possible solutions becomes prohibitively large. For instance, the total number of possible solutions to the ORS problem is $7,962,624$ when $N = 4$ and $M = 5$. The most straightforward method to minimize $\sigma(s(t_f))$ is the linear traversal technique, as described in Section 6.1 of [33]. This algorithm does not require a sorting function to be called. However, this approach requires evaluating the objective function of the ORS problem (2) a total of $N!^M$ times.

2) Modified brutal force algorithm (A2): In the following, we will try to reduce the number of goal function calls of algorithm A1 by utilizing the following lemma.

Lemma 2: [32, Fact 2.12.8] Consider real numbers $y_1, \ldots, y_N$ and $z_1, \ldots, z_N$. Then,

$$\sum_{i=1}^{N} y_{[i]} z_{[N-i+1]} \leq \sum_{i=1}^{N} y_i z_i \leq \sum_{i=1}^{N} y_{[i]} z_{[i]}.$$

Before proceeding, let us define a SOC-order-based map

$$s(t) = \begin{bmatrix} s_1(t) \\ \vdots \\ s_N(t) \end{bmatrix} \longmapsto \hat{s}(t) = \begin{bmatrix} s_{[1]}(t) \\ \vdots \\ s_{[N]}(t) \end{bmatrix}$$

and the corresponding index mapping is defined as

$$\pi_t : \{1, \ldots, N\} \to \{1, \ldots, N\},$$

whose arguments are vehicle labels and outputs are the SOC orders of vehicles at time t. Here, $\pi_t(i)$ represents the position of EV i in $\hat{s}$ at time t. Since π_t is a bijection mapping, $\pi_t^{-1}(i)$ represents the EV with the ith highest SOC among EVs at time t. We refer to this special formation where EV i is at $\pi_t(i)$th position in the platoon for $i = 1, \ldots, N$, as the SOC-order-based formation. In this formation, the ordinal position of the EVs' available SOCs corresponds to the energy consumption order of positions in the platoon.

Now we are ready to present a modified brutal force algorithm to reduce the time complexity of algorithm A1. Given the SOCs $s(t_{M-1})$ of all EVs at t_{M-1}, the formation during the last platooning phase can be optimized using the following theorem.

Theorem 2: To minimize the standard deviation of the final SOCs of all EVs, it is optimal to make the vehicle fleet follow the SOC-order-based formation at the last re-sequencing phase, no matter what the formations in the previous re-sequencing phases are.

Proof: Suppose that SOCs of EVs at t_{M-1} are $s_i(t_{M-1})$ for $i = 1, \ldots, N$. The final SOCs can be calculated as

$$s_i(t_f) = s_i(t_{M-1}) - \Delta_{P_{iM}}^M. \tag{5}$$

Then, we have

$$N\sigma^2(s(t_f)) = \sum_{i=1}^{N} \left[s_i(t_{M-1}) - \Delta_{P_{iM}}^M - \mu(s(t_f))\right]^2.$$

Note that

$$\begin{aligned} &\sum_{i=1}^{N} \left[s_i(t_{M-1}) - \Delta_{P_{iM}}^M - \overline{s}(t_f)\right]^2 \\ &= \sum_{i=1}^{N} s_i^2(t_{M-1}) + \sum_{i=1}^{N} (\Delta_{P_{iM}}^M)^2 + N\overline{s}^2(t_f) \\ &\quad - 2\overline{s}(t_f)\sum_{i=1}^{N} s_i(t_{M-1}) + 2\overline{s}(t_f)\sum_{i=1}^{N} \Delta_{P_{iM}}^M \\ &\quad - 2\sum_{i=1}^{N} s_i(t_{M-1})\Delta_{P_{iM}}^M. \end{aligned} \tag{6}$$

Note that $\overline{s}(t_f)$ and Δ_i^M for $i = 1, \ldots, N$ are constant. Given $s(t_{M-1})$, the first 5 terms in (6) are constant. To minimize $\sigma(s(t_f))$, it is equivalent to maximizing the last term of the above equation by optimizing the last column of position change matrix, $P(:, M)$, that is,

$$\max_{P(:,M)} \sum_{i=1}^{N} s_i(t_{M-1}) \Delta_{P_{iM}}^M. \tag{7}$$

According to the second inequality in Lemma 2, (7) is maximized if the ith largest element of $s(t_{M-1})$ matches with the ith largest element in the Mth column of Δ, that is, $P_{iM} = \pi_{t_{M-1}}(i)$. In other words, the platoon should choose the SOC-order-based formation at the last re-sequencing phase for minimizing the standard deviation of the final SOCs. ■

Using the above theorem, the brutal force algorithm A1 can be modified by making EVs follow the SOC-order-based formation during the last platooning phase. As the formation at the last re-sequencing phase is determined by SOC orders according to Theorem 2, only the first $M-1$ columns of the position change matrix need to be considered. The total number of possible combinations of the first $M-1$ columns of P is $N!^{M-1}$. The number of times of algorithm A2 for calculating the standard deviation is $N!^{M-1}$. We can apply linear traversal in a similar way to algorithm A1 to find the solution with the smallest standard deviation of the final SOCs. By using $N!^{M-1}$ goal function calls and $N!^{M-1}$ sorting function calls, we can obtain the desired solution. Algorithm A2 differs from A1 in that it requires significantly fewer goal function calls. For instance, the total number of feasible solutions to the ORS problem using algorithm A2 is 331, 776 when $N = 4$ and $M = 5$. However, algorithm A2 is still not a polynomial time algorithm.

3) Modified fixed platooning algorithm (A3): The fixed platoon can be seen as a special case of the unfixed platoon with one re-sequencing phase at the departure point and one platooning phase. This leads to the corollary below directly following from Theorem 2.

Corollary 1: The optimal formation for the fixed platoon is the SOC-order-based formation.

Algorithm A3 only necessitates a single call to both the goal function and sorting function.

4) SOC ranking algorithm (A4): Inspired by Theorem 2, one may wonder if the SOC-order-based re-sequencing at all phases is optimal. This simplifies the re-sequencing phase, reducing the ORS problem to a sorting problem M times. With algorithm A4, the goal function only needs to be evaluated once at the end of the trip.

Despite its simplicity, this algorithm is not guaranteed to be an optimal solution in all cases. Let us consider the case of a 2-EV platoon in a trip that consists of 2 re-sequencing phases. The electricity usage per unit distance is $\delta = 340$ Wh/mile, and the maximum battery capacity is $Q = 100$ kWh. The vehicle fleet is allowed to change formations at the distances $d_0 = 0$ mile and $d_1 = 33$ mile. The total distance of the trip is 110 miles. The electricity reduction rates are $\eta_1^1 = 10\%$, $\eta_1^2 = 55\%$, $\eta_2^1 = 5\%$, and $\eta_2^2 = 62\%$, respectively. At the start of the trip, EV 1 has a SOC of 80% and EV 2 has a SOC of 70%. If we place EV 1 at the lead position since it has higher SOC, and place EV 2 at the trailing position, the SOC of EV 1 and EV 2 is 70% and 65% when they enter the second re-sequencing phase, respectively, and thus this formation should be maintained during the platooning phase based on Theorem 2. Final SOCs for EVs 1 and 2 are 45% and 55%, respectively, and the standard deviation of the final SOCs is 0.05. However, if we make EV 2 the lead vehicle and EV 1 the trailing one in the first platooning phase, as they enter the second re-sequencing phase, EV 1 and EV 2 have SOCs of 75% and 60%, respectively. Thus, in the second platooning phase, EV 1 should become the lead vehicle and EV 2 the trailing one according to Theorem 2. Finally, both EVs have the same SOC of 50%, and the standard deviation of the final SOCs is 0, which is smaller than that of the previous re-sequencing plan. Therefore, the strategy that high SOC EVs are always placed at the lead positions at re-sequencing phases is not always optimal for minimizing standard deviations.

5) Max-min swap algorithm (A5): We continue our quest to achieve a better trade-off between performance and computational complexity. The following corollary shows that the standard deviation of the final SOCs can be bounded for any given position change matrix P and initial SOCs.

Corollary 2: Let $s(t_f) = [s_1(t_f), \ldots, s_N(t_f)]$ be the final SOCs of an N-EV platoon with position change matrix P. Then its standard deviation $\sigma(s(t_f))$ satisfies:

$$\frac{\max\left\{s_{[1]}(t_f) - \overline{s}(t_f), \overline{s}(t_f) - s_{[N]}(t_f)\right\}}{\sqrt{N-1}} \leq \sigma(s(t_f))$$
$$\leq \sqrt{N-1} \min\left\{s_{[1]}(t_f) - \overline{s}(t_f), \overline{s}(t_f) - s_{[N]}(t_f)\right\}. \tag{8}$$

Proof: Corollary 2 follows directly from Lemma 1. ■

Inspired by Corollary 2, decreasing the gap between the maximum and minimum final SOCs seems to be an effective way of minimizing the standard deviation of the final SOCs. Therefore, a new algorithm called max-min swap based on this idea is proposed.

The algorithm begins with an arbitrary initial position change matrix P^0. According to P^0, the final SOCs of all EVs can be calculated as

$$s_i(t_f) = s_i(t_0) - \sum_{j=1}^{M} \Delta_{P_{ij}^0}^j, \tag{9}$$

and the standard deviation of the final SOCs $\sigma^0 = \sigma(s(t_f))$. The indices of EVs with the highest and lowest final SOCs can be denoted as $\pi_{t_f}^{-1}(1)$ and $\pi_{t_f}^{-1}(N)$, that is,

$$s_{\pi_{t_f}^{-1}(1)}(t_f) \geq s_i(t_f), \text{ and } s_{\pi_{t_f}^{-1}(N)}(t_f) \leq s_i(t_f),$$

for $i \in \{1, \ldots, N\}$. Then for each column j in the first $M-1$ columns with the two elements in rows $\pi_{t_f}^{-1}(1)$ and $\pi_{t_f}^{-1}(N)$ such that

$$P^0_{\pi_{t_f}^{-1}(1)j} > P^0_{\pi_{t_f}^{-1}(N)j},$$

swap the values of $P^0_{\pi_{t_f}^{-1}(1)j}$ and $P^0_{\pi_{t_f}^{-1}(N)j}$ in P^0 while other columns unchanged. The last column is determined based on SOC-order-based formation. This new position change matrix

and its corresponding standard deviation of the final SOCs are denoted as P_j^1 and σ_j^1, respectively. If

$$\min_j \sigma_j^1 < \sigma^0$$

then a better re-sequencing strategy $P_{j^*}^1$ is found, where

$$j^* = \arg\min_j \sigma_j^1.$$

Let $\sigma^1 = \sigma_{j^*}^1$ denote the best performance after 1 swap, and the corresponding position change matrix P^1. Repeat the above process until the performance cannot be improved or the number of iterations exceeds the predetermined upper bound ζ. Details of the max-min swap algorithm are summarized as pseudo-code in Algorithm A5.

Algorithm A5 Max-min swap algorithm

Input: $N, M, s(t_0), \Delta, P^0, \zeta$
Output: P
  $P = P^0$
  stopUpdate $= False$
  $t = T$
  $\bar{\zeta} = \zeta$
  while !stopUpdate and $\bar{\zeta} \geq 0$ **do**
    stopUpdate $= True$
    Calculate final SOCs of EVs, $s_i(t_f)$ $(i = 1, \ldots, N)$, according to (9).
    Calculate the standard deviation of $s(t_f)$, and give this value to σ^0.
    Find the largest and smallest elements of $s(t_f)$ and record their indexes as $\pi_{t_f}^{-1}(1)$ and $\pi_{t_f}^{-1}(N)$, respectively.
    Create an empty set $\mathcal{J}$.
    for $j \in \{1, \ldots, M-1\}$ **do**
      if $P^0_{\pi_{t_f}^{-1}(1)j} > P^0_{\pi_{t_f}^{-1}(N)j}$ **then**
        Add j into set $\mathcal{J}$.
      end if
    end for
    for $j \in \mathcal{J}$ **do**
      Swap the values of $P^0_{\pi_{t_f}^{-1}(1)j}$ and $P^0_{\pi_{t_f}^{-1}(N)j}$, and record the position change matrix after swapping as P_j^1.
      Calculate the standard deviation of final SOCs following P_j^1, and record it as σ_j^1
      if $\sigma_j^1 < \sigma^0$ **then**
        stopUpdate $= False$
        $P \leftarrow P_j^1$
        $\sigma^0 \leftarrow \sigma_j^1$
      end if
    end for
    $P^0 \leftarrow P$
    $\bar{\zeta} = \bar{\zeta} - 1$
  end while
  return P

The following corollary describes the convergence property of algorithm A5.

Corollary 3: The sequence $\{\sigma^0, \sigma^1, \ldots\}$ converges.

Proof: By construction, the sequence $\{\sigma^0, \sigma^1, \ldots\}$ is monotonically decreasing, and it is lowered bounded. According to the monotone convergence theorem [34, Theorem 11.28], $\{\sigma^0, \sigma^1, \ldots\}$ converges. ■

The maximum numbers of goal function calls and sorting function calls of algorithm A5 are both σM. In practice, algorithm A5 typically finds a suboptimal solution within a short period of time, albeit no global optimality guarantees.

Table II summarizes the aforementioned algorithms including the number of goal function calls and the number of sorting function calls. From this table, we can see that algorithms A1 and A2 can find the optimal solution, even though the number of goal function calls could be very large. As for algorithms A3-A5, they can find a suboptimal solution using less number of goal function calls than algorithms A1 and A2. However, global optimality can not be guaranteed.

Alg.	The number of goal function calls	The number of sorting function calls	Optimal Solution
A1	$N!^M$	0	Yes
A2	$N!^{M-1}$	$N!^{M-1}$	Yes
A3	1	1	No
A4	1	M	No
A5	no more than ζM	no more than ζM	No

TABLE II
COMPARISON BETWEEN DIFFERENT ALGORITHMS

IV. Re-Sequencing Algorithm with Stochastic Environment Conditions

In certain situations, there can be significant alterations in traffic conditions, such as an unforeseen heavy rainfall, which renders Δ not credible for depicting the changes in SOCs of EVs. Therefore, only algorithms A3 and A4 proposed for the deterministic case are applicable to the stochastic environment. To handle the randomness, deep Q network learning will be adopted.

The deep Q network (DQN) learning method is an online, off-policy reinforcement learning algorithm that operates without a model [8]. DQN agents can be trained in environments with continuous and discrete observation space $\mathcal{S}$, and discrete action space $\mathcal{A}$ [10]. The hybrid observation space belongs to $\mathcal{S} \in [0,1]^N \times \{0, 1, \ldots, M\}$ with dimensions $N+1$, where the first N observed states are SOCs of EVs, and the last observed state is the number of remaining locations that allows re-sequencing. The action space $\mathcal{A}$ contains all possible combinations of N EVs. The objective is to determine the optimal policy based on the observed state information. Especially, $S(N+1) = 0$ if the platoon reaches its destination. Take the platoon in Fig. 1 as an example. Assume that the SOCs of EVs at t_1 are $s_1(t_1) = 90\%$, $s_2(t_1) = 85\%$, and $s_3(t_1) = 80\%$. There are two reaming locations that allow re-sequencing. Thus, the observed states at t_1 is $S_{t_1} = [90\%, 85\%, 80\%, 2]$. To balance between exploitation and exploration, the ε-greedy method is used for the action selection, which means that the action with the greatest expected long-term return is selected with probability $1-\varepsilon$, and, with probability ε, a random action is selected from all actions with equal probability. Until the next location for re-sequencing is reached, electric vehicles

Algorithm A6 Deep Q learning method

Initialization:

Initialize the critic $Q(S, A)$ with random parameter values ϕ, and initialize the target critic parameters ϕ_t with the same values: $\phi_t = \phi$.

Initialize parameters ε, $\mathcal{M}$, and τ.

For each episode

for each training time step **do**

if $S(N+1)$ is 1 **then**

A is the SOC-order-based formation

else

For the current observation S, select a random formation A for the platoon with probability ε. With probability $1-\varepsilon$, select the formation for which the target critic value function is greatest.

end if

Make the platoon drive with the formation A until it enters the next re-sequencing phase.

Record the SOCs at the next re-sequencing phase as the first N states of S', *i.e.*. $S'(1:N)$, and the number of remaining locations that allowed re-sequencing as the last state of S', *i.e.*, $S'(N+1)$.

if $S'(N+1)$ is 0 **then**

$R \leftarrow -100\text{In}(N\sigma^2(S'(1:N)))$.

else

$R \leftarrow 0$.

end if

Store the experience (S, A, R, S') in the experience buffer.

Sample a random mini-batch of $\mathcal{M}$ experiences (S_i, A_i, R_i, S_i') from the experience buffer.

if $S_i'(N+1)$ is 0 **then**

Set the value function f_i according to (11).

else

Set the value function f_i according to (12).

end if

Update the critic parameters by one-step minimization of the loss function (10).

Update the target critic parameters according to (13).

end for

follow the formation corresponding to the selected action. When the vehicle fleet reaches the last re-sequencing location, the SOC-order-based formation is selected and followed by the vehicle fleet until it reaches the destination, which constitutes one episode. The reward is $-100\ln(\sigma(s(t_f)))$ when the fleet reaches the destination, and 0 otherwise. Learning from batches of consecutive samples causes a big problem called correlated samples, which results in inefficient learning. Experience replay is a technique that can help solve this problem [8], where we store the current state S, the selected action A, the reward R, and the next state S' in an experience buffer. Then we can train the network on a random mini-batch of $\mathcal{M}$ experiences from the experience buffer. To stabilize the learning process, a DQN agent maintains two function approximators with the same structure, called critic $Q(S, A|\phi)$ and target critic $Q_t(S, A|\phi_t)$ [8]. Here ϕ and ϕ_t are parameters of the critic and the target critic, respectively. The critic takes observation $S \in \mathcal{S}$ and action $A \in \mathcal{A}$ as inputs and returns the corresponding expectation of the long-term reward. Let

$$A^* = \begin{cases} \text{SOC-order-based formation, if } S(N+1) = 1, \\ \max\limits_{A} Q_t(S, A|\phi_t), \text{ otherwise.} \end{cases}$$

In order to train this critic, we define the following loss function:

$$L = \frac{1}{\mathcal{M}} \sum_{i=1}^{\mathcal{M}} (f_i - Q(S_i, A_i|\phi))^2, \tag{10}$$

where

$$f_i = -100\ln(N\sigma^2(s(t_f))), \tag{11}$$

if $S_i'(N+1)$ is 0, and

$$f_i = Q_t(S_i', A^*|\phi_t), \tag{12}$$

otherwise. Then we update the critic parameters by one-step minimization of the loss function L. We update the target parameters at every time step using the following equation:

$$\phi_t = \tau\phi + (1-\tau)\phi_t, \tag{13}$$

where $\tau \in (0, 1)$ is the smoothing factor. Despite being designed for the ORS problem in a stochastic environment, the deep Q learning method described here can also be applied in deterministic environments. More details can be found in pseudo-code Algorithm A6.

V. Simulation

The proposed re-sequencing algorithms are evaluated in this section. The performance metric is the standard deviation of the final SOCs with given initial SOCs and environmental conditions. To simulate the algorithms in a more realistic environment, we consider a vehicle fleet with 4 fully charged Tesla automobiles departing from Baton Rouge to Mobile as shown in Fig. 2. During the trip, the vehicle platoon is allowed

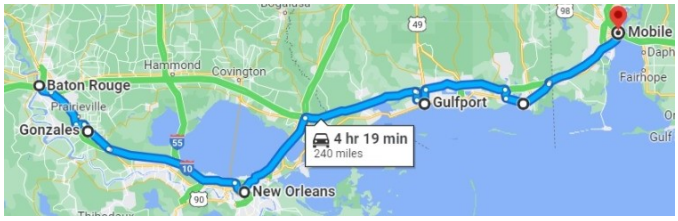

Fig. 2. A trip from Baton Rouge to Mobile

to change formation at Baton Rouge ($d_0 = 0$ miles), Gonzales ($d_1 = 24.2$ miles), New Orleans ($d_2 = 81.6$ miles), Gulfport ($d_3 = 159.1$ miles), and Pascagoula ($d_4 = 200$ miles). The total distance is 240 miles. Tesla battery packs are made up of thousands of 18650, 2170, and 4680 battery cells, which range in charge from 3400 mAh to 5000 mAh. These cells, when connected, have a total storage capacity of 85 kWh.

A. Simulation with deterministic environment conditions

In this example, the re-sequencing algorithms are tested in a deterministic environment. On average, Tesla uses 34 kWh of electricity per 100 miles, which corresponds to $\delta = 340$ Wh/mile. The electricity usage reduction rates η_i^j are borrowed from the result in [21], that is, $\eta_1^j = 0.043$, $\eta_2^j = 0.1$, $\eta_3^j = 0.14$, and $\eta_4^j = 0.14$, for $j = 1, \ldots, 5$. Thus, the electricity usage matrix Δ is obtained as

$$\Delta = \begin{bmatrix} 0.1302 & 0.1334 & 0.2522 & 0.1868 & 0.0787 \\ 0.1224 & 0.1255 & 0.2372 & 0.1756 & 0.0741 \\ 0.1170 & 0.1199 & 0.2266 & 0.1678 & 0.0708 \\ 0.1170 & 0.1199 & 0.2266 & 0.1678 & 0.0708 \end{bmatrix}.$$

A 4-layer neural network is designed to train algorithm A6. The neural network consists of one input layer, one output layer, and two hidden layers, as shown in Fig. 3. Each hidden layer has 12 neurons with the rectified linear

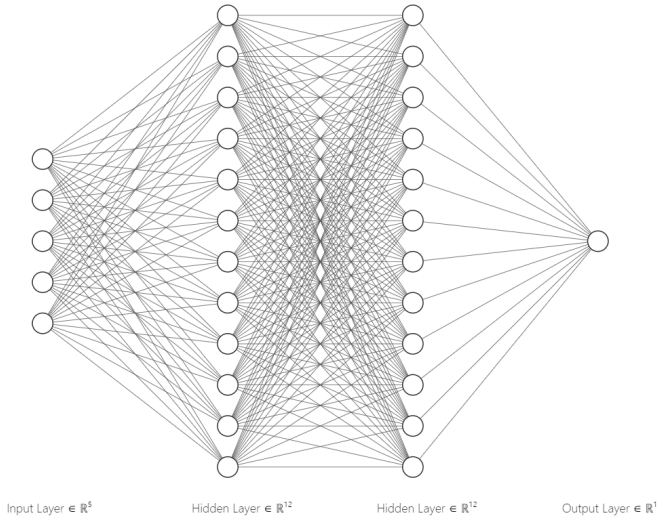

Fig. 3. 4-layer deep Q learning network

activation function, whose weights and bias are initialized with the Glorot initializer [9] and zeros, respectively. We set the probability threshold $\varepsilon = 0.1$, mini-batch size $\mathcal{M} = 256$, and smoothing factor $\tau = 0.001$. After 18,000 episodes of training, algorithm A6 can generate a re-sequencing policy whose average reward with the window length 5 is 650.9, which means that the standard deviation is 0.001492. In Fig. 4, we see how the episode reward evolves as the training progresses. At the beginning of each episode, episode Q_0 represents the long-term reward estimated from the initial observation of the environment. Episode reward is the total reward the agent receives at the end of every episode. Average reward refers to the average value of the last 5 episode rewards. The training ends when the number of episodes reaches $25,000$. From Fig. 4, we can observe that the episode reward still fluctuates widely. It is because the accurate estimation for a state-action pair's Q value requires testing it infinite times, which requires a much larger number of episodes to do so. To enable a performance comparison between the algorithms presented in this paper and the Monte Carlo method used in [30], we also

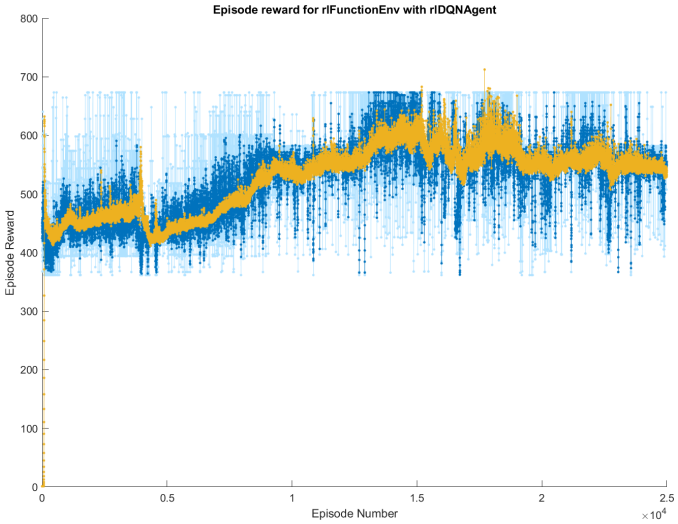

Fig. 4. Training process of algorithm A6

train a resequencing algorithm using the Monte Carlo method described in [30] under the same conditions, i.e., using the same electricity usage matrix Δ, probability threshold ϵ, and number of episodes. To simplify the discussion, we refer to the resequencing algorithm based on the Monte Carlo method described in [30] as Algorithm A7. We also denote the position change matrix obtained from Algorithm A7 as P_7.

Setting the initial position change matrix

$$P^0 = \begin{bmatrix} 1 & 1 & 1 & 1 & 4 \\ 2 & 2 & 2 & 2 & 3 \\ 3 & 3 & 3 & 3 & 2 \\ 4 & 4 & 4 & 4 & 1 \end{bmatrix}$$

for algorithm A5, then position change matrices found by algorithms A1-A6 are listed below:

$$P_1 = \begin{bmatrix} 1 & 2 & 2 & 3 & 3 \\ 2 & 1 & 3 & 2 & 4 \\ 3 & 3 & 1 & 4 & 2 \\ 4 & 4 & 4 & 1 & 1 \end{bmatrix}, \quad P_2 = \begin{bmatrix} 1 & 2 & 2 & 3 & 4 \\ 2 & 1 & 3 & 2 & 3 \\ 3 & 3 & 1 & 4 & 2 \\ 4 & 4 & 4 & 1 & 1 \end{bmatrix},$$

$$P_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 1 & 4 & 1 & 4 & 1 \\ 2 & 3 & 2 & 3 & 2 \\ 3 & 1 & 4 & 1 & 4 \\ 4 & 2 & 3 & 2 & 3 \end{bmatrix},$$

$$P_5 = \begin{bmatrix} 4 & 4 & 4 & 1 & 1 \\ 2 & 2 & 2 & 2 & 4 \\ 1 & 1 & 3 & 3 & 3 \\ 3 & 3 & 1 & 4 & 2 \end{bmatrix}, \quad P_6 = \begin{bmatrix} 1 & 1 & 1 & 1 & 3 \\ 2 & 2 & 2 & 2 & 4 \\ 3 & 3 & 3 & 3 & 2 \\ 4 & 4 & 4 & 4 & 1 \end{bmatrix},$$

$$P_7 = \begin{bmatrix} 3 & 3 & 2 & 1 & 1 \\ 2 & 2 & 4 & 2 & 2 \\ 1 & 4 & 1 & 3 & 4 \\ 4 & 1 & 3 & 4 & 3 \end{bmatrix}.$$

Table III displays the standard deviations of the final SOCs and the running time for algorithm A1-A6 to find a solution, when executed on an NVIDIA(R) GeForce RTX(TM) 4090 with 24GB GDDR6X. The table includes two metrics: the standard deviation of the final State of Charges (SOCs), denoted by $\sigma(s(t_f))$, and the running time of each algorithm

in seconds. It is worth noting that the running time for Algorithms A6 and A7 does not include their training time. In terms of the standard deviation of final State of Charges (SOCs), all of the algorithms discussed in this paper, except Algorithm A3, demonstrate better performance than Algorithm A7. The standard deviation of final SOCs for Algorithms A1, A2, A4, A5, and A6 ranges from 0.0011 to 0.0015, while Algorithm A7 has a higher standard deviation of 0.0021. As we can see, both algorithms $A1$ and $A2$ are able to achieve the optimal standard deviation of final SOCs, despite their different position matrices. The algorithm $A5$ is able to find a solution that is very close to the optimal one in terms of the performance metric. The performance of algorithm $A3$ is the worst, which indicates the disadvantage of the fixed platoon method compared with non-fixed platoon methods. In terms of running time, algorithms A3-A6 can find a solution much faster than algorithms A1 and A2. When performance and running time are both considered, algorithm A5 is a potentially good choice for vehicle platoon re-sequencing since it can achieve 90% optimal performance within 0.003% algorithm A2's running time.

Alg.	$\sigma(s(t_f))$	running time [s]
A1	0.0011	17.344302
A2	0.0011	7.951402
A3	0.0324	0.000581
A4	0.0054	0.000826
A5	0.0012	0.027033
A6	0.0015	0.016362
A7	0.0021	0.000791

TABLE III
THE COMPARISON OF ALGORITHMS A1-A6 IN DETERMINISTIC ENVIRONMENT

B. Simulation with stochastic environment conditions

This example simulates vehicle platoon driving in a stochastic environment. According to the uniform distribution, we randomly pick the average rate of electricity usage per unit distance in the range $[300, 400]$ Wh/mile. Here are the steps we execute to set reduction rates η_i^j (for $i = 1, \ldots, 4, j = 1, \ldots 5$):

(1) For each $j \in 1, \ldots, 5$, we generate a 4-dimensional random vector η^j in the range of $[0.04, 0.15]$ according to the uniform distribution.
(2) Sort the elements of η^j in decreasing order, and η_i^j is the i-th element of η^j after the sorting.

The initial SOCs are set randomly between 85% and 100% based on a uniform distribution. Other settings are the same as that in the last example. As we mentioned before, algorithms A3, A4, and A6 can be used with stochastic environment conditions. Besides, Algorithm A7, which employs the Monte Carlo method described in [30], can also be employed in stochastic environments. To compare the performance of these three algorithms fairly, we perform 1000 simulations and calculate their average values of the standard deviation of the final SOCs and the running time. The results are shown in Table IV. From the data presented in Table IV, it can be observed that algorithm A4 achieves the smallest standard deviation of final SOCs among the four algorithms, with a moderate running time. Algorithm A3, on the other hand, requires the least amount of time to find a suboptimal solution. Algorithm A6 yields a moderate standard deviation of final SOCs, and it is important to note that the training time for both algorithms A6 and A7 are not included in the reported running time. Comparing these three algorithms with algorithm A7, we can see that A7 has a slightly higher standard deviation of final SOCs than algorithms A4 and A6, but the running time of A7 is significantly smaller than A6. However, it is important to note that A7 requires a long training time, which is not reflected in the running time presented in Table IV. Comparing the performance and running time, algorithm A4 is the most efficient among the three algorithms.

Alg.	$\sigma(s(t_f))$	running time [s]
A3	0.0286	0.00049
A4	0.0084	0.00364
A6	0.0183	0.17350
A7	0.0196	0.06451

TABLE IV
THE COMPARISON OF ALGORITHMS A3, A4, AND A6 IN STOCHASTIC ENVIRONMENT

VI. CONCLUSION

A research gap in the formulation of electric vehicle platooning is addressed, viz., the imbalance of energy consumption caused by fixed platoons. Several re-sequencing algorithms, namely brutal force, modified brutal force, modified fixed platoon, SOC ranking, and max-min swap algorithms for a deterministic environment and a reinforcement learning algorithm using deep Q networks for a stochastic environment, are designed for solving this problem which allow vehicles in a platoon changing positions during a trip. The performances of all proposed algorithms are compared in a real transportation route in terms of time cost and the standard deviation of the final SOCs. Another major contribution of this work is the finding of the optimal formation at the last stage. Experiments confirm the efficiency of the max-min swap algorithm in the deterministic environment. The results also reveal that the SOC ranking algorithm is more efficient in the stochastic environment compared with other algorithms. The results can be used to guide transportation planners in making re-sequencing decisions in an efficient manner and in a reasonable time.

REFERENCES

[1] S. E. Li, Y. Zheng, K. Li, Y. Wu, J. K. Hedrick, F. Gao, and H. Zhang, "Dynamical modeling and distributed control of connected and automated vehicles: Challenges and opportunities," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 3, pp. 46–58, 2017.
[2] A. Coppola, D. G. Lui, A. Petrillo, and S. Santini, "Eco-driving control architecture for platoons of uncertain heterogeneous nonlinear connected autonomous electric vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 24220–24234, 2022.

[3] X. He and X. Wu, "Eco-driving advisory strategies for a platoon of mixed gasoline and electric vehicles in a connected vehicle system," *Transportation Research Part D: Transport and Environment*, vol. 63, pp. 907–922, 2018.

[4] C. Guo, C. Fu, R. Luo, and G. Yang, "Energy-oriented car-following control for a front- and rear-independent-drive electric vehicle platoon," *Energy*, vol. 257, p. 124732, 2022.

[5] K. B. Devika, G. Rohith, and S. C. Subramanian, "Impact of v2v communication on energy consumption of connected electric trucks in stable platoon formation," in *2023 15th International Conference on COMmunication Systems and NETworkS (COMSNETS)*, pp. 42–47, 2023.

[6] L. Xu, W. Zhuang, G. Yin, and C. Bian, "Energy-oriented cruising strategy design of vehicle platoon considering communication delay and disturbance," *Transportation Research Part C: Emerging Technologies*, vol. 107, pp. 34–53, 2019.

[7] I. Srisomboon and S. Lee, "Efficient position change algorithms for prolonging driving range of a truck platoon," *Applied Sciences*, vol. 11, no. 22, 2021.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[9] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, 2010.

[10] Z. Zhan, J. Li, and J. Zhang, "Evolutionary deep learning: A survey," *Neurocomputing*, vol. 483, pp. 42–58, 2022.

[11] L. Song and W. Fan, "Traffic signal control under mixed traffic with connected and automated vehicles: A transfer-based deep reinforcement learning approach," *IEEE Access*, vol. 9, pp. 145228–145237, 2021.

[12] Y. Yao, J. Zhao, Z. Li, X. Cheng, and L. Wu, "Jamming and eavesdropping defense scheme based on deep reinforcement learning in autonomous vehicle networks," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1211–1224, 2023.

[13] M. Wu, F. R. Yu, P. X. Liu, and Y. He, "A hybrid driving decision-making system integrating markov logic networks and connectionist ai," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 3, pp. 3514–3527, 2023.

[14] S. T. Kaluva, A. Pathak, and A. Ongel, "Aerodynamic drag analysis of autonomous electric vehicle platoons," *Energies*, vol. 13, no. 15, 2020.

[15] X. Meng and C. G. Cassandras, "Trajectory optimization of autonomous agents with spatio-temporal constraints," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 3, pp. 1571–1581, 2020.

[16] S. E. Shladover, C. A. Desoer, J. K. Hedrick, M. Tomizuka, J. Walrand, W. Zhang, D. H. McMahon, H. Peng, S. Sheikholeslam, and N. McKeown, "Automated vehicle control developments in the path program," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 1, pp. 114–130, 1991.

[17] J. Smith, R. Mihelic, B. Gifford, and M. Ellis, "Aerodynamic impact of tractor-trailer in drafting configuration," *SAE International Journal of Commercial Vehicles*, vol. 7, pp. 619–625, sep 2014.

[18] M. Zabat, N. Stabile, S. Farascaroli, and F. Browand, "The aerodynamic performance of platoons: A final report," report, UC Berkeley: California Partners for Advanced Transportation Technology, 1995.

[19] P. Vegendla, T. Sofu, R. Saha, M. Madurai Kumar, and L.-K. Hwang, "Investigation of aerodynamic influence on truck platooning," in *SAE 2015 Commercial Vehicle Engineering Congress*, SAE International, sep 2015.

[20] M. P. Lammert, A. Duran, J. Diez, K. Burton, and A. Nicholson, "Effect of platooning on fuel consumption of class 8 vehicles over a range of speeds, following distances, and mass," *SAE International Journal of Commercial Vehicles*, vol. 7, pp. 626–639, sep 2014.

[21] X. Lu and S. Shladover, "Automated truck platoon control," in *California PATH Research Report; University of California: Berkeley, CA, USA*, pp. 1–8, 2011.

[22] J. Scholl, N. Boysen, and A. Scholl, "E-platooning: Optimizing platoon formation for long-haul transportation with electric commercial vehicles," *European Journal of Operational Research*, vol. 304, no. 2, pp. 525–542, 2023.

[23] A. J. Qarebagh, F. Sabahi, and D. Nazarpour, "Optimized scheduling for solving position allocation problem in electric vehicle charging stations," in *2019 27th Iranian Conference on Electrical Engineering (ICEE)*, pp. 593–597, 2019.

[24] J. N. Forestieri and M. Farasat, "Energy flow control and sizing of a hybrid battery/supercapacitor storage in mvdc shipboard power systems.," *IET Electrical Systems in Transportation*, vol. 10, no. 3, pp. 275 – 284, 2020.

[25] A. C. Baller, M. van Ee, M. Hoogeboom, and L. Stougie, "Complexity of inventory routing problems when routing is easy," *Networks*, vol. 75, no. 2, pp. 113–123, 2020.

[26] D. J. Papageorgiou, M.-S. Cheon, G. Nemhauser, and J. Sokol, "Approximate dynamic programming for a class of long-horizon maritime inventory routing problems.," *Transportation Science*, vol. 49, no. 4, pp. 870 – 885, 2015.

[27] L. Bertazzi, A. Bosco, and D. Laganà, "Managing stochastic demand in an inventory routing problem with transportation procurement.," *Omega*, vol. 56, pp. 112 – 121, 2015.

[28] Q. Rabbani, A. Khan, and A. Quddoos, "Modified hungarian method for unbalanced assignment problem with multiple jobs," *Applied Mathematics and Computation*, vol. 361, pp. 493–498, 2019.

[29] A. H. Pereira, G. R. Mateus, and S. A. Urrutia, "Valid inequalities and branch-and-cut algorithm for the pickup and delivery traveling salesman problem with multiple stacks," *European Journal of Operational Research*, vol. 300, no. 1, pp. 207–220, 2022.

[30] S. Guo, X. Meng, and M. Farasat, "Energy efficient and battery soc-aware coordinated control of connected and autonomous electric vehicles," in *2022 American Control Conference (ACC)*, pp. 4707–4712, 2022.

[31] R. Klette, *Concise computer vision: an introduction into theory and algorithms*. Undergraduate Topics in Computer Science, Springer, 2014.

[32] B. Dennis S., *Scalar, Vector, and Matrix Mathematics : Theory, Facts, and Formulas - Revised and Expanded Edition.* Princeton University Press, 2018.

[33] D. Knuth and Addison-Wesley, *The Art of Computer Programming*. Addison-Wesley series in computer science and information processing, Addison-Wesley, 1997.

[34] W. Rudin, *Principles of mathematical analysis.* International series in pure and applied mathematics, McGraw-Hill, 1976.