How to ensure a safe control strategy? Towards a SRL for urban transit autonomous operation

Zicong Zhao, *Graduate Student Member, IEEE*

arXiv:2311.14457v1 [cs.AI] 24 Nov 2023

***Abstract*—Deep reinforcement learning has gradually shown its latent decision-making ability in urban rail transit autonomous operation. However, since reinforcement learning can not neither guarantee safety during learning nor execution, this is still one of the major obstacles to the practical application of reinforcement learning. Given this drawback, reinforcement learning applied in the safety-critical autonomous operation domain remains challenging without generating a safe control command sequence that avoids overspeed operations. Therefore, a SSA-DRL framework is proposed in this paper for safe intelligent control of urban rail transit autonomous operation trains. The proposed framework is combined with linear temporal logic, reinforcement learning and Monte Carlo tree search and consists of four mainly module: a post-posed shielding, a searching tree module, a DRL framework and an additional actor. Furthermore, the output of the framework can meet speed constraint, schedule constraint and optimize the operation process. Finally, the proposed SSA-DRL framework for decision-making in urban rail transit autonomous operation is evaluated in sixteen different sections, and its effectiveness is demonstrated through an ablation experiment and comparison with the scheduled operation plan.**

***Index Terms*—Safe Reinforcement Learning, Urban Rail Transit, Autonomous Operation, Intelligent Control.**

I. Introduction

REINFORCEMENT learning (RL) is now a widely used method to handle complex muti-state decision-making issues such as the Gridworld and Cliffwalking [1]. Deep reinforcement learning (DRL), which combines the RL method with neural network, has tremendous potential to solve realistic continuous space problem such as the train autonomous control and high dimensional discrete space game like Chess.

Automatic train operation (ATO) system of urban rail transit is to help to control the train automatically track the recommended speed profile to complete operation plan with high efficiency [2]. However, with the widespread use of fully automatic operation (FAO) system and train autonomous circumambulation system (TACS), the operation control system is required to has the ability of autonomous operation, self adjustment, self protection, thus the traditional speed tracking system may restrict the research of advanced train control system.

DRL is of widespread use to improve the performance of control system. There have been some researchers who had applied DRL method to train speed profile optimization or automatic control. But many of them ignored one important thing that due to the random exploration or the useless punishment, unavoidable safety issues are big problems for safety-critical applications in real world. As for the operation of urban rail transit, it means that traditional DRL may lead to an overspeed operation. We admit here the train operation is under the protect of automatic protection system (ATP), but since the ATO and ATP are two independent systems, an advanced control method should not only control the operation process comfortable and efficient but also cause as little protection as possible. To overcome the disadvantage above, safe reinforcement learning (SRL), a subfield of RL has become a hot research in recent years. However there are few researches focusing on how to construct a SRL based control method for urban rail transit. In that case, it is of great importance to study a safety-considered SRL control method for urban rail transit autonomous operation.

A. Related Work

In optimal urban transit control area, most studies can be regarded as an extension of the study of speed profile optimization. In the past few decades, due to the characteristics of speed tracking control, researchers mainly focused on how to optimize the speed profile, so that the control algorithm, especially PID control, can achieve better control effect. Up to now, most studies are based on Pontryagin's Maximum Principle (PMP) to analysis the switching of working condition such as maximum acceleration (MA), coasting (CO), maximum braking (MB) to optimize the speed profile [3]–[8]. Such optimizing methods are also called energy-efficient train control (EETC) [9].

With the development of intelligent control theory and the requirement of self decision-making in autonomous operation, intelligent control methods represented by dynamic programming (DP) and RL are widely studied to improve automation level of traditional ATO system.

As the basis of RL and DP, Bellman optimal equation is used to build a multi-stage decision making problem for train operation control [10]. Compared with the famous bionic optimization algorithm such as genetic algorithm and ant colony optimization, DP has a better performance under different operation time and inter-section distance [11]. With the development of train motors and controllers, continuous control commands are more and more used in urban rail transit, which makes the traditional RL or DP method unsuitable. DRL and approximate dynamic programming (ADP) are then used to handle this problem. For the optimal control of heavy haul train with uncertain environment conditions, the maximum utility of regenerative braking energy and the optimization of speed profile with parametric uncertainty, ADP all has a good performance [12]–[14]. As for the specific use of DRL, researches on one hand use it directly to output control

command for train [15]–[17], and on the other hand combine it with other framework such as the expert system or history experience to correct the given control command [18]–[20].

In RL research area, with the development of computer science and graphics process unit (GPU), several famous algorithms represented by the Q-learning, actor-critic (AC), deep deterministic policy gradient (DDPG) and soft actor-critic (SAC) have achieved tremendous achievement in two-player game and computer game [21]–[24]. The two most famous researches are AlghaGo and AlphaZero in Chess and AlphaStar in StarCraft for they have almost beaten every human player without pressure [25]–[27].

Though DRL has shown great potential in decision-making area, researchers have also found that RL algorithms do not necessarily guarantee safety during learning or execution phases, which leads to an unavoidable drawback for safety-critical application in real world such as robot control [28], [29]. To get over this issue, researches have studied to ensure reasonable system performance and respect safety constraints during the learning and deployment processes, such researches are so-called SRL [30]. Considering there are many different ways to classify SRL algorithms, application area will be used in this paper to make the literature review.

In the car autonomous driving area, many methods have been proposed for autonomous driving based on modern, advanced techniques. A constrained RL based motion planning method is proposed for automated driving. Traditional motion planning and RL methods are combined to perform better than pure RL or traditional methods [31]. Different with [31], a third layer called risk neural network is added to AC algorithm to realize safe autonomous driving [32]. Moreover, control barrier functions (CBF) and MCTS are also used for safe autonomous driving [33], [34].

In the robotics area, the past research always do not consider the safety of robot as an optimization objective, the study of SRL has established a bridge between the simulation and application. Optlater ensures safe action sets that a robot can only taken from [35]. Safe exploration derived from risk function is used to construct PI-SRL and PS-SRL algorithm, which makes SRL based robot walking come true [36], [37].

SRL is also widely used in other areas. In recommender system, SRL is deployed to optimize the healthy recommendation sequences by utilizing a policy gradient method [38]. In wireless security, an Inter-agent transfer learning based SRL method is proposed [39]. In urban aerial vehicles (UAV) control, a brain-inspired reinforcement learning model (RSNN) is proposed to realize self-organized decision-making and online collision avoidance [40]. In high speed train operation optimization, a Shield SARSA algorithm is proposed to plan an energy-efficient speed profile without overspeed operation [41]. In diabetes treatment, a new mathematical problem formulation framework called Seldonian optimization problem is proposed, and it is used in the optimization of insulin dosing for type 1 diabetes treatment [42].

B. Problems and Contributions

Through the literature review, two important information can be acquired are that SRL are now widely used in safety-critical area to make RL based method more real world realizable and in urban rail transit area there are few researches typically considering how to construct a SRL based intelligent control method for automatic operation. Table I summarized the safety protection methods for solving train operation optimization or control problems using RL in recent years. It is clear that the widely used method to prevent overspeed operation nowadays are adding a punishment or set the speed equal to the limit speed. However, the effect of a punishment may be influenced by the value of punishment weight and can only be known after several simulations which is obvious unsuitable for the operation in real world. When setting the speed equal to limit speed, this behavior may break the principle that at each time step the agent receives some representation of the environment's state and on that basis selects an action. Moreover, since this approach ignore the behavior policy, it may not maximize the long term reward.

TABLE I
TYPICAL SAFETY PROTECTION METHODS

References	Safety Protection Method
[19]	If $v > v_l$, adapt minimum deceleration
[20]	If $v > v_l$, using reference system to brake
[43]	If $v > v_l$, safety index equal to 0
[44]	If $v > v_l$, add an overspeed punishment
[45]	If $v > v_l$, regard as infeasible
[46]	If $v > v_l$, reset environment
[47]	$v \leq v_l$ in model but not mentioned in RL algorithm
[48]	$v \leq v_l$ in model but not mentioned in RL algorithm
[49]	If $v > v_l$, add an overspeed punishment -1
[50]	Add an avoidance punishment
[51]	If $v > v_l$, add an operspeed punishment

Then in this paper, a SRL framework called SSA-DRL is proposed for safe decision making of urban rail transit autonomous operation. The framework is consists of a Shield, a searching tree, a DRL framework and an additional safe actor. The main contributions of this paper can be summarized as follows:

- The proposed SSA-DRL framework enables agent to learn safe control policies and ensure schedule constraints and operation efficiency.
- The proposed SSA-DRL framework can not only suitable for deterministic continuous algorithm but also suitable for stochastic discrete algorithm.
- The proposed SSA-DRL framework can can effectively reduce the number of times that the protection mechanism works which means that the final agent has self-protection ability.

The remainder is organized as follows. In Section II, the preliminaries are introduced. In Section III, the proposed SSA-DRL framework is elaborated. In Section IV, simulation results are discussed and the conclusions are given in Section V.

II. PRELIMINARIES

A. Markov Decision Process

A finite Markov Decision Process is usually denoted by the 5-tuple $(\mathcal{S}, s_0, \mathcal{A}, p, \mathcal{R})$ with a finite state set $\mathcal{S} = \{s_0, ..., s_n\}$, a unique initial state $s_0 \in \mathcal{S}$, a finite action set $\mathcal{A} =$

$\{a_1, ..., a_n\}$, a dynamic function $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ and a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

The solving of a RL task is to find a mapping called policy written as π from states to probabilities of selecting each possible action to achieve a lot of reward over the long run. For the optimal policy, it is better than or equal to all other policies. Noted that there may be more than one optimal policy, thus all the optimal policies are denoted by π_*. All the optimal policies share the same optimal state-value function v_* and optimal action-value function q_* defined as

$$\begin{cases} v_* (s) \doteq \max\limits_{\pi} v_\pi (s) \\ q_* (s, a) \doteq \max\limits_{\pi} q_\pi (s, a) \end{cases} \tag{1}$$

B. State Value and Action Value

Value function is widely used in RL to evaluate a given state or a state-action pair. Since the return an agent can get almost depend on the chosen action, thus value function is associated with the policy. The function $v_\pi (s)$, which calculates the value of state s under policy π is called a state value function and is denoted by (2).

$$\begin{aligned} v_\pi (s) &\doteq \mathbb{E}_\pi [G_t \mid S_t = s] \\ &= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right], \text{ for all } s \in \mathcal{S} \end{aligned} \tag{2}$$

Similarly, the value function q_π calculates the value of taking an action a at state s under policy π and is denoted by (3).

$$\begin{aligned} q_\pi(s, a) &\doteq \mathbb{E}_\pi [G_t \mid S_t = s, A_t = a] \\ &= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right] \end{aligned} \tag{3}$$

$\mathbb{E}[\cdot]$ denotes the expected value of a random variable.

C. Off Policy DRL

In off-policy RL, the agent uses a behavior policy for action selection during the learning process, and a target policy for updating the policy. This means that the policies used by the agent while learning are different from those actually executed. The core feature of off-policy RL is to seek the global optimal value. In DRL especially, due to the introduction of replay buffer, off-policy DRL algorithms are more common. The DDPG and SAC algorithms are two examples of off-policy methods based on policy gradient framework AC, which are used as benchmarks.

DDPG is an deterministic algorithm typically designed for continuous action set which concurrently learns a Q-function $Q(s, a)$ and a policy. It has two AC structures and uses off-policy data and the Bellman equation to learn the Q-function, and uses the Q-function to learn the policy. At each state s, the optimal action is acquired by solving (4).

$$a_* (s) = \arg\max_a Q_* (s, a) \tag{4}$$

SAC is an algorithm that optimizes a stochastic policy in an off-policy way, forming a bridge between stochastic policy optimization and DDPG-style approaches. Different from DDPG, SAC is suitable for both continuous and discrete action set. The most core feature of SAC is entropy regularization, which means the algorithm is designed to search a trade-off between expected return and entropy. Unlike the traditional DRL algorithm, SAC finds the optimal policy by solving (5).

$$\pi^* = \arg\max_\pi \mathop{\mathrm{E}}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t, s_{t+1}) + \alpha H(\pi(\cdot|s_t)) \right) \right] \tag{5}$$

$$H(P) = \mathop{\mathrm{E}}_{x \sim P} [-\log P(x)] \tag{6}$$

H is the entropy of x calculated by its distribution P.

Although both DDPG and SAC have learned good agents on several benchmark tasks, there is no guarantee of safety in these algorithms, nor any other traditional off-policy RL algorithm. Therefore, the purpose of this paper is to combine DRL agents with other modules to both improve control efficiency and ensure safety.

D. Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is an algorithm based on tree search and Monte Carlo method for decision-making problems, widely used in the field of games and RL. It simulates multiple possible states of a game or problem and selects the optimal action scheme to find the best decision. MCTS iteratively simulates the subsequent development of a searching tree, updates the nodes in the tree according to the simulated results, and selects one node by a policy as the action for the next step. The widely used policies are upper confidence bounds and $\epsilon - greedy$. The following step consist the basic steps of MCTS which are selection, expansion, simulation and backpropagation. In this paper, the process of MCTS is addressed to better suitable for the proposed framework.

E. Linear Temporal Logic

Linear Temporal Logic (LTL) is a widely used temporal logic method in areas such as formal modeling and model checking. It can describe constraints and temporal relationships that need to be satisfied by a system in the past, present, and future using time sequence operators. Therefore, it is particularly suitable for describing reactive systems that generate outputs based on external inputs and the system's current state. LTL can conveniently and accurately describe the properties and constraints that a system needs to meet, and is typically described using linear temporal logic formulas.

A **word** is typically used to express an atomic proposition (AP) or the negation of an AP. Alphabet Σ of AP is denoted as 2^{AP}, where 2^{AP} represents the power set of AP. Subsequently, the sets of all finite and infinite sequences of elements from the alphabet Σ are denoted as Σ^* and Σ^ω, respectively. Important and widely used properties such as safety, satisfiability, and liveness can be defined through linear temporal logic formulas.

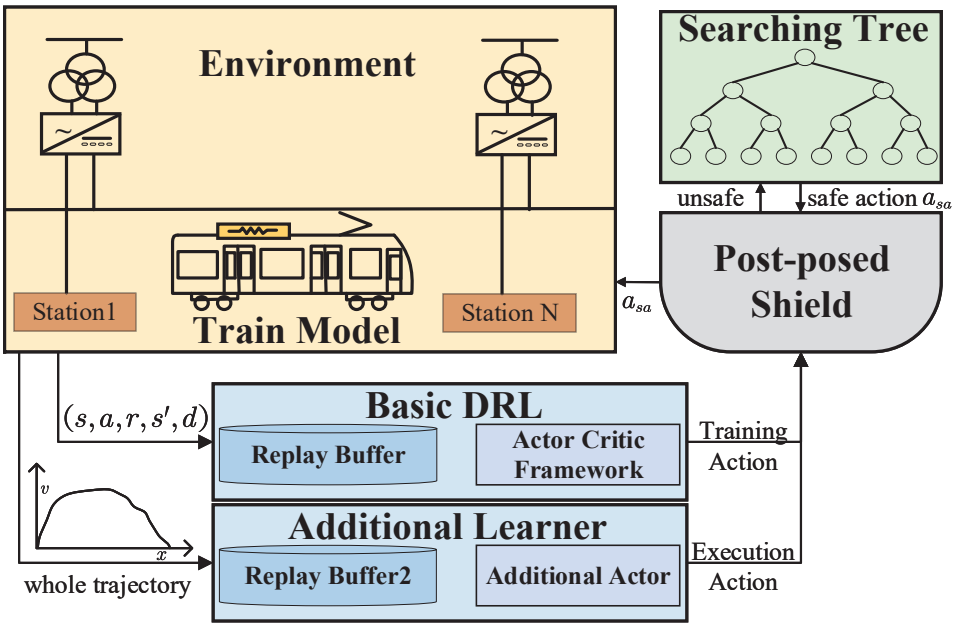


Fig. 1. Framework of SSA-DRL.

III. Method Formulation

In this section, the framework of the proposed SSA-DRL is firstly shown in Fig. 1. It is clearly that SSA-DRL consists of four main modules: a Shield based protective module, a searching tree based module, a DRL module and an additional actor module. Then we will explain how these four modules work in detail.

A. A Post-Posed Shielding

The Shielding in this paper comes directly from [28]. The Shielding consists of finite-state reactive system, safety specification, an observer function, a label and other components. Finite-State Reactive System is usually denoted by a tuple $\mathcal{FS} = \{\mathcal{Q}, q_0, \Sigma_I, \Sigma_O, \delta, \lambda\}$, where $\mathcal{Q}$ is the finite set of states, $q_0 \in \mathcal{Q}$ is the initial state, $\Sigma_I = \Sigma_I^1 \times \Sigma_I^2, \Sigma_O$ are the input and output alphabet. Then, $\delta : \mathcal{Q} \times \Sigma_I \to \mathcal{Q}, \lambda : \mathcal{Q} \times \Sigma_I^1 \to \Sigma_O$ are the transition and output function respectively. Specification ϕ defines the set of all allowed traces, and once a system satisfies all the properties of a specification, we can say a system satisfies ϕ. Moreover, safety specification is used to construct the Shield. Formally speaking, a safety specification is a specification that if every trace is not in the language represented by the specification has a prefix such that all words starting with the prefix are also not in the language [28]. The above expression may be difficult to understand and readers of this paper can simply recognize a safety specification holds that "bad things will never happen" and a safety automaton can be used to represent a safety specification [52]. Observer function $f : \mathcal{S} \to L$ is usually a mapping for an MDP $\mathcal{M} = (\mathcal{S}, s_0, \mathcal{A}, p, \mathcal{R})$ to describe some information at state s and L is a finite set of label. Then, once an RL task can be formulated as $\mathcal{M} = (\mathcal{S}, s_0, \mathcal{A}, p, \mathcal{R})$ while satisfying a safety specification ϕ^S with an observer function $f : \mathcal{S} \to L$, a Shield can be modeled by a reactive system.

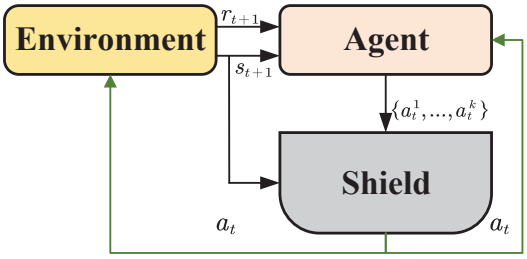


Fig. 2. Structure of post-posed Shielding.

Post-Posed Shielding is a specific form of Shielding that is set after the learning algorithm as depicted in Fig. 2. It is clear that the actions chosen by the agent are monitored by the Shield and the action violating safety specification ϕ^S will be replayed by a safe action $a^{'}$. A new action set consists of $a^{'}$ is denoted as $\mathcal{A}_s^{'}$. Then the reactive system can be re-written as $\mathcal{FS} = (\mathcal{Q}_\mathcal{S}, q_{0,\mathcal{S}}, \Sigma_{I,\mathcal{S}}^1 \times \Sigma_{I,\mathcal{S}}^2, \Sigma_{O,\mathcal{S}}, \delta_\mathcal{S}, \lambda_\mathcal{S})$.

A simple example is made here to show how to build a post-posed Shielding for the safe control of urban transit. Considering the train is running in a section with only one speed limitation which is 120km/h. The action set is $\mathcal{A} = \{\text{acceleration}, \text{coasting}, \text{braking}\}$. In the operation process, the speed must hold in the range of 1-119 km/h and the working condition cannot directly changed from acceleration to braking so as the braking to acceleration. Firstly a safety specification for the speed controller can be formulated by the temporal

logic formula as follow (7).

$$\begin{aligned} &\mathrm{G}\left(\text{speed} > 1\right) \\ \wedge &\mathrm{G}\left(\text{speed} < 119\right) \\ \wedge &\mathrm{G}\left(\text{acceleration} \rightarrow \mathrm{X}\left(\text{coasting}\right)\ \mathrm{U}\ \text{braking}\right) \\ \wedge &\mathrm{G}\left(\text{braking} \rightarrow \mathrm{X}\left(\text{coasting}\right)\ \mathrm{U}\ \text{acceleration}\right) \end{aligned} \tag{7}$$

The meaning of LTL formulas G,X(),U are globally, next time and until respectively, and the label set can be formulated as $L = \{speed < 1, 1 \leq speed \leq 119, speed > 119\}$.

Then the component of Shielding are discussed. Firstly, the finite state set $Q_{\mathcal{S}}$ can be set as $Q_{\mathcal{S}} = G$, where G is the finite safe set of a safety game satisfies safety specification $\phi^{\mathcal{S}}$ [28]. Then the initial state is $q_{0,\mathcal{S}} = (q_0, q_{0,\mathcal{M}})$. We make a brief introduction of $q_{0,\mathcal{M}}$ here. As mentioned above, the reactive system to construct Shield should satisfy the safety specification, actually, it should satisfy another specification which is the MDP specification $\phi^{\mathcal{M}}$, and $q_{0,\mathcal{M}}$ is the initial state of $\phi^{\mathcal{M}}$. The input alphabet is $\Sigma_{I,\mathcal{S}} = \Sigma^1_{I,\mathcal{S}} \times \Sigma^2_{I,\mathcal{S}} = L \times \mathcal{A} = \{\text{acceleration}, \text{coasting}, \text{braking}\} \times \{speed < 1, 1 \leq speed \leq 119, speed > 119\}$ and the output alphabet is $\Sigma_{O,\mathcal{S}} = \mathcal{A} = \{\text{acceleration}, \text{coasting}, \text{braking}\}$. The output function can be formulated as (8) where $a \in \mathcal{A}, a' \in \mathcal{A}^{'}_{\mathcal{S}}, g \in G, l \in L$ and W is the set of the winning state of safety game G.

$$\lambda_{\mathcal{S}}\left(g, l, a\right) = \begin{cases} a \text{ if } \delta\left(g, l, a\right) \in W \\ a' \text{ if } \delta\left(g, l, a\right) \notin W, \text{but } \delta\left(g, l, a'\right) \in W \end{cases} \tag{8}$$

And the transition function is $\delta_{\mathcal{S}}(g, l, a) = \delta(g, l, \lambda_{\mathcal{S}}(g, l, a))$.

The above example points out the steps to build a post-posed Shield for urban rail transit control and then a tree search based module is proposed to better find a safe action $a^{'}$.

B. Safe Action Search Tree

In this subsection, a search tree based module is proposed to output the final safe action. The idea of the search tree derives from roll out algorithm and is more like a trade-off between MCTS and exhaustive search. Firstly, the Post-Posed Shielding provide a safe action set $\mathcal{A}^{'}$ and the searching module using several steps to finally choose the high long-term reward safe action. The framework of the module is depicted in Fig. 3.

A detailed example is made here to explain how to construct and use the searching tree. Suppose that the initial unsafe state is s_{un} and the safe action set is $\mathcal{A}^{'}_{sa} = (a^{'}_{sa,1}, ..., a^{'}_{sa,n})$. To output a high long-term reward safe action, each action in $\mathcal{A}^{'}_{sa}$ should be evaluated. $a_{sa,1}$ is chosen firstly then the state will transfer to a new state $s_{sa,1}$, this is actually a roll out or simulation step and $s_{sa,1}$ is the root node. At state $s_{sa,1}$, the DRL agent will output n_{ex} actions, and a simulation step is then executed according to $s_{sa,1}$ and the n_{ex} actions. It is noted here that all these simulation steps are monitored by the Shield which means that only safe action will be executed and in this step those unsafe actions will not be replaced by a safe one. This means that only safe actions can expand nodes and root node $s_{sa,1}$ will have $n^{'}_{ex}$ children nodes ($n^{'}_{ex} \leq n_{ex}$). Then, for these nodes, a roll out step can be executed again. Considering in most DRL algorithms the policy neural network will be updated in the learning phase, thus if the searching tree is a full-depth tree there will be a tricky that the action in real training and in expansion at the same state may be different. For example, if the root node is step 3, the update frequency is 5 and the current expansion step is 6. In expansion, the action at step 6 is output by μ^{θ}_0 but in training, the exact action at step 6 is output by μ^{θ}_5 where μ is the parameter of the policy neural network so that the expansion can not deduce a specific future. A trick is used here that the depth of the searching tree will not be fixed but dynamically equal to the remaining step to update the net, which means that the step of the leaf nodes will always be the same as the update step. In this case, the depth of the searching tree will not be too large and the each searching tree in training phase can be step-adaptive.

Once the expansion step reaches the update step, the searching tree needs to be pruned. In pruning, all children nodes that are not extended to the update step are deleted. Pruning can help to remove those nodes that will lead an unsafe state and guarantee that only safe state are returned. After pruning, the searching tree needs to be returned. For the q_{th} node at step p, the return $r^{ex}_{p,q}$ is calculated by (9)

$$\begin{cases} r^{ex}_{p,q} = r^{si}_{p,q} + 0.9 * \mathbb{E}\left[r^{ex}_{p,q,chi}\right], \text{branch node} \\ r^{ex}_{p,q} = r^{si}_{p,q}, \text{leaf node} \end{cases} \tag{9}$$

where $r^{si}_{p,q}$ is the roll out reward and $\mathbb{E}\left[r^{ex}_{p,q,chi}\right]$ is the expectation return of all children nodes of $node_{p,q}$. The final safe action a_{sa} of state s_{un} can be chosen by (10)

$$a_{sa} = \arg\max_{a \in \mathcal{A}^{'}_{sa}} r^{ex}_{s_{un},a} \tag{10}$$

The pseudocode of the searching tree is shown in Alg.1.

C. DRL based guiding learner

Though a Post-posed Shield has a strong ability to prevent the occurance of unsafe actions, it also has two disadvantages:

- Unsafe actions may be part of the final policy, thus the Shield needs to be active even in the learning phase.
- Unsafe action always will be replaced by safe actions, thus the agent will never learn how to avoid unsafe actions by itself.

These two disadvantages are both severe for calculating time-critical tasks like urban rail transit control because if the Shield is always active, the solving time of a safe action may be longer than the control period. The second disadvantage will also lead to another problem that the agent does not has the self-protection ability. Then the SSA-DRL is introduced based on the Shield and searching tree and simple AC algorithm is used here to illustrate how the learner work.

The SSA-DRL seeks to solve the following optimization problem:

$$\begin{aligned} \pi^* = \underset{\pi}{\text{argmax}} &\mathbb{E}[\textstyle\sum_{t=0}^{T} \gamma^t r\left(s_t, a_{sa,t}\right)] \\ \text{s.t. } &a_{sa,t} \in \mathcal{A}^{'} \end{aligned} \tag{11}$$

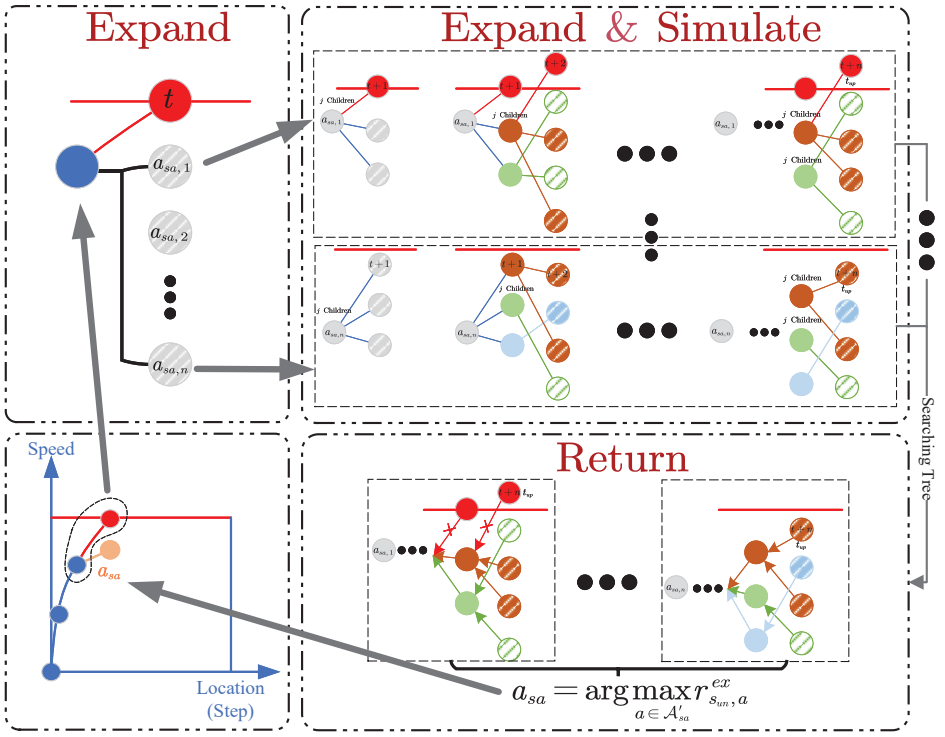


Fig. 3. Framework of safe action searching tree.

Then the action-value function $Q^{\pi}(s_t, a_t)$ for policy π at step t can be calculated by the given policy net μ_θ or the searching tree $\mathcal{T}(\cdot)$denoted by (12).

$$\begin{aligned} Q^{\pi}(s_t, a_t) &= Q\left(s_t, \left(\mu^{\theta}(s_t), \mathcal{T}(s_t)\right)\right) \\ &= \mathbb{E}\left[R_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1})\right], a_t, a_{t+1} \in \mathcal{A}^{'} \end{aligned} \tag{12}$$

In the learning phase of DRL, random noise is always used to increase exploration. The noise is added to the policy net and facilitates exploration in the action set for more efficient exploration. Ornstein-Uhlenbeck (OU) noise and Guassian noise are widely used since OU noise is autocorrelation and Gaussian noise is easy to design and realize in real world. Thus, the action chosen by the policy net can be represented as:

$$\begin{cases} \mu^{\theta}(s_t) = \mu^{\theta}(s_t|\Theta^{v}_t) + \varepsilon \\ \varepsilon \sim \mathcal{N}_{\mathrm{OU}} \end{cases}, \text{OU noise} \tag{13}$$

$$\begin{cases} \mu^{\theta}(s_t) = \mu^{\theta}(s_t) + \lambda\beta \\ \beta \sim \mathcal{N}_{\mathrm{Ga}}(0,1) \end{cases}, \text{Ga noise} \tag{14}$$

The parameter of the action-value function ϕ can be learned by minimizing the loss function of the critic net as presented by (15).

$$\nabla_{\phi}\mathbb{E}_{(s,a,r,s',d)\sim\mathcal{B}}\left[(Q^{\pi}_{\phi}(s,a) - y(r,s',d))^2\right] \tag{15}$$

Where $\mathcal{B}$ is a sample batch of transitions (s, a, r, s', d) stored in replay buffer $\mathcal{D}$. $y(\cdot)$ is called the target and is usually computed by (16).

$$y(r,s',d) = r + \gamma(1-d)Q_{\phi}(s', \mu_{\theta}(s')) \tag{16}$$

It is noted here that the memory in replay buffer $\mathcal{D}$ follows first in first out principle and the experience batch is randomly sampled thus the sampled experience may not contain a whole trajectory. Then another replay buffer $\hat{\mathcal{D}}$ and an additional policy neural net $\hat{\mu}$ are introduced.

The replay buffer $\hat{\mathcal{D}}$ is used to save $\hat{N}$ whole trajectories $(S_{\hat{n}}, A_{\hat{n}}, R_{\hat{n}})$ with highest reward $\hat{R}_{\hat{n}}$ denoted by (17). The experiences in $\hat{\mathcal{D}}$ follow the best in worst out principle and are ranked by the value of the reward.

$$\begin{aligned} \hat{\mathcal{D}} &= \left[\mathrm{tr}_1, ..., \mathrm{tr}_{\hat{n}}\right], \hat{n} \in [1, \hat{N}] \\ \mathrm{tr}_{\hat{n}} &= \left[s^{\mathrm{tr}_{\hat{n}}}_t, a^{\mathrm{tr}_{\hat{n}}}_t, r^{\mathrm{tr}_{\hat{n}}}_t\right], t \in [0, T] \end{aligned} \tag{17}$$

The additional net $\hat{\mu}$ with parameter $\hat{\theta}$ is used to study the self-protection ability. Since there exists $a^{\mathrm{tr}_{\hat{n}}}_t \in \mathcal{A}'_{sa}$, thus if $\forall\hat{n} \in \hat{N}, \exists\mathbb{E}[|a^{\mathrm{tr}_{\hat{n}}}_t - \hat{\mu}^{\hat{\theta}}(s^{\mathrm{tr}_{\hat{n}}}_t)|^2] < \hat{\epsilon}$, the additional policy net can be used as the final policy net. The parameter $\hat{\theta}$ can be updated by (18).

$$\nabla_{\hat{\theta}} \frac{1}{|\hat{\mathcal{B}}|} \sum_{(s^{\mathrm{tr}_{\hat{n}}}_t, a^{\mathrm{tr}_{\hat{n}}}_t) \in \hat{\mathcal{B}}} \left(a^{\mathrm{tr}_{\hat{n}}}_t - \hat{\mu}^{\hat{\theta}}(s^{\mathrm{tr}_{\hat{n}}}_t)\right)^2, |\hat{\mathcal{B}}| \leq \hat{N} \tag{18}$$

Where $\hat{\mathcal{B}}$ is a sample batch of several whole trajectories stored in $\hat{\mathcal{D}}$. Moreover, the structure of $\hat{\mu}$ may be different from the structure of μ, they only have the same dimension and scale of input and the output which are actually the states and actions. This method can improve the generalization ability and deployability of the final policy net. It is also noted here that this method does not mean the Shield is not needed after

Algorithm 1 Searching Tree Process

Input: The unsafe state,s_{un}^{t};The safe action set, $\mathcal{A}_{sa}^{'}$;The policy net;The expansion width, W;The current step,t;The update frequency,t_{up}
Output: A safe action, a_{sa}

```
while m ≤ |A'_sa| do
  for each safe action a'_m ∈ A'_sa do
    Get a new state s^{t+1}_{sa,m} by roll out policy with action a_m
    if (t+1)%t_up ≠ 0 then
      while (t+1)%t_up ≠ 0 do
        while w ≤ W do
          Get action a_w by policy net
          if a_w is monitored safe by Shield then
            Get a new state s^{t+2}_{sa,w} by roll out policy with action a_m
            if (t+2)%t_up = 0 then
              Return Searching Tree by (9)
            end if
            w = w + 1
          else
            w = w + 1
          end if
        end while
        t = t + 1
      end while
    else
      Return Searching Tree by (9)
    end if
  end for
  m = m + 1
end while
a_sa = arg max_{a∈A'_sa} r^{ex}_{s_un,a}
```

training. In the simulation section, the results will prove that this method can make the Shield be less used and save the solving time. Then the whole SSA-DRL algorithm is outlined in Alg.2. It is also noted here that the specific steps for updating different DRL algorithms are not exactly the same, the steps in Alg.2 only involves three core ideas in DRL which are policy evaluation, policy improvement and target network. Once other DRL algorithm is used to implement the SSA-DRL algorithm, only some update steps in Alg.2 need to be addressed but the core idea is unchanged. In simulation section, the effectiveness of SAC based SSA-DRL algorithm is a good example to verify the above idea.

D. Optimality and Convergence Analysis

1) Optimality: In the learning process, there actually exists two policies to get an action, the policy net and the searching tree. The optimality of the policy net does not need to be proved. Then two aspects of the optimality of searching tree are discussed, the first is optimality of a_{sa} and the second is the policy π_{sa} to get a_{sa} is no less than the policy net μ^{θ} to get an action once a state is monitored unsafe.

Algorithm 2 SSA-DRL algorithm

Input: Policy neural net parameter,θ;Q-function neural net parameter,ϕ;Additional policy neural net parameter,$\hat{\theta}$;The maximum training episode,J
Output: Safe policy parameter, $\hat{\theta}$

```
Initialize parameter θ, φ, θ̂
Construct Post-posed FS_sh
while j ≤ J do
  repeat
    Observe state s and get an action a by (13) or (14)
    FS_sh check action a
    if a is safe then
      a_sa = a
    else
      Choose a safe action a_sa by Alg.1
    end if
    Execute action a_sa and observe next state s', done signal d and reward r
    D ← D ∪ (s, a, r, s', d)
    if s' is the terminal step and R_j ≥ min_{tr∈D̂} tr[r] then
      D̂ ← D̂ ∪ tr_j
    else
      Reset the environment
    end if
    if update basic neural network then
      Sample a batch of transitions B = {(s, a, r, s', d)} from D
      Compute targets by (16), update φ by (15)
      Update θ by
          ∇_θ (1/|B|) Σ_{s∈B} Q_φ(s, μ_θ(s))
      if there exists target network then
        Update target networks by soft update
      end if
    end if
    if update additional policy neural network then
      Sample a batch of whole trajectories B̂ = {tr} from D̂
      for many updates do
        Update θ̂ by (18)
      end for
    end if
  until Environment is reset
end while
```

Lemma 1. *For policies to get a safe action, π_{sa} to get a_{sa} is better than any other policies to get another $\pi_{sa}^{'}$. Moreover, π_{sa} is no less than the original policy μ^{θ} to get a safe action.*

Proof. The concept of policy improvement is used here to make the proof. In Alg.1 obviously, the final safe action a_{sa} is actually chosen by the greedy strategy, thus a_{sa} is the action with the highest long-term reward and then π_{sa} is better than other $\pi_{sa}^{'}$. Then, the idea of policy improvement is used to prove that when choosing a safe action, π_{sa} is no less than μ^{θ}. Since the actions used in searching tree are all generated by μ^{θ}, thus they are identical equal besides the initial unsafe state $\pi_{sa}(s_{un}) \neq \mu^{\theta}(s_{un})$. Then if $q_{\mu_\theta}(s_{un}, \pi_{sa}(s_{un})) > v_{\mu^\theta}(s_{un})$, the policy π_{sa} is no less than policy μ^{θ}. Keep expanding q_{μ^θ} we can get (19). It is obvious $q_{\mu_\theta}(s_{un}, \pi_{sa}(s_{un})) > v_{\mu^\theta}(s_{un})$, thus $\pi_{sa}(s_{un})$ is no less than μ^{θ}. We must admit here this proof is not very rigorous, since the original policy improvement method requires $t+n$ to be infinite, but in this paper a clipped form is used.

$$
\begin{aligned}
& q_{\mu^\theta}\left(s_{un}, \pi_{sa}\left(s_{un}\right)\right) \\
& = \mathbb{E}\left[R_{t+1} + \gamma v_{\mu^\theta}\left(S_{t+1}\right) \mid S_t = s_{un}, A_t = \pi_{sa}\left(s_{un}\right)\right] \\
& = \mathbb{E}_{\pi_{sa}}\left[R_{t+1} + \gamma v_{_{\mu^\theta}}\left(S_{t+1}\right)\right] \\
& \leq \mathbb{E}_{\pi_{sa}}\left[R_{t+1} + \gamma q_{_{\mu^\theta}}\left(S_{t+1}, \pi_{sa}\left(S_{t+1}\right)\right)\right] \\
& = \mathbb{E}_{\pi_{sa}}\left[R_{t+1} + \gamma \mathbb{E}\left[R_{t+2} + \gamma v_{\mu^\theta}\left(S_{t+2}\right)\right]\right] \\
& = \mathbb{E}_{\pi_{sa}}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_{\mu^\theta}\left(S_{t+2}\right)\right] \\
& \leq \mathbb{E}_{\pi_{sa}}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_{\mu^\theta}\left(S_{t+3}\right)\right] \\
& \vdots \\
& \leq \mathbb{E}_{\pi_{sa}}\left[\underbrace{R_{t+1} + \cdots + \gamma^{n-1} R_{t+n}}_{\text{eq.}(11)}\right], (t+n)\,\%t_{up} \neq 0 \\
& = v_{\pi_{sa}}\left(s_{un}\right)
\end{aligned}
\tag{19}
$$

□

2) Convergence: The convergence of the Post-posed Shield has been verified in [28] and the feasibility of using MDP specification and safety specification to protect the operation of train has been verified in [41], thus the Post-posed Shield in this paper satisfies the convergence analysis in [28]. The simulation results in Section V also verifies the convergence of SSA-DRL algorithm.

E. Algorithm Implementation

In this subsection, the state set, action set, reward function and the relationship between action and the acceleration are discussed to complete the SSA-DRL based urban rail transit autonomous operation algorithm.

1) State Set: The location *loc*, velocity *vel*, running time *time* are used to formulated the state set. Thus the state of the agent at step t can be formulated as (20).

$$s_t = (loc_t, vel_t, time_t) \tag{20}$$

2) Action Set: The percentage of the traction braking control command output to the motor of train is used as the action. The action set is continuous and ranges from -1 to 1. If the value is less than 0, the command is braking otherwise the command is traction. Then the action at step t can be formulated as (21).

$$a_t \in [-1, 1] \tag{21}$$

3) Reward Function: Since the operation has been protected and the main purpose is autonomous control, then operation energy consumption E, operation time difference D_T and the comfort of passengers C are used to build the reward function. In each transition step t, E, D_t, C are calculated by (22),(23) and (24).

$$E = \begin{cases} \alpha_{E_t r} * E_{tra}, a > 0 \\ \alpha_{E_r e} * E_{reg}, a \leq 0 \end{cases} \tag{22}$$

$$T = \begin{cases} \alpha_{D_T} * |T_{total} - T_{sch}|, \text{terminal} \\ \alpha_{D_T}^{'} * |\bar{v}_t - \bar{v}|, \text{mid step} \end{cases} \tag{23}$$

$$C = \begin{cases} \kappa, \Delta acc > \sigma \\ 0, \Delta acc < \sigma \end{cases} \tag{24}$$

Where E_{tra} is the traction energy consumption, E_{reg} is the recovered regenerative braking energy, $\alpha_{E_t r}, \alpha_{E_r e}, \alpha_{D_T}, \alpha_{D_T}^{'}$ are the weights of energy reward and time reward, T_{total}, T_{sch} are the total operation time and scheduled operation time, $\hat{v_t}, \hat{v}$ are the average speed in step t and the overall average speed, κ is a punishment indicator, Δacc is the rate of change of acceleration and σ is a threshold. The reward R then is defined as (25).

$$R = -E - T - C \tag{25}$$

It is also noted that the value of E_{reg} is set negative then the train can learn to reduce total energy consumption.

4) Relationship between control command and train operation: The operation of train is restricted by the traction braking characteristic thus the control command cannot directly represent the movement of train. Suppose a given command $a_{tra}^t > 0$ at step t, the acceleration output by the traction motor is $acc_{motor}^t = F_{tra}^+(v^t) * a_{tra}/m_{train}$ where $F_{tra}^+(v^t)$ denotes the max traction force at speed v^t is a function of speed and m_{train} is the weight of train. Likely, the acceleration of braking can be computed in the same way. Then the actual acceleration of train can be computed by (26).

$$acc_{train} = \frac{(acc_{motor} - acc_r + g\left(x\right))}{v} \tag{26}$$

Where acc_r is the resistance acceleration calculated by the Davis function and $g(x)$ is the gravity acceleration at location x. Moreover, there exists no steep slope in this paper thus (27) always holds.

$$0 \leq acc_r - g\left(x\right) \leq \max\left(acc_{motor}\right) \tag{27}$$

IV. Simulation Results and Performance Evaluation

A. Simulation Environment

The simulation is based on Chengdu urban rail transit line 17, there are a total of sixteen section in up and down

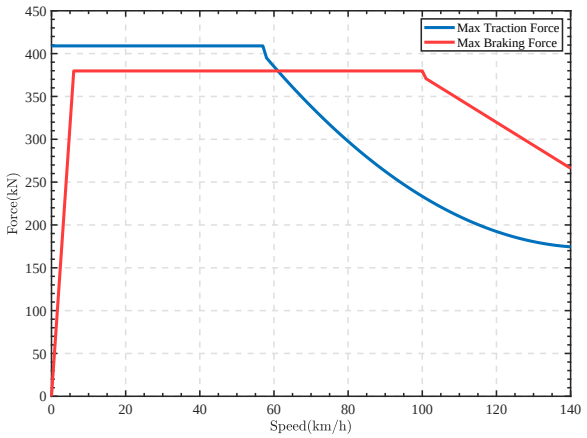


Fig. 4. Traction and braking characteristic.

directions. The value of the Davis parameters are $r_1 = 8.4, r_2 = 0.1071, r_3 = 0.00472$ respectively, the weight of the train is 337.8 ton, the max and min acceleration are $\pm 1.2\text{m/s}^2$ and the traction braking characteristic is shown in Fig. 4. The main parameters used to construct the SSA-DRL algorithm are shown in TableII.

TABLE II
THE MAIN PARAMETERS OF THE ALGORITHM

Parameters	Value	Parameters	Value
Traction Weight $\alpha_{E_{tr}}$	3	Regenerative Weight $\alpha_{E_{re}}$	3
Time Weight α_{D_T}	15	Time Weight α'_{D_T}	25
Comfort Punishment κ	10	Change Threshold σ	3
Minibatch $\lvert\mathcal{B}\rvert$	256	Minibatch $\lvert\hat{\mathcal{B}}\rvert$	10
Max Episode J	1100	Update Frequency t_{up}	5

TABLE III
THE MAIN HYPERPARAMETERS OF THE ALGORITHM

Parameters (DDPG)	Value	Parameters (SAC)	Value
Actor learning rate	$1e^{-5}$	Policy learning rate	$1e^{-5}$
Critic learning rate	$1e^{-3}$	Value learning rate	$1e^{-3}$
Discount factor	0.99	Discount factor	0.99
Soft update factor	$1e^{-2}$	Soft update factor	$1e^{-2}$
/	/	Soft-Q learning rate	$3e^{-5}$

The basic DDPG and SAC are used as the baseline to implement autonomous driving in this paper, and the actor and critic networks are both designed through four fully connected hidden layer, the activation functions in hidden layers are Relu and the size of hidden layers is 256 and the final activation function of the actor net is tanh to ensure the output ranges in $[-1, 1]$. The optimizers are all Adam. The main hyperparameters of these two algorithms are shown in Table.III. It is noted that in the simulation SSA-DRL algorithm shares the same hyperparameters with the baseline. Moreover, the originally additional actor is designed the same with the actor, but with a five times learning rate. The influence caused by the design of additional actor will be discussed by the ablation experiment. The proposed algorithm is implemented in Matlab and Python on a computer with an AMD Ryzen 7 5800X CPU @ 3.80Ghz and 32GB RAM running Windows 10 x64 Edition.

B. Basic Simulation

The basic simulation aims to verify that the proposed SSA-DRL can control the train complete the operation plan with higher reward and better performance under less protect times in both training and execution process. Here,the protect times is the number of counts the algorithm re-chooses a safe action to correct the original action in training or execution process. Fig. 5 and Fig. 6 are the reward curves of SSA-DRL, Shield-DRL and common DRL algorithms in the eight different sections under two different directions. It can be clearly seen that in most scenarios SSA-DRL can achieve a higher reward than Shield-DRL and the reward of Shield-DRL is also higher than common DRL. Moreover, the reward curves of SSA-DRL are smoother than Shield-DRL and common DRL and SSA-DRL can achieve convergence at a earlier step. It is noted that the reward curves are all smoothed by the moving average and the size of the window is 8. The detailed numerical results are shown in Table.IV. The data in time and energy columns are acquired from one operation plan. It is noted that the operation time is not fixed and a margin of thirty seconds is allowed. And in the simulation columns, the data are recorded by $ave \pm std$.

Fig. 7 and Fig. 8 are the speed profiles of the SSA-DRL algorithm in one simulation, readers may think that the speed profiles do not match Table.IV, it should be made clear that the speed profiles are only results of one simulation but the data in Table.IV are numerical results after several simulation times.

Fig. 9 shows bar graphs of average protect times in training and execution process of SSA-DRL and Shield-DRL. Compared with the Shield-DRL algorithms, the protect times of SSA-DRL algorithms has greatly reduced. Moreover, the distributions of the protect times of SSA-DRL in all scenarios are more concentrated and errors are all smaller, which indicates that the SSA-DRL is more stable to some extent.

Then, combine the five figures with the detailed numerical results shown in Table.IV, a conclusion can be drawn that SSA-DRL can control the train complete the operation plan and reduce traction energy consumption without overspeed danger which we also say the proposed SSA-DRL ensures a safe control strategy.

C. Transferability Experiment

This experiment aims to test whether the trained neural network of SSA-DRL can be deployed to a new environment. The trained SSA-DDPG and SSA-SAC algorithms in section8 of up and down direction are deployed to section1-section7 of the same direction and the results are shown in TableV.

The meaning of noise test is briefly explained here. Since in this experiment the trained network is deployed in a new environment, there is a possibility that the network cannot work and the output will always be a noise. Considering the characteristic of tanh function, once the network cannot work, there may exist two special noise action sequences, all -1 or all 1, which indicates that the train will always brake or accelerate. Obviously, the all -1 action sequence cannot complete the operation plan. However, once the action sequence is all 1, it may complete the operation plan because

TABLE IV
THE ORIGINAL SIMULATION RESULTS

Section	Direction	Time(s)	Simulation Time(s)		Energy(kw·h)	Simulation Energy(kw·h)		Overspeed Counts	
			SSA-SAC	SSA-DDPG		SSA-SAC	SSA-DDPG	SSA-SAC	SSA-DDPG
1	Up	86	96.55±4.34	83.79±15.53	52	29.38±1.12	21.98±2.02	0	0
	Down	92	91.31±12.53	95.56±2.83	45	21.61±1.66	24.12±5.25	0	0
2	Up	183	185.54±7.54	224.51±36.49	116	53.52±2.04	47.67±7.34	0	0
	Down	204	208.24±6.34	206.38±2.53	88	23.41±1.25	42.37±9.78	0	0
3	Up	276	276.25±3.47	283.65±6.64	160	83.85±11.91	55.93±3.09	0	0
	Down	259	267.76±14.16	293.65±6.95	142	85.10±15.43	31.53±3.01	0	0
4	Up	104	118.88±4.37	106.74±3.93	56	30.54±1.87	36.73±6.63	0	0
	Down	101	107.75±1.89	104.71±3.88	61	16.44±1.32	31.01±6.85	0	0
5	Up	104	131.73±11.41	108.99±5.33	68	21.80±0.71	33.77±6.63	0	0
	Down	103	107.64±2.58	106.03±3.49	67	31.60±2.68	39.84±4.87	0	0
6	Up	105	119.75±8.83	110.07±5.33	63	42.10±2.70	47.08±4.69	0	0
	Down	106	110.03±4.31	109.46±3.63	48	20.19±9.13	16.32±5.46	0	0
7	Up	105	113.42±5.09	114.54±10.82	63	40.07±1.40	43.86±3.86	0	0
	Down	138	138.82±4.74	139.46±1.36	48	13.90±3.59	11.31±4.11	0	0
8	Up	172	170.64±4.30	180.07±3.70	63	75.15±6.27	60.23±4.26	0	0
	Down	171	176.52±2.28	172.50±2.59	48	39.28±3.04	33.75±3.70	0	0

TABLE V
THE RESULTS OF TRANSFERABILITY EXPERIENT

Sec	Direction	Actor	Time	Real Energy	Protect Counts	Noise Test	Maximum Step
1	Up	Section8(SSA-DDPG)	95.2	22.61	48	92	146
	Down	Section8(SSA-SAC)	97.05	35.67	0	18	29
2	Up	Section8(SSA-DDPG)	199.54	44.92	56	163	208
	Down	Section8(SSA-SAC)	391.35	85.48	0	92	109
3	Up	Section8(SSA-DDPG)	1146.95	64.46	38	143	165
	Down	Section8(SSA-SAC)	499.53	144.69	0	133	165
4	Up	Section8(SSA-DDPG)	120.14	32.95	18	42	54
	Down	Section8(SSA-SAC)	138.85	36.15	0	40	54
5	Up	Section8(SSA-DDPG)	124.87	22.8	18	35	50
	Down	Section8(SSA-SAC)	134.32	42.61	0	34	50
6	Up	Section8(SSA-DDPG)	122.53	39.41	21	38	49
	Down	Section8(SSA-SAC)	117.11	20.98	0	40	49
7	Up	Section8(SSA-DDPG)	151.16	40.49	39	60	79
	Down	Section8(SSA-SAC)	292.52	45.55	0	66	79

the original action 1 always forces the train to accelerate and when the train is overspeed, the Shield and the searching tree will provide a safe action help the train to slow down. In this case, though the train completes the operation plan, it cannot be regarded as the trained network is transferability. In order to distinguish this case, the noise test is designed. In this test, we directly provide a noise action sequence with all 1 to record the protect counts. Once the trained network can complete the operation plan and the protect counts is far less than the noise test, the trained network is transferability.

Then from TableV, the SSA-DDPG is transferability in Section1,2,4,5,6,7 and the SSA-SAC is transferability in Section1,6,7. It is noted that this experiment is not rigorous and the experiment result can only verify the SSA-DRL may be transferability.

D. Robustness Experiment

This experiment aims to verify the robustness of the SSA-DRL, that is, the ability to complete the operation plan under the condition that the action is disturbed. Two parameters ε_r, δ_r are introduced in this experiment to control the probability and magnitude of action disturbance. That is, for an action a_r, it has $\varepsilon_r \times 100\%$ probability to change to another action $a_r + \mathcal{N}_r, \mathcal{N}_r \in [-\delta_r, \delta_r]$. The range of $\varepsilon_r, |\delta_r|$ are both $[0.1, 0.5]$ and the stepsize is 0.1. The Pearson correlation coefficient (PCC) is used here to measure the degree of correlation between the original sequence and the disturbed sequence. SSA-DDPG in Section1 up direction and SSA-SAC

in Section1 down direction are used. Fig. 10 shows the original and disturbed curves of speed profile, action sequence and acceleration sequence. In Fig. 10, the original curves are red bold and the disturbed curves are blue dash. It is clear that the disturbed curves are of the same trend with the original curve and there is no completely different curve or one curve is totally changed after a disturbance. Fig. 11 shows the changing trend of PCC when ε_r and δ_r change. The PCCs of speed profile , action sequences and acceleration sequences are all larger than 0.985, 0.7 and 0.75, since the PCC is more close to 1, the two curves are more linear correlation, thus combined with Fig. 10, it can be concluded that the SSA-DRL may have a strong robustness in some scenarios.

V. Conclusion

Aiming at the safe control strategy for urban rail transit autonomous operation, an SRL framework called SSA-DRL is proposed in this paper. The SSA-DRL uses a LTL based post-posed Shield to check whether an original action is safe and then uses a searching tree to find a safe action with the highest long term reward to correct the unsafe action. An additional learner consists of a replay buffer and an additional actor is also added to help to reduce the protect times in execution process.

Our framework is verified in simulations under three different aspects with two basic DRL algorithms. The basic experiment shows that the framework can control the train complete the operation plan with a lower energy consumption and protect times. And compared with the basic DRL or Shield-DRL algorithms, the SSA-DRL can get a higher reward and achieve convergence earlier in most simulation scenarios. The transferability and robustness experiment verify that a trained network can transfer to a new environment and can still complete the operation plan under some disturbances.

In the future work, we will try to extend this framework to a multi trains scenario and try to find a method to deal the situation that the algorithm is attacked by Generative Adversarial Network.

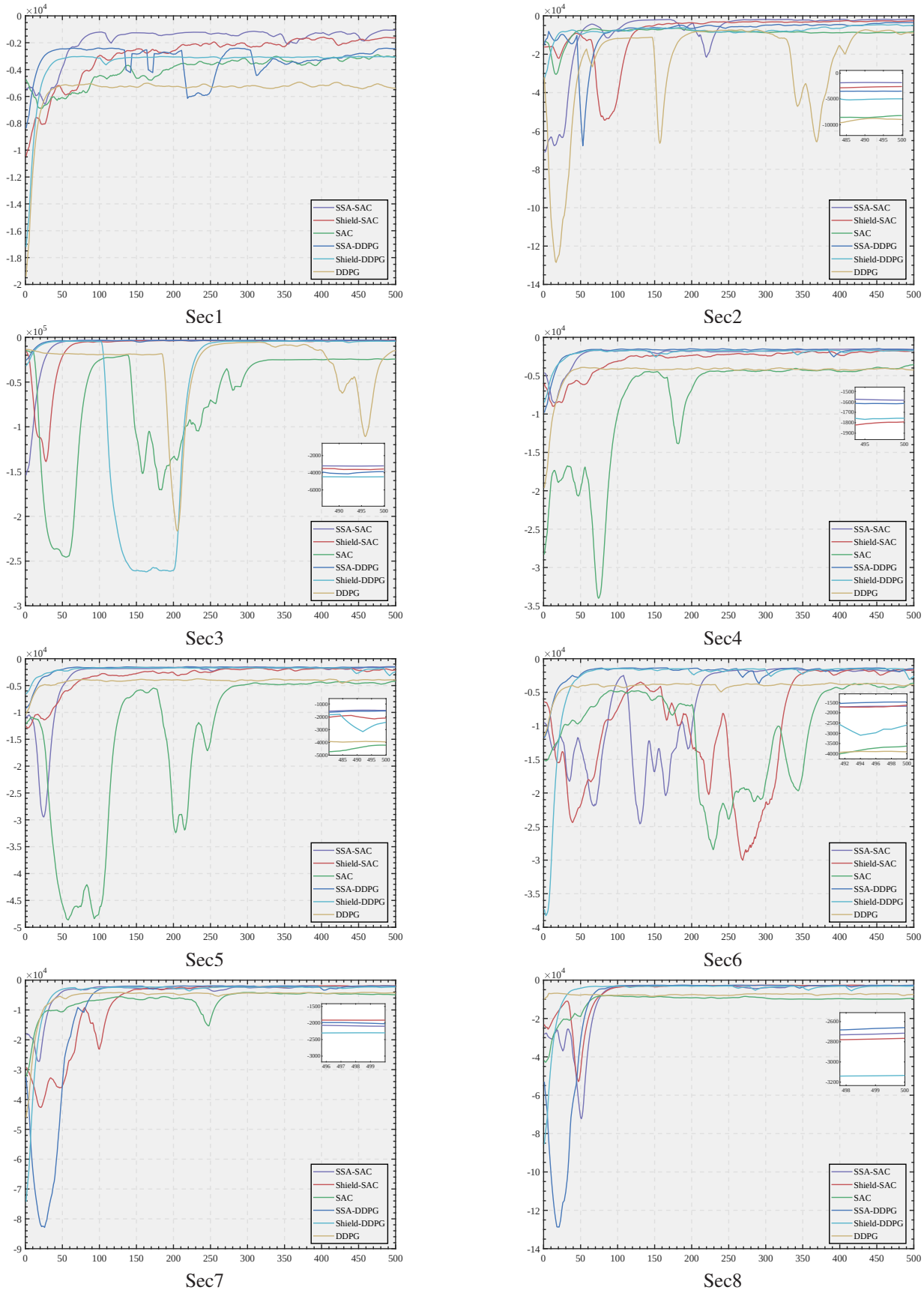


Fig. 5. Up Direction Reward Curve

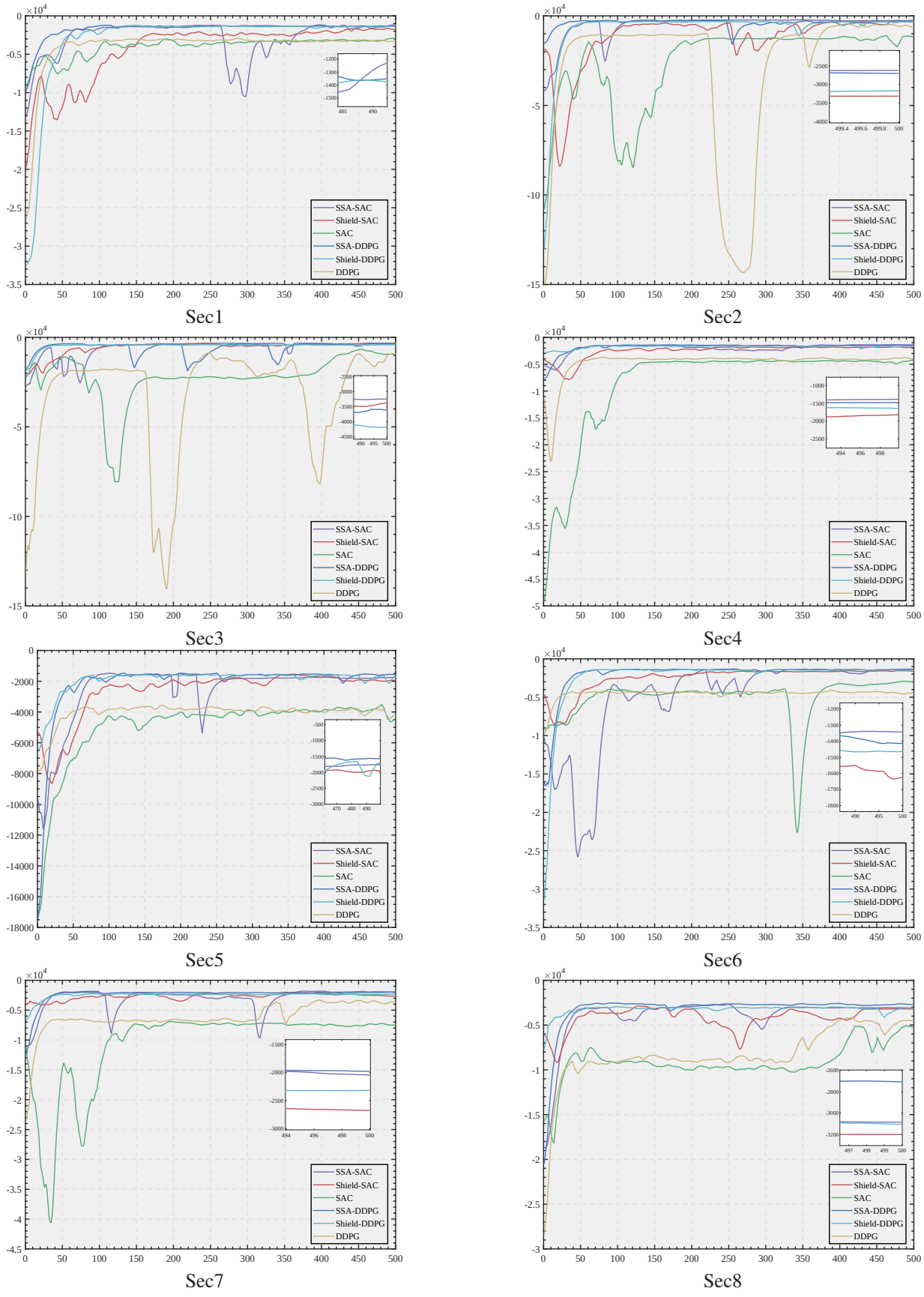


Fig. 6. Down Direction Reward Curve

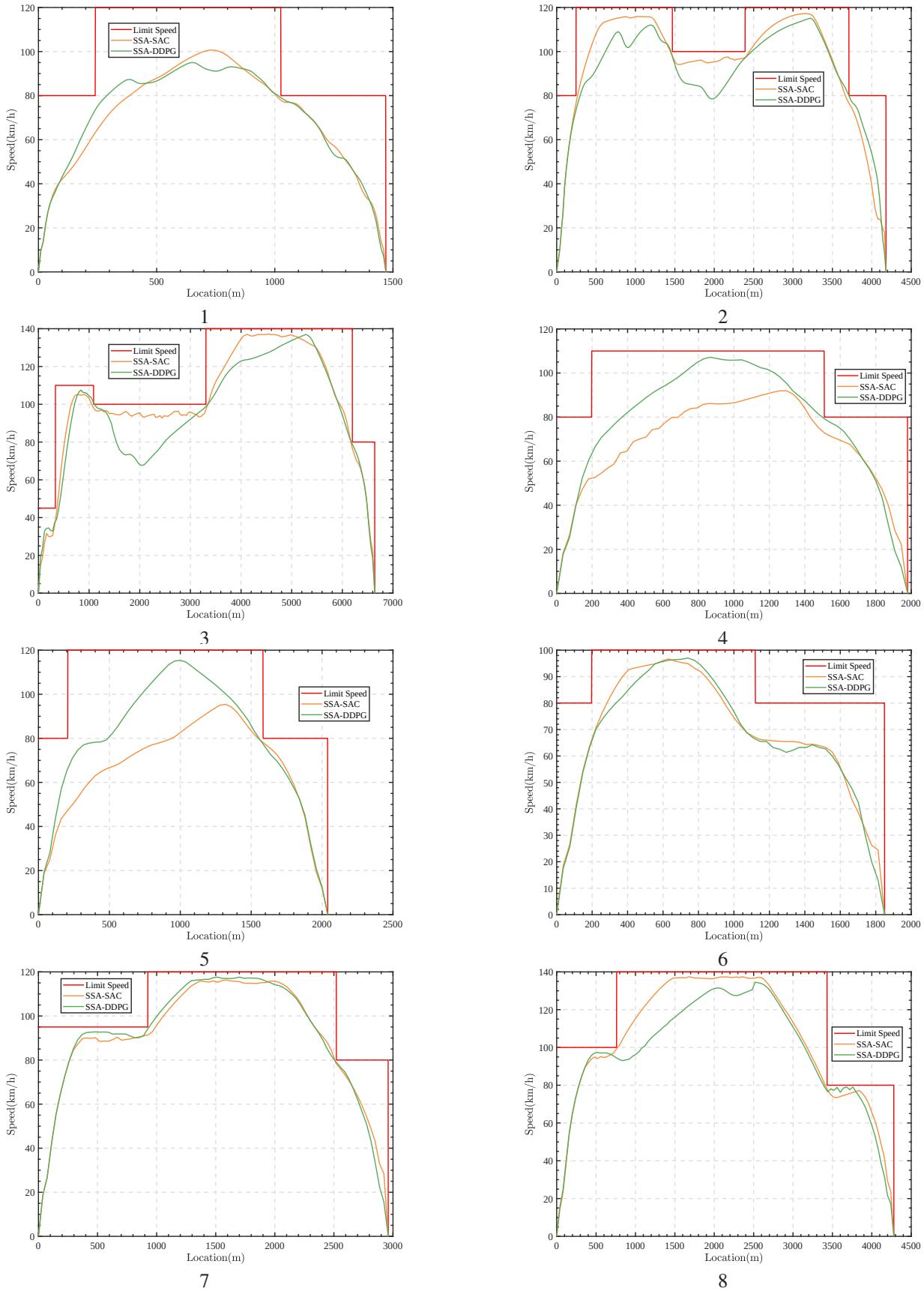


Fig. 7. Up Direction Speed Profile

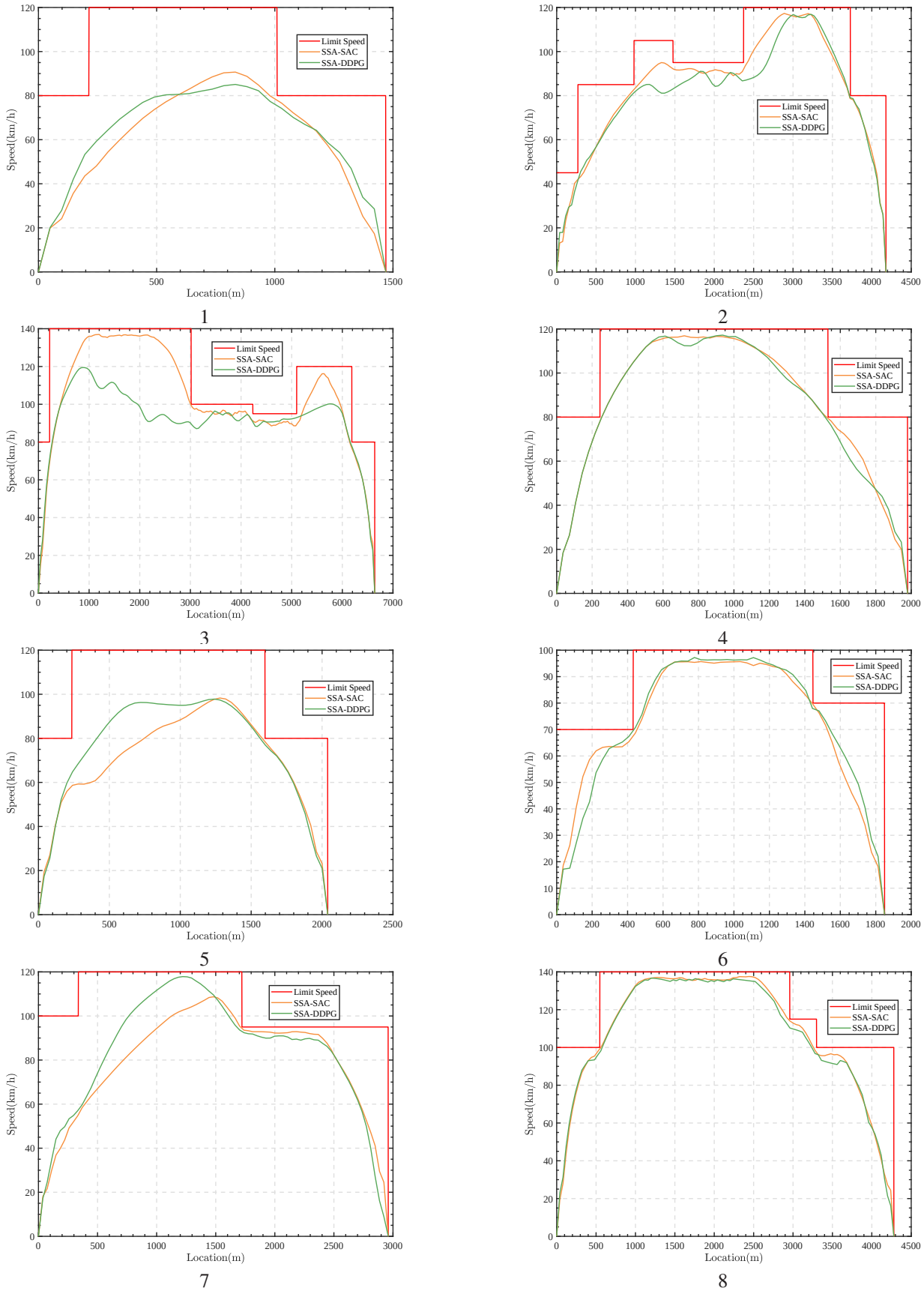
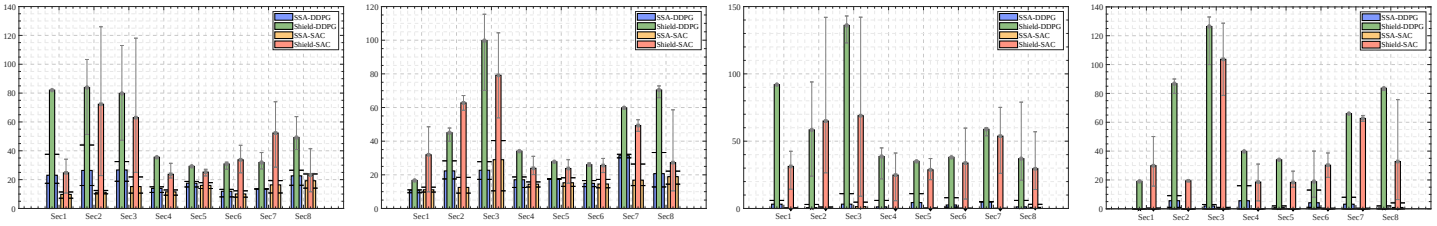
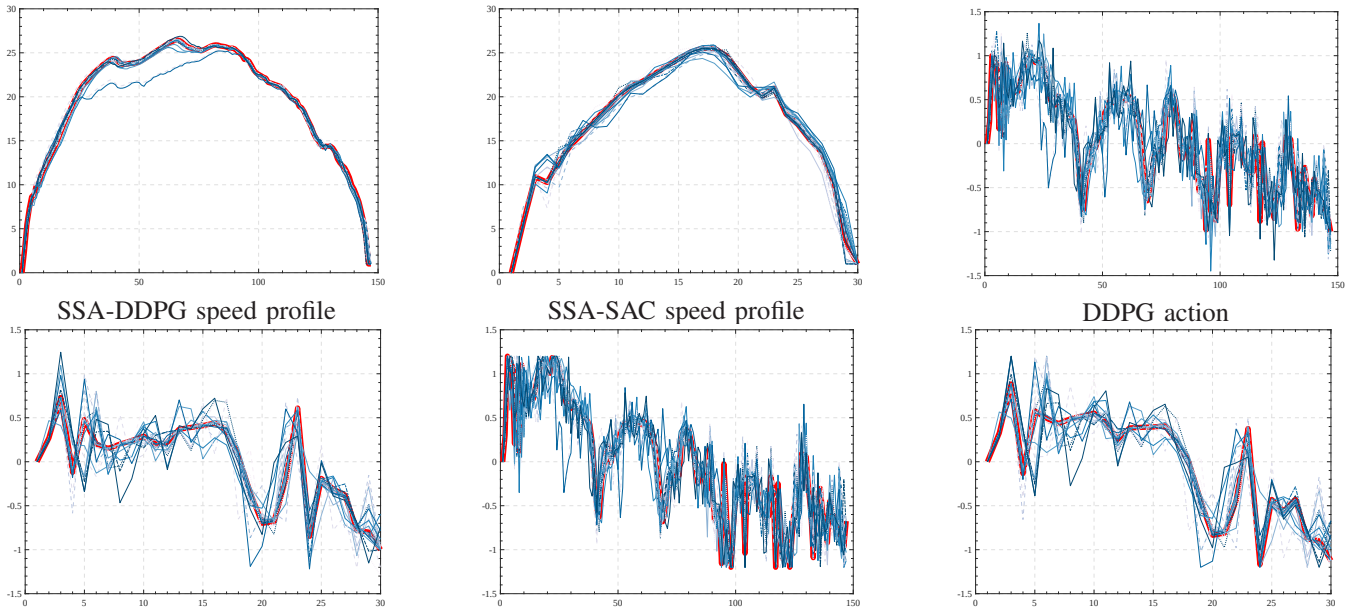


Fig. 8. Down Direction Speed Profile



Up Direction (Training) Down Direction (Training) Up Direction (Execution) Down Direction (Execution)

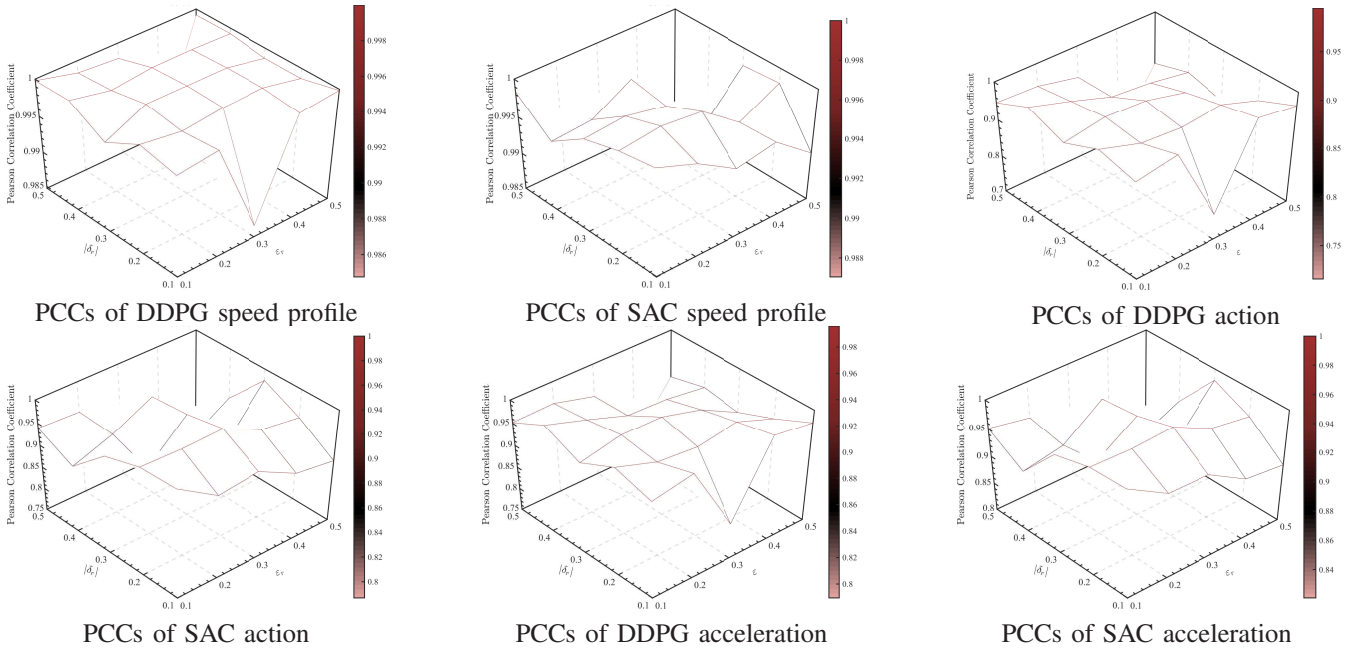
Fig. 9. Protect Times



SSA-DDPG speed profile SSA-SAC speed profile DDPG action

SAC action DDPG acceleration SAC acceleration

Fig. 10. Results of robustness experiments



PCCs of DDPG speed profile PCCs of SAC speed profile PCCs of DDPG action

PCCs of SAC action PCCs of DDPG acceleration PCCs of SAC acceleration

Fig. 11. Changing trend of PCCs

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] J. Yin, T. Tang, L. Yang, J. Xun, Y. Huang, and Z. Gao, "Research and development of automatic train operation for railway transportation systems: A survey," *Transportation Research Part C: Emerging Technologies*, vol. 85, pp. 548–572, 2017.

[3] P. Howlett, "An optimal strategy for the control of a train," *The ANZIAM Journal*, vol. 31, no. 4, pp. 454–471, 1990.

[4] P. G. Howlett and P. J. Pudney, *Energy-efficient train control*. Springer Science & Business Media, 2012.

[5] E. Khmelnitsky, "On an optimal control problem of train operation," *IEEE transactions on automatic control*, vol. 45, no. 7, pp. 1257–1266, 2000.

[6] A. Albrecht, P. Howlett, P. Pudney, X. Vu, and P. Zhou, "The key principles of optimal train control—part 2: Existence of an optimal strategy, the local energy minimization principle, uniqueness, computational techniques," *Transportation Research Part B: Methodological*, vol. 94, pp. 509–538, 2016.

[7] ——, "The key principles of optimal train control—part 1: Formulation of the model, strategies of optimal type, evolutionary lines, location of optimal switching points," *Transportation Research Part B: Methodological*, vol. 94, pp. 482–508, 2016.

[8] R. M. Goverde, G. M. Scheepmaker, and P. Wang, "Pseudospectral optimal train control," *European Journal of Operational Research*, vol. 292, no. 1, pp. 353–375, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221720308948

[9] G. M. Scheepmaker, R. Goverde, and L. G. Kroon, "Review of energy-efficient train control and timetabling," *European Journal of Operational Research*, vol. 257, no. 2017, p. 355–376, 2017.

[10] H. Ko, T. Koseki, and M. Miyatake, "Application of dynamic programming to the optimization of the running profile of a train," *WIT Transactions on The Built Environment*, vol. 74, 2004.

[11] S. Lu, S. Hillmansen, T. K. Ho, and C. Roberts, "Single-train trajectory optimization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 743–750, 2013.

[12] X. Wang, T. Tang, and H. He, "Optimal control of heavy haul train based on approximate dynamic programming," *Advances in Mechanical Engineering,9,4(2017-4-01)*, vol. 9, no. 4, p. 168781401769811, 2017.

[13] T. Liu, J. Xun, J. Yin, and X. Xiao, "Optimal train control by approximate dynamic programming: Comparison of three value function approximation methods," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2741–2746.

[14] P. Wang, A. Trivella, R. Goverde, and F. Corman, "Train trajectory optimization for improved on-time arrival under parametric uncertainty," *Transportation Research Part C Emerging Technologies*, vol. 119, p. 102680, 2020.

[15] L. Zhu, Y. He, F. R. Yu, B. Ning, T. Tang, and N. Zhao, "Communication-based train control system performance optimization using deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10 705–10 717, 2017.

[16] W. Liu, S. Su, T. Tang, and X. Wang, "A dqn-based intelligent control method for heavy haul trains on long steep downhill section," *Transportation Research Part C: Emerging Technologies*, vol. 129, p. 103249, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X2100262X

[17] ——, "A dqn-based intelligent control method for heavy haul trains on long steep downhill section," *Transportation Research Part C: Emerging Technologies*, vol. 129, p. 103249, 2021.

[18] J. Yin, D. Chen, and L. Li, "Intelligent train operation algorithms for subway by expert system and reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 6, pp. 2561–2571, 2014.

[19] K. Zhou, S. Song, A. Xue, K. You, and H. Wu, "Smart train operation algorithms based on expert knowledge and reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 2, pp. 716–727, 2020.

[20] M. Shang, Y. Zhou, and H. Fujita, "Deep reinforcement learning with reference system to handle constraints for energy-efficient train control," *Information Sciences*, vol. 570, pp. 708–721, 2021.

[21] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[22] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[23] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[24] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[25] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[26] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[27] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[28] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[29] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, Y. Yang, and A. Knoll, "A review of safe reinforcement learning: Methods, theory and applications," *arXiv preprint arXiv:2205.10330*, 2022.

[30] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[31] S. Gu, G. Chen, L. Zhang, J. Hou, Y. Hu, and A. Knoll, "Constrained reinforcement learning for vehicle motion planning with topological reachability analysis," *Robotics*, vol. 11, no. 4, p. 81, 2022.

[32] L. Wen, J. Duan, S. E. Li, S. Xu, and H. Peng, "Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–7.

[33] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 3387–3395.

[34] S. Mo, X. Pei, and C. Wu, "Safe reinforcement learning for autonomous vehicle using monte carlo tree search," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6766–6773, 2021.

[35] T.-H. Pham, G. De Magistris, and R. Tachibana, "Optlayer-practical constrained optimization for deep reinforcement learning in the real world," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6236–6243.

[36] J. Garcia and F. Fernández, "Safe exploration of state and action spaces in reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 45, pp. 515–564, 2012.

[37] J. García and D. Shafie, "Teaching a humanoid robot to walk faster through safe reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 88, p. 103360, 2020.

[38] A. Singh, Y. Halpern, N. Thain, K. Christakopoulou, E. Chi, J. Chen, and A. Beutel, "Building healthy recommendation sequences for everyone: A safe reinforcement learning approach," in *FAccTRec Workshop*, 2020.

[39] X. Lu, L. Xiao, G. Niu, X. Ji, and Q. Wang, "Safe exploration in wireless security: A safe reinforcement learning algorithm with hierarchical structure," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 732–743, 2022.

[40] F. Zhao, Y. Zeng, B. Han, H. Fang, and Z. Zhao, "Nature-inspired self-organizing collision avoidance for drone swarm based on reward-modulated spiking neural network," *Patterns*, vol. 3, no. 11, 2022.

[41] Z. Zhao, J. Xun, X. Wen, and J. Chen, "Safe reinforcement learning for single train trajectory optimization via shield sarsa," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 412–428, 2022.

[42] P. S. Thomas, B. C. da Silva, A. G. Barto, S. Giguere, Y. Brun, and E. Brunskill, "Preventing undesirable behavior of intelligent machines," *Science*, vol. 366, no. 6468, pp. 999–1004, 2019.

[43] J. Yin, D. Chen, and L. Li, "Intelligent train operation algorithms for subway by expert system and reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 6, pp. 2561–2571, 2014.

[44] H. Tang, Y. Wang, X. Liu, and X. Feng, "Reinforcement learning approach for optimal control of multiple electric locomotives in a heavy-haul freight train: A double-switch-q-network architecture," *Knowledge-Based Systems*, vol. 190, p. 105173, 2020.

[45] X. Wang, A. D'Ariano, S. Su, and T. Tang, "Cooperative train control during the power supply shortage in metro system: A multi-agent reinforcement learning approach," *Transportation Research Part B: Methodological*, vol. 170, pp. 244–278, 2023.
[46] X. Chen, X. Guo, J. Meng, R. Xu, S. Li, and D. Li, "Research on ato control method for urban rail based on deep reinforcement learning," *IEEE Access*, vol. 11, pp. 5919–5928, 2023.
[47] L. Ning, M. Zhou, Z. Hou, R. M. Goverde, F.-Y. Wang, and H. Dong, "Deep deterministic policy gradient for high-speed train trajectory optimization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 11 562–11 574, 2021.
[48] X. Lin, Z. Liang, L. Shen, F. Zhao, X. Liu, P. Sun, and T. Cao, "Reinforcement learning method for the multi-objective speed trajectory optimization of a freight train," *Control Engineering Practice*, vol. 138, p. 105605, 2023.
[49] G. Li, S. W. Or, and K. W. Chan, "Intelligent energy-efficient train trajectory optimization approach based on supervised reinforcement learning for urban rail transits," *IEEE Access*, vol. 11, pp. 31 508–31 521, 2023.
[50] L. Zhang, M. Zhou, Z. Li *et al.*, "An intelligent train operation method based on event-driven deep reinforcement learning," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 10, pp. 6973–6980, 2021.
[51] S. Su, W. Liu, Q. Zhu, R. Li, T. Tang, and J. Lv, "A cooperative collision-avoidance control methodology for virtual coupling trains," *Accident Analysis & Prevention*, vol. 173, p. 106703, 2022.
[52] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Kluwer Academic Publishers*, 2001.