Single Agent Robust Deep Reinforcement Learning for Bus Fleet Control

Yifan Zhang*

Abstract

Bus bunching still as a challenge for urban transit systems, due to the randomness of traffic condition and stochastic passenger demand. Traditional method address this issue using multi-agent reinforcement learning (MARL) under idealized loop-line environments. However, such settings often fail to capture realistic bus operations, which are typically governed by heterogeneous trip line, predefined timetables, fluctuating passenger demands, and varying fleet sizes[1]. In this study, we propose a novel single-agent reinforcement learning (RL) framework for bus holding control, which circumvents the data imbalance and convergence issues associated with MARL in (almost) real-world simulation. We build a bidirectional timetabled networks with dynamic passenger demand. Our key contributions focus on transforming the inherently multi-agent problem into a single-agent formulation by explicitly encoding categorical identifiers—such as vehicle ID, station ID, and time period alongside traditional numerical features like headway, occupancy and segment velocity together as the state representation. This feature-space augmentation allows the single agent to work out in a higher-dimensional space where only MARL algorithm only used to be. Analogous to projecting linearly inseparable inputs into higher-dimensional spaces to achieve separability. Furthermore, we introduce a structured reward function that incentivizes alignment with both uniform headway and scheduled departure intervals. Rather than penalizing headway deviation through exponential heuristics, our reward adopts a ridge-shaped function that rewards both inter-vehicle regularity and fidelity to scheduled intervals—reflecting operational objectives emphasized in public transit planning literature. Compared to benchmark methods, including the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) model, our modified soft actor-critic (SAC) approach achieves significantly more stable and higher rewards (-430k vs.-530k) under stochastic demand and travel-time conditions. Our results suggest that single-agent deep reinforcement learning, when informed by categorical structuring and realistic schedule-aware design, can effectively manage bus holding decisions in non-loop, real-world settings. This paradigm offers a robust and scalable alternative to those conventional MARL-based control frameworks, particularly in environments where agent-specific experience distributions are inherently imbalanced.

1 Introduction

Bus bunching is a common issue in modern urban public transportation systems. When two or more buses running on the same route traveling too closely, it will undermine service reliability and increase passengers' waiting time. Contrary to the simplistic explanation that buses bunch merely due to schedule deviations, real-world operations reveal a more nuanced mechanism: As rooted in

*Central South University, erzhu419@gmail.com, 204201048@csu.edu.cn

1. Fleet size in bus operations typically refers to the total number of buses allocated to a route or set of routes, determined by long-term factors such as annual budget, demand analysis, and vehicle condition assessments. In practice, this number remains relatively constant. In this paper, the term "varying fleet size" is used to distinguish our approach from loop-line settings where a fixed number of vehicles operate continuously. However, the essence is actually "varying crew number" or, more precisely, the number of buses in active service at any given time. This dynamic in-service fleet size better reflects real-world operations: during off-peak hours, only a few buses are running while others are parked at terminals with drivers resting; during peak hours, all buses and drivers are deployed to meet increased demand.

dynamic interactions between road traffic conditions and passenger demand patterns[1]. Specifically, bunching tends to emerge not directly at the peak of rush hours, but rather in the periods leading up to the following peaks. During these transitional windows, a sudden increase in passenger demand or a drop in average road speed can slow down a leading bus significantly—either due to longer boarding times or congestion-induced delays[2]. Meanwhile, the following bus, facing fewer passengers or less congestion, begins to catch up. If the trailing bus eventually overtakes and becomes the new leader, it then inherits the higher demand at subsequent stops, once again slowing down. This self-reinforcing process—where the new leader is continuously penalized by higher demand—results in both buses alternately slowing and accelerating relative to one another, locking into a tight formation[3]. Over time, such unstable headway dynamics degrade the entire service's temporal regularity.

This phenomenon highlights the importance of designing such control strategies that could have high speed response not only to current headway but also to demand and speed conditions that evolve second frequently. Traditional methods, and even many RL-based approaches, often fail to capture the asynchronous, locally-triggered nature of these interactions[3, 4]. Our work addresses this gap by proposing a single-agent deep reinforcement learning framework tailored to realistic, timetable-based bus operations.

Traditional approaches to mitigating bus bunching typically involve station-level control strategies such as schedule-based holding or headway-based regulation. While conceptually simple, these methods often assume idealized and static environments, making them brittle in the face of real-world uncertainties such as non-stationary demand[2], fluctuating traffic conditions, and heterogeneous bus schedules. More recently, MARL frameworks have been introduced to address the complex temporal dynamics of fleet coordination[5]. MARL allows each bus to learn its own policy based on local observations, and has shown promise in loop-line or circular route simulations where the number of vehicles and trips per vehicle are fixed throughout an episode.

However, MARL approaches exhibit limitations in realistic, especially for bidirectional, timetable-driven networks. Theoretically, they will suffer from severe data imbalance across agents (which prove to be true in experiments): buses that are only active during peak hours (e.g., the 13^{th} bus deployed only in the morning rush) have far fewer experiences than those which continuously operating in the fleet. This leading to unstable or degenerate results[4]. Additionally, the bidirectional and timetable-driven nature of real bus operations,where each trip is constrained by terminal stops and does not continuously loop like in loop-line settings, poses unique challenges for MARL. Unlike loop-line environment, where buses can accumulate sufficient rewards or penalties over multiple trips, bidirectional lines are frequently truncated by terminals, which limiting the ability of correct actions to receive adequate rewards or incorrect actions to be sufficiently penalized. On the other hand, MARL algorithms generally face credit assignment issues. The asynchronous and event-driven nature of bus operations—where buses make decisions independently and rarely arrive at stops simultaneously—means that the primary challenge in this domain is not traditional credit assignment but rather the inability to propagate long-term rewards effectively across truncated episodes[4]. MARL algorithms were originally designed for domains with synchronous multi-agent decision-making, like robotic soccer, where all 11 agents must simultaneously decide whether to pass, shoot, or defend at each timestep. In contrast, bus control (and traffic signal control) problems are inherently asynchronous: agents (buses) make decisions at irregular, event-triggered intervals, leading to sparse and temporally misaligned experiences. This fundamental difference further challenges the applicability and stability of MARL in realistic transit environments.

To this end, this paper makes four contributions. **First**, we develop a realistic bidirectional bus simulation environment to bridge the gap between theoretical RL research and practical transit deployment. Unlike the most commonly used loop-line settings in previous paper that assume fixed fleet sizes and even trip distribution among fleet, our environment works under real-world constraints such as time varying demand, asymmetric direction stop, stochastic traffic conditions, and rolling fleet ac-

2. In this paper, the non-stationarity is addressed by adding a time dimension to the state, allowing the agent to be aware of the current time period, and assuming that demand at the same stop and same time period follows the same distribution. In contrast, the traditional non-stationary problem refers to the case where, even at the same stop and same time, the demand distribution can slowly drift over time, and thus does not remain stationary.

tivation based on timetable triggers. **Second**, we propose a novel single-agent reinforcement learning (RL) framework that transforms multi-agent bus control problem into single-agent formulation. By explicitly embedding categorical features (e.g., vehicle ID, stop ID, and trip ID) concatenated with continuous features such as headway and segment speed, our model enables an agent to generalize across heterogeneous buses and temporal contexts. This design leverages the intuition behind high-dimensional feature mappings—analogous to kernel methods in machine learning—where linearly inseparable patterns become linearly separable once augmented with sufficient structural feature dimensions. **Third**, we introduce a new reward function that replaces common exponential-based[5] heuristics with a "ridge-shaped" structure, which simultaneously encourages uniform headways and adherence to the scheduled inter-vehicle departure intervals. This design draws upon principles from transit service planning literature[6], which emphasizes that the schedule itself is the core reason of service quality. **Fourth**, we demonstrate that SAC, even in its standard form, provides inherent robustness in stochastic and dynamically disturbed environments, consistent with recent theoretical findings that connect maximum entropy RL with robustness guarantees[7]. Extensive experiments confirm that our SAC-based approach outperforms the MARL baseline (MADDPG) by achieving lower variance and higher reward, particularly under peak-period stress scenarios.

The remainder of this paper is organized as follows: Section 2 reviews related work on bus bunching problems and reinforcement learning approaches in public transportation systems (especially for bus operation). Section 3 describes the problem formulation and simulation environment, including the bus operation model and passenger demand stochastic. Section 4 details the proposed method, including the reinforcement learning framework and reward formulation. Section 5 presents experimental results and performance comparisons. Finally, Section 6 concludes the paper and discusses future research orientations.

2 Related Work

2.1 Bus Bunching Mitigation Strategies

Bus bunching mitigation strategies have traditionally been classified into station-based and inter-station-based control approaches[6]. Station holding aims to regulate headways by delaying early-arriving buses, while interstation control adjusts cruise speeds or leverages traffic signal priority to maintain spacing[8, 9]. More recent efforts have explored integrated, hybrid strategies that combine both control dimensions. For instance,[10] proposed a multi-strategy system leveraging DRL to unify stop-level holding, speed guidance, and signal adaptation.

While such integration improves adaptability under complex conditions, it also introduces training instability, especially in high-dimensional or temporally asynchronous environments. Furthermore, much of the existing literature assumes closed-loop or loop-line configurations[11, 12], which oversimplify realistic bidirectional and timetable-governed operations. Our work addresses these limitations by designing a learning architecture compatible with flexible scheduling and dynamic fleet size.

2.2 Reinforcement Learning Approaches in Transit Systems

Reinforcement learning, particularly MARL, has been a prominent tool for optimizing transit operations under stochastic passenger demand and traffic fluctuations. Notably,[4] and[13] introduced asynchronous MARL frameworks capable of handling event-driven control through macro-actions, which significantly reduce policy horizon complexity. These efforts have been extended in hierarchical MARL studies such as[14], where high-level agents coordinate holding and acceleration actions under a decentralized paradigm.

Despite these advancements, MARL approaches suffer from scalability and data imbalance issues, especially when vehicles appear sporadically across time or operate under temporally sparse policies. This phenomenon, frequently observed during peak-only deployments or irregular schedules, leads to convergence instability and sample inefficiency[15]. Hierarchical and curriculum-based

RL frameworks[15, 16] have been proposed to address these issues, yet often require domain-specific design and expert priors.

By contrast, our work leverages a single-agent RL architecture with explicit encoding of categorical identifiers (e.g., time period, direction), enabling uniform policy learning across all vehicles regardless of deployment frequency. This transformation mitigates data imbalance and improves generalization without necessitating multiple cooperating agents.

2.3 Single-Agent Soft Actor-Critic and Robustness Guarantees

SAC has emerged as a state-of-the-art off-policy RL algorithm known for its entropy-maximizing objective, which balances exploration and exploitation[17]. In environments subject to noise and temporal disturbances, SAC has demonstrated superior sample efficiency and stability compared to deterministic policy gradients or Q-learning variants[7].

Recent theoretical work[7] has shown that SAC approximately solves a robust reinforcement learning objective of the form:

$$\max_{\pi} \min_{r \in \mathcal{R}, p \in \mathcal{P}} \mathbb{E}_{\pi,p} \left[\sum_{t} r(s_t, a_t) \right],$$

where the inner minimization spans a bounded uncertainty set of reward functions $\mathcal{R}$ and transition kernels $\mathcal{P}$. This equivalence indicates that entropy-regularized RL policies inherently exhibit robustness to bounded adversarial perturbations in both reward and dynamics.

Our approach builds on SAC, but modifies it by embedding categorical features then using in a highly stochastic bidirectional bus environment. By introducing time-varying origin-destination (OD) flows and Gaussian-distributed travel time disturbances, we aim to enhance the robustness of the policy we learned. Combined with a structured, ridge-shaped reward function that emphasizes headway regularity and timetable punctuality, our method not only achieves robust convergence but also aligns operational decisions with service quality objectives.

2.4 (robust) optimization-based approaches

Recent simulation-optimization based studies have explored a lot on bus scheduling and operational strategies, even on electric bus fleet management and rescheduling. For instance,[18] optimized a single-line electric bus fleet services by using skip-stop as the action, resolving the trade-off between efficiency and demand of passengers. Further,[19] explored hybrid transit services with modular autonomous vehicles, introducing flexible and demand-responsive service ideas. The robustness of schedule performance was also examined through vehicle-type selection and departure-time shifting in electric bus routes[20]. At the fleet level,[21] investigated replacement strategies for variability kinds of electric buses, particularly, the heterogeneity in vehicle characteristics. Besides, multi-line methods were proposed for single-line operations to enhance the stability of fleet headway and passenger service quality[22]. While these contributions provide valuable optimization frameworks, they largely rely on deterministic formulations, which may face challenges in transit environments with high uncertainty like fluctuating demand and travel time variability. To bridge this gap, we propose a reinforcement learning-based approach capable of robustly adapting to fluctuating demand and uncertain travel conditions.

Robust simulation-based optimization has also been investigated in multiobjective settings. For example,[23] proposed robust approaches for constrained multiobjective problems, while[24] studied biobjective robust optimization in unconstrained settings. These works highlight the relevance of robust optimization techniques for complex stochastic environments.

3 Problem Formulation and Simulation Environment

3.1 Bus Operation Model

This study focuses on a bidirectional scheduled bus corridor system, which more accurately reflects real-world transit operations compared to the widely adopted loop-line abstractions in the literature[4, 15]. The bus route consists of 22 stops in total: two terminal stops (terminal_up and terminal_down), and 20 intermediate stops denoted by $X01$ through $X20$. All passengers board and alight only at intermediate stops; terminals serve solely as trip start and end points.

The vehicle dispatching process follows a pre-defined timetable spanning a 13-hour operating window. Buses are launched every 360 seconds (6 minutes) in both directions. The upstream terminal (terminal_up) initiates its first departure at time 6:00 AM, while the downstream terminal (terminal_down) begins dispatching at time 180 seconds latter, namely 6:03 AM. This 180-second phase shift ensures an initial staggering between the two directions, simplifying the analysis of headway evolution by eliminating inter-directional interference.

The consistent 360-second dispatch interval is maintained independently in both directions, resulting in a symmetric offset schedule. This design provides a stable reference frame for evaluating dynamic changes in bus headways, which subsequently emerge solely from stochastic road conditions and passenger boarding activities—not from irregularities in the schedule itself. Notably, the timetable is fixed and not optimized for demand alignment; service level calibration with passenger flow patterns is beyond the scope of this study.

Upon reaching a dispatch time:

- If the designated terminal has idle vehicles (i.e., those that have completed a previous trip), one is selected and dispatched.
- If no vehicles are available, a new bus is instantiated to fulfill the scheduled trip.

Each vehicle operates a single trip from one terminal to the other, determined by the direction flag: if direction = 1, the vehicle travels from stop terminal_up to terminal_down, otherwise in reverse. After arrival, the vehicle enters a rest state at the terminal, making it eligible for reuse in subsequent trips. This dispatch mechanism naturally reflects operational realities such as dynamic fleet sizing, asymmetric headways, and peak-hour demand injection—conditions that are difficult to handle in traditional loop-based simulations.

The simulation ends (i.e., one full episode) when all trips in the timetable have been executed and all active vehicles return to their respective terminals.

3.2 Passenger Demand and Traffic Disturbance

Passenger demand and interstop travel speed are both modeled as temporally heterogeneous and realistically stochastic. We extract time-varying OD demand from `passenger_OD.xlsx`, which contains a separate OD matrix for each of the 13 operational hours in the simulation. Each matrix element μ_{ij} denotes the expected hourly flow of passengers from stop n_i to stop n_j. During simulation, passenger arrivals at each stop are modeled as Poisson processes with intensity $\lambda_{ij} = \mu_{ij}/3600$, representing per-second flow. Passengers are assigned to a direction based on the relative position of their origin and destination—for example, a trip from $N05$ to $N03$ corresponds to the downstream direction (direction = 0).

Road traffic conditions are governed by the file `route_news.xlsx`, which records the average speed between adjacent stops for each hour. During simulation, the actual travel speed for a segment is sampled from a Gaussian distribution with the given hourly mean and a standard deviation δ. This models real-world disturbances such as random congestion or unexpected slowdowns.

Figure 1 provides an aggregated view of both passenger volume and average link speed throughout the day. Two prominent demand peaks are clearly visible: the morning peak around 09:00 and the evening peak around 17:00. These align with urban commuting patterns and correspond to sharp declines in average link speed—suggesting that peak-hour congestion both increases passenger load

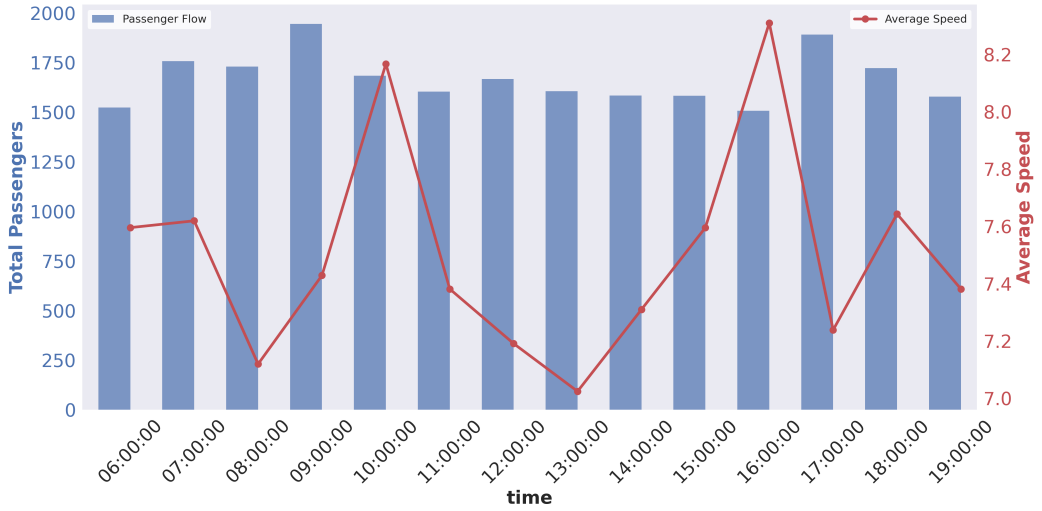

Figure 1: Visualization of Bus Peak/Off-Peak Periods: Passenger Flow vs. Average Speed

and simultaneously reduces travel velocity. Such a combination exacerbates the risk of bus bunching and reinforces the need for adaptive control strategies.

3.3 Decision Timing and State Transition

Unlike traditional methods where decisions are made immediately upon bus arrival at a stop (denoted $x_{i+1,j-1}$ in Fig. 2)[11, 16], our model explicitly delays decision-making to a more rational moment after all passenger boarding and alighting activities have completed. This moment is denoted $y_{i+1,j-1}$ in Fig. 2, aligning more closely with operational control opportunities in practice.

At $y_{i,j-1}$, the agent receives a state observation and outputs a control action. The environment then executes the action (e.g., holding), and at the next stop, after reaching $y_{j,i}$, the agent receives a reward based on the resulting system state (e.g., headway smoothness). This design allows the RL agent to respond to the most updated post-service information, leading to more stable and interpretable control decisions.

Figure 2 illustrates the holding decision process and headway computation using three sample buses traveling along a corridor. Each polyline represents the trajectory of an individual bus, where time progresses along the horizontal axis and station index increases along the vertical axis. The darkest bus symbol on each line denotes the most recently visited stop, while progressively lighter segments indicate historical stops previously visited. Holding durations are overlaid along each segment: *red* bars indicate the current holding time decision, and *orange* bars denote previous holding actions.

The dash line *red* rectangle in the figure highlights a critical region used to compute both forward and backward headways. Specifically, the horizontal time difference between $y_{i,j-1}$ and $y_{i+1,j-1}$ quantifies the *forward headway* h_f of bus i and simultaneously the *backward headway* h_b of bus $i+1$. This quantity becomes available only when **both** buses have completed dwell operations at the same station, ensuring that the headway reflects the realized temporal interval between two consecutive active vehicles.

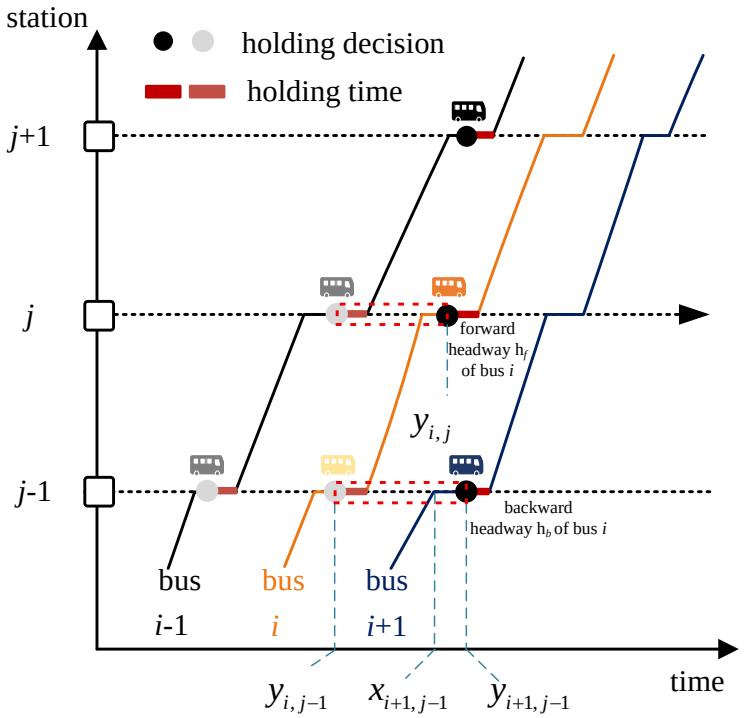

Figure 2: The timing of sequential decision and feedback.

4 Methodology

4.1 Components of Reinforcement Learning

Rather than assigning an individual policy to each vehicle, the system employs a single reinforcement learning agent that observes the operational context of each bus at decision points and outputs control actions accordingly. Unlike conventional time-stepped RL settings, decisions here are triggered by discrete events specifically, upon completion of passenger boarding and alighting at a stop. Let $x_{i,j}$ denote the time when bus i arrives at stop i, and $y_{i,j}$ be the moment when all passenger activities are completed. At time $y_{i,j}$, the agent receives a state observation $s_{i,j}$, obtains a reward $r_{i,j}$, and selects a continuous holding action $a_{i,j} \in [0, T]$. This action determines how many additional seconds the vehicle will holding at the stop after natural dwelling, with the goal of improving headway regularity. The next state $s'_{i,j}$ is only observed when bus i completes boarding and alighting at stop i, i.e., at $y_{i,j+1}$. The entire process is illustrated in Fig. 2.

State Representation. The state vector $s_{i,n} \in \mathbb{R}^d$ includes both categorical and numerical features. The categorical component consists of:

`[bus_id, station_id, time_period, direction]`

where `time_period` indicates the current hour (e.g., $t \div 3600$). These are encoded through learnable embedding layers to accommodate nonlinearity and improve generalization. The numerical features are:

`[forward headway, backward headway, current segment speed]`

which directly reflect local traffic and service level explicitly. This hybrid structure enables the policy network to incorporate heterogeneous information sources and handle data imbalance arising from irregular trip activation. As a common challenge in real-world systems where seldom vehicles are only service during peak periods, resulting in skewed experience distributions across agents. By

embedding categorical variables such as bus and trip identifiers, the single-agent framework generalizes across heterogeneous spatio-temporal contexts without requiring separate parameterizations for each vehicle instance.

Action Definition. While inter-station-based speed control could theoretically provide finer granularity in maintaining regular headways, we deliberately adopt station-based holding control as our primary action space. This choice is motivated by the following practical constraints observed in real-world bus operations:

- **Safety and Road Conditions**: Dynamic speed adjustment is constrained by urban traffic rules, varying road conditions, and safety requirements. Buses operating in mixed traffic environments cannot arbitrarily accelerate or decelerate.
- **Vehicle Inertia and Passenger Load**: Acceleration or deceleration behavior is affected by bus occupancy levels. Heavily loaded buses exhibit higher inertia, limiting their responsiveness to speed commands.
- **Action Uncertainty**: In realistic scenarios, instructions to modify speed are subject to interpretation and execution by human drivers, introducing uncertainty in action realization. This makes direct speed control less reliable than station-based holding.
- **Operational Preference**: Bus companies are more inclined to adopt stop-level holding strategies since these can be clearly communicated to drivers and executed without ambiguity. Many transit agencies already implement static holding as part of daily operations.

Therefore, the action in our reinforcement learning formulation corresponds to holding time at the current station. This is not only operationally feasible and enforceable but also aligns with industry best practices in mitigating bus bunching through minimal disruption to passenger experience.

So we set action $a_{i,j}$ as a scalar value sampled from a bounded continuous space $[0, T]$, where T is a preset upper limit (e.g., 60 seconds). It corresponds to the additional dwell time (holding) a bus will incur beyond the necessary passenger exchange duration. This continuous formulation contrasts with discrete or macro-action settings in prior, allowing finer control granularity[11, 8].

Reward Function Design The reward is designed to promote both symmetry in headways and schedule adherence. At each stop n, once a bus i completes its boarding process, it receives a scalar reward $r_{i,n}$ based on its h_f and h_b. The function consists of three components:

- **Schedule alignment**: A soft penalty proportional to the deviation from the nominal target headway of 360 seconds.
- **Headway symmetry**: A similarity bonus inversely proportional to the absolute difference between h_f and h_b, encouraging platoon balance.
- **Robust penalization**: Additional penalty is applied when either headway deviates from the target by more than 180 seconds.

Let

$$R(h_f, h_b) = \omega(h_f, h_b) \cdot \phi(h_f) + (1 - \omega(h_f, h_b)) \cdot \phi(h_b) - 0.5 \cdot |h_f - h_b| - 20 \cdot \mathbb{I}_{|h_f - 360| > 180 \text{ or } |h_b - 360| > 180}$$

where:

$$\phi(h) = -|h - 360|, \quad \omega(h_f, h_b) = \frac{|h_f - 360|}{|h_f - 360| + |h_b - 360| + \epsilon}$$

Here $\phi(\cdot)$ rewards headways close to 360, while ω dynamically weighs the forward and backward components. The final penalty term accounts for extreme outliers.

As visualized in Fig. 3, the reward surface exhibits a prominent ridge along the line $h_f = h_b = 360$, where vehicle spacing is both balanced and schedule-aligned. In contrast, deviations along the

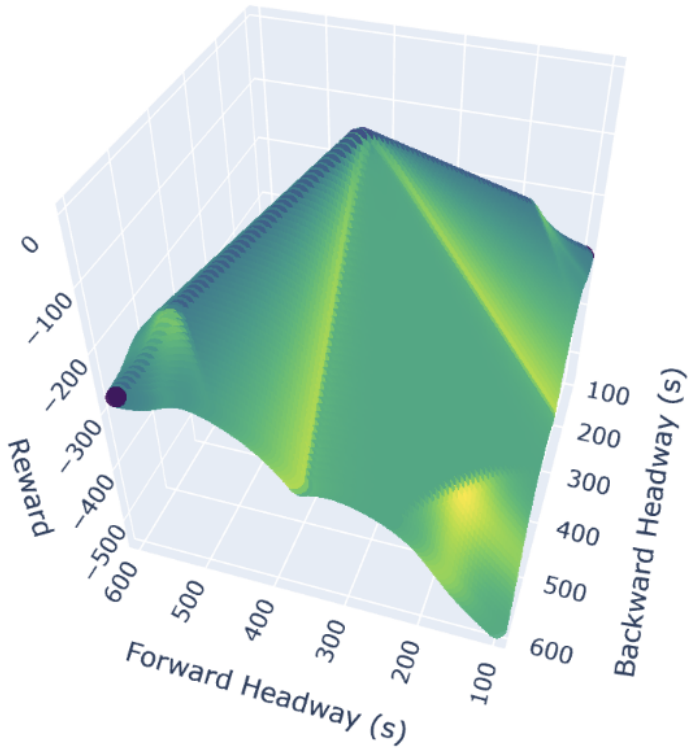

Figure 3: Visualization of the reward surface

direction $h_f \neq h_b$ produce sharp declines in reward due to penalized asymmetry. This geometry ensures that the agent is incentivized not only to maintain consistent spacing, but also to adaptively adjust dwell times based on real-time headway context.

Although many prior studies advocate inter-station cruise speed control[25, 10], we explicitly choose station-based holding in this work due to three pragmatic considerations rooted in real-world operations:

1. ***Limited Action Executability**: Speed control inherently requires the driver or vehicle control unit to continuously adjust velocity profiles in response to subtle policy outputs. In realistic bus operations, this is rarely feasible: driver response is delayed, bounded by road conditions, and strongly affected by onboard load-induced inertia. Consequently, the actual control action (e.g., reduce speed or accelerate by 10%) may not be reliably executed, introducing action uncertainty into the control loop.*

2. ***Safety and Regulatory Constraints**: Rapid speed modulation to maintain headway (especially deceleration) often conflicts with traffic safety rules and leads to passenger discomfort, which transit agencies are keen to avoid. This makes speed control politically and contractually unacceptable in many public transit settings.*

3. ***Holding as the De Facto Industry Norm**: In practice, holding buses at stations (especially terminal or major stops) is a widely accepted operational strategy, as it is easier to communicate, enforce, and integrate into static schedules. Agencies also prefer holding because it enables pre-emptive coordination (e.g., scheduled rest periods or crew shifts) without increasing perceived risk.*

Therefore, we model bus control as discrete holding actions only, to better reflect feasible and enforceable policy deployment in real-world bidirectional corridors. While inter-station control remains

a promising theoretical extension, its robustness under human-in-the-loop and partially controllable systems deserves separate investigation.

Transition and Experience Buffer. Due to the asynchronous nature of events, transitions (s, a, r, s') cannot be immediately constructed. Instead, each agent maintains a temporary buffer indexed by its identifier. When a bus completes its stop interaction at $y_{i,j}$, it stores (s, a, r). Once the next state s' is available at $y_{i,j+1}$, the full transition tuple (s, a, r, s') is assembled and inserted into the global replay buffer. This design accommodates interleaving transitions from multiple agents and enables stable training in partially observable and event-driven environments.

4.2 Feature Representation and Embedding Network

To enable a unified policy to generalize across different vehicles, stops, and temporal contexts, we explicitly incorporate four categorical variables into the state space:

$$\texttt{[bus\_id, stop\_id, direction, time\_period]}$$

These features do not carry intrinsic numerical meaning but are critical for distinguishing instances with context-specific dynamics. Rather than one-hot encoding which would lead to high-dimensional sparse inputs, we adopt an embedding-based approach.

Each categorical variable is mapped to a dense, learnable vector through an embedding layer. Let c_j denote the value of the j-th categorical feature, and let $E_j \in \mathbb{R}^{n_j \times d_j}$ be the corresponding embedding matrix, where n_j is the number of unique categories and d_j is the embedding dimension (typically $d_j = \min(50, n_j/2)$). Then the embedded representation is:

$$\mathbf{e}_j = E_j[c_j]$$

All categorical embeddings are then concatenated:

$$\mathbf{e} = \texttt{concat}(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4)$$

This dense vector $\mathbf{e}$ is further concatenated with the numerical part of the state vector:

$$\mathbf{s} = \texttt{concat}(\mathbf{e}, h_f, h_b, \mathbf{v}_{\text{segment}})$$

where $\mathbf{v}_{\text{segment}}$ is the current route segment speed. We also experimented with including the speeds of all route segments as input features, but observed little improvement in performance.

The final input vector $\mathbf{s} \in \mathbb{R}^d$ is fed into both the Q-networks and policy network in the SAC framework. Fig. 4 illustrates this process.

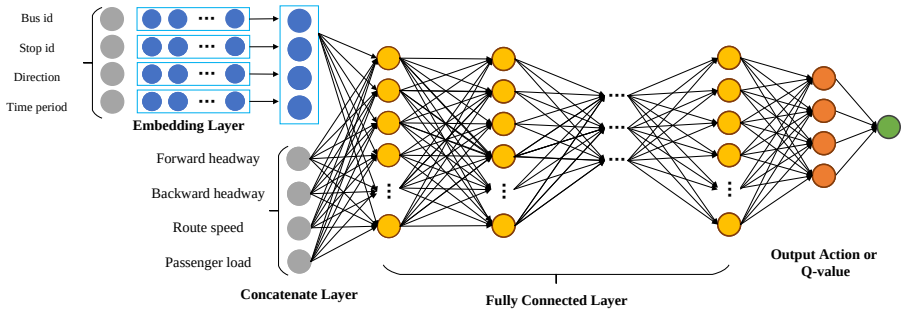

Figure 4: Network architecture

This architecture not only reduces the number of the agents, which reduce the dimension of the feature space indirectly, but also enables the network to capture semantic similarity between different context entities. For instance, two buses operating during the same peak hour or at adjacent stops can share latent representations because of the reuse of embeddings, improving generalization and training stability. The use of entity embeddings is particularly important in our single-agent setting, where data distribution is inherently imbalanced across different vehicle instances.

4.3 Soft Actor-Critic Framework

SAC is an off-policy deep reinforcement learning algorithm that integrates the maximum entropy principle into the actor-critic paradigm [17, 26]. Unlike traditional actor-critic algorithms that focus purely on maximizing cumulative reward, SAC augments the objective with an entropy regularization term. This encourages the learned policy to remain stochastic during training, which improves exploration and naturally leads to more robust behavior in environments with noise or partial observability.

Formally, SAC optimizes the following objective:

$$J(\pi) = \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right] \tag{1}$$

where $\mathcal{H}(\pi(\cdot|s_t)) = -\mathbb{E}_{a_t\sim\pi(\cdot|s_t)}[\log \pi(a_t|s_t)]$ is the policy entropy, and α is a temperature parameter balancing reward and entropy. In high variance stochastic environment setting, this structure is particularly useful for ensuring the policy does not collapse to greedy behavior too early during training, which is crucial given the asynchronous, unevenly sampled bus experience data.

To evaluate and update the policy, SAC relies on a soft version of the Bellman backup operator, which iteratively estimates the soft Q-function under the current policy:

$$Q^\pi(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p} \left[V^\pi(s_{t+1}) \right] \tag{2}$$

$$V^\pi(s_t) = \mathbb{E}_{a_t\sim\pi} \left[Q^\pi(s_t, a_t) - \alpha \log \pi(a_t|s_t) \right] \tag{3}$$

Here, Q^π is the soft state-action value function, and V^π is the soft state value function that includes an entropy adjustment. The role of the Bellman backup is to converge toward the true Q-values under policy π, forming the core target signal for critic learning.

The critic networks are trained by minimizing the following squared soft Bellman residual:

$$J_Q(\theta_i) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}} \left[\left(Q_{\theta_i}(s_t, a_t) - y_t \right)^2 \right], \quad \text{where} \quad y_t = r_t + \gamma V^\pi_{\bar{\theta}}(s_{t+1}) \tag{4}$$

In our implementation, two Q-networks Q_{θ_1} and Q_{θ_2} are trained simultaneously to reduce overestimation bias, and the target value V^π is computed using the smaller of the two Q-values and a slowly updated target network $\bar{\theta}$. This stabilizes the critic learning under highly variable headway dynamics and OD noise.

The policy $\pi_\phi(a|s)$ is modeled as a squashed Gaussian—a Gaussian distribution whose samples are passed through a tanh nonlinearity to enforce bounded actions. This is critical in our domain, where the action represents additional holding time and must lie within a predefined range.

Policy learning is performed by minimizing the expected KL divergence between the policy and a softmax over the Q-function, originally expressed as:

$$\pi^* = \arg\min_{\pi_\phi} D_{\mathrm{KL}} \left(\pi_\phi(\cdot|s) \;||\; \frac{\exp\left(Q_\theta(s, \cdot)\right)}{Z_\theta(s)} \right) \tag{5}$$

Here, the normalization constant $Z_\theta(s) = \int_A \exp(Q(s, a)/\alpha)\, da$ ensures that the exponentiated Q-function defines a proper probability density over the action space. Notably, $Z_\theta(s)$ is independent of the current policy π, and thus does not affect gradient-based updates.

By expanding the KL divergence and taking gradients w.r.t. π, this yields the policy loss:

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D}} \left[\mathbb{E}_{a_t\sim\pi_\phi} \left[\alpha \log \pi_\phi(a_t|s_t) - Q_\theta(s_t, a_t) \right] \right] \tag{6}$$

which is the standard SAC actor loss minimized during training. To avoid manual tuning of the temperature parameter α, SAC treats it as a learnable dual variable and updates it via dual gradient descent to enforce a target entropy. Specifically, α is optimized by minimizing:

$$J(\alpha) = \mathbb{E}_{a_t\sim\pi} \left[-\alpha \log \pi(a_t|s_t) - \alpha \bar{\mathcal{H}} \right] \tag{7}$$

where $\bar{\mathcal{H}}$ denotes the desired entropy level. This mechanism ensures that the policy maintains high stochasticity in the early stages of training and gradually becomes more deterministic as convergence is approached. The inner term encourages the policy to choose actions with high Q-values while maintaining sufficient entropy.

SAC is well-suited to our asynchronous, event-driven dispatching environment. It allows for off-policy updates using irregular, vehicle-dependent transitions and gracefully handles nonstationary feedback signals. In contrast to MARL methods that require dense and balanced interactions across agents, our SAC implementation can learn a single policy that generalizes over time, space, and bus identity.

4.4 Equivalence of SAC and Robust Reinforcement Learning

The primary challenge in our bus holding control problem lies in dynamic uncertainty: travel times between stops vary stochastically throughout the day due to congestion and irregular passenger demand. This introduces uncertainty not in the reward function itself, but in the transition dynamics. In reinforcement learning terms, this is fundamentally a *dynamically robust* decision problem—where the policy must perform well even under small, adversarial perturbations to the dynamics model.

Directly solving dynamic robust RL problems is often intractable, especially in continuous control settings. However, Theorem 4.2 in [7] provides a key insight: optimizing a maximum entropy RL objective with a modified reward function $\bar{r}(s_t, a_t, s_{t+1}) \triangleq \frac{1}{T} \log r(s_t, a_t) + \mathcal{H}[s_{t+1} \mid s_t, a_t]$ under the unperturbed dynamics p yields a lower bound on the robust objective under the true (perturbed) dynamics $\tilde{p}$. That is, instead of optimizing:

$$\max_{\pi} \min_{\tilde{p} \in \Delta_P} \mathbb{E}_{\pi, \tilde{p}} \left[\sum_t r(s_t, a_t) \right],$$

we optimize:

$$\mathbb{E}_{\pi, p} \left[\sum_t \bar{r}(s_t, a_t) \right],$$

which serves as a tight lower bound on the robust formulation.

When the agent has seen enough trajectories—such that the sample mean of $r(s, a)$ becomes close to constant across time—the Jensen gaps introduced in the theoretical derivation (see Eq. 9 in [7]) become small, and the lower bound becomes increasingly tight. This behavior is analogous to Bayesian optimization, where high posterior variance early in training warrants broad exploration, but posterior uncertainty shrinks as more samples are acquired. Similarly, the stochastic policy induced by SAC begins with high entropy (large α) to hedge against uncertainty in reward and dynamics, and gradually converges to a more deterministic policy as uncertainty reduces.

Of course, this analogy is not perfect—Bayesian posterior variance reflects epistemic uncertainty, while real-world bus dynamics include irreducible aleatoric uncertainty (e.g., signal jitter or human boarding times). Nonetheless, the mechanism aligns well: early-stage high entropy enables SAC to explore and tolerate dynamics variability, while later-stage entropy decay naturally aligns the policy with the most likely transition behavior.

By adopting SAC, we effectively optimize a provably grounded surrogate for the dynamic robust control problem we care about—without explicitly modeling worst-case perturbations. This makes SAC particularly well-suited to our setting, and empirically, it leads to greater stability than standard MARL baselines under the same simulation conditions.

5 Experimental Setup and Results

5.1 Experimental Setup and Hyperparameters

To evaluate the performance and robustness of our proposed holding control method under realistic transit conditions, we develop a custom discrete-event simulation environment that models bidi-

rectional bus operations with non-stationary(time-varying) demand and traffic uncertainty. The experimental setup is structured as follows.(You can find our resource code at https://github.com/erzhu419/Catetorical-Feature-SAC-in-bus-simulation.git)

5.1.1 Simulation Environment and Data Sources

The simulator is implemented in Python 3.8 and is designed to emulate dynamic bus fleet operations along a corridor with two terminal stations. Each simulation episode begins at 6:00 AM and runs until all scheduled trips in the daily timetable are completed.

Passenger demand and traffic conditions are synthetically generated following the modeling paradigm proposed in Zheng et al. [27]. Specifically:

- **Passenger demand** is defined by an hourly OD matrix stored in `passenger_OD.xlsx`. For each pair of stations and each time period, the matrix specifies the expected number of passengers boarding per hour. During simulation, passenger arrivals at each stop are sampled from a Poisson distribution with a rate equal to the OD entry divided by 3600, generating fine-grained second-by-second demand.
- **Traffic speed** is provided in `route_news.xlsx`, which contains the average speed (in m/s) for each inter-stop segment, varying by hour. Actual travel speed during simulation is sampled from a Gaussian distribution with the recorded mean and a fixed standard deviation of 1.5, capturing random perturbations in traffic conditions. This setup closely mirrors the uncertainty modeling approach in [27].
- **Route structure** is described in `stop_news.xlsx`, which defines 22 stops including two terminals and 20 intermediate stops. Each vehicle operates in a single direction per trip, either from terminal_down to terminal_up ("up" direction) or the reverse ("down" direction).
- **Timetable-driven dispatching** is defined in `time_table.xlsx`. Vehicles are dispatched from terminals according to direction-specific scheduled departure times. Once a vehicle completes a trip and returns to its terminal, it may either rest or be reused for a subsequent trip depends on if there is following trip.

This bidirectional, schedule-triggered structure generalizes the unidirectional simulation approach in [27], and allows modeling of peak vs. off-peak asymmetry, heterogeneous headway patterns, and dynamic fleet scaling.

5.1.2 Network Architectures and Hyperparameters

All models are implemented using PyTorch. The architectural and training configurations are:

Embedding-SAC

- **Embedding layers**: for each categorical variable (bus ID, station ID, time period, direction), a learnable embedding is constructed with dimension set as $\min(50, \lfloor N_i/2 \rfloor)$, where N_i is the number of unique values for category i;
- **Actor and critic networks**: both are 4-layer multilayer perceptrons (MLPs) with hidden sizes of [32, 32, 32], followed by task-specific output layers (mean/log-std for policy; scalar value for Q-function);
- **Activation function**: ReLU;
- **Optimizer**: Adam;
- **Learning rate**: 1×10^{-5};
- **Batch size**: 2048;
- **Target smoothing coefficient**: 0.005.

MADDPG (PS/NPS)

- **Actor and critic networks**: 3-layer MLPs with layer sizes $[64, 64, \text{output\_dim}]$ (where output_dim is determined by the action or Q-value dimension);
- **Activation function**: ReLU;
- **Optimizer**: Adam;
- **Learning rate**: 1×10^{-5} for both actor and critic;
- **Batch size**: 128;
- **Target smoothing coefficient**: 0.01 (Polyak averaging);
- **Parameter sharing**: In PS, all agents share a single actor and critic network, with agent IDs encoded as part of the input; in NPS, each agent trains its own independent actor and critic networks.
- **Exploration noise**: Gaussian noise with standard deviation 0.2, clipped to the action bounds;
- **Replay buffer size**: 1×10^6 transitions per agent.

Batch Size Selection. The batch size settings for MADDPG and SAC differ significantly because the agents in MARL algorithms often share the data in replay buffer they collected based on each one's experiences. But in SAC, there is only one agent and one buffer. Every experience is stored in this single buffer, and the batch size is determined by the total number of experiences available. To maintain the same number of available transition tuple for every single agent, which prevents any single agent from being overwhelmed by the experiences of others, the batch size is significantly varying from the single agent RL and the MARL algorithms.

In contrast, SAC employs a shared replay buffer across all buses, which allows it to sample from a larger and more diverse dataset. This design enables SAC to leverage experiences from other buses in the same fleet, improving generalization and stability. However, the most relevant data for SAC still comes from experiences associated with the same bus ID and station ID as the current decision point. Given that the fleet size is approximately 20 (varying slightly with traffic conditions), dividing the SAC batch size by the fleet size provides an effective per-bus batch size comparable to MADDPG's setting. For example, with a SAC batch size of 2048, the effective per-bus batch size is approximately $\frac{2048}{20} \approx 128$, aligning with MADDPG's batch size.

This approach balances the benefits of global experience sharing with the need for local relevance, ensuring that SAC maintains both robustness and specificity in its policy updates.

5.2 Performance Comparison

Figure 5 presents the training reward trends of the proposed Embedding-SAC method versus two variants of MADDPG: one with parameter sharing (MADDPG-PS) and one without (MADDPG-NPS). To better reveal convergence dynamics and stability, we display both 10-episode rolling means (solid lines) and exponentially weighted moving averages (EWM, dashed lines), along with shaded areas indicating the corresponding standard deviations. A horizontal dashed gray line is added to represent the average reward of the uncontrolled case, serving as a performance baseline for comparison.

Several key observations can be made from the results:

(1) Convergence Speed Embedding-SAC rapidly achieves stable high reward within approximately 30 episodes. In contrast, MADDPG-PS and MADDPG-NPS require over 150 and 180 episodes, respectively, to reach convergence. This faster ascent is accompanied by significantly reduced variance, suggesting stable and sample-efficient learning.

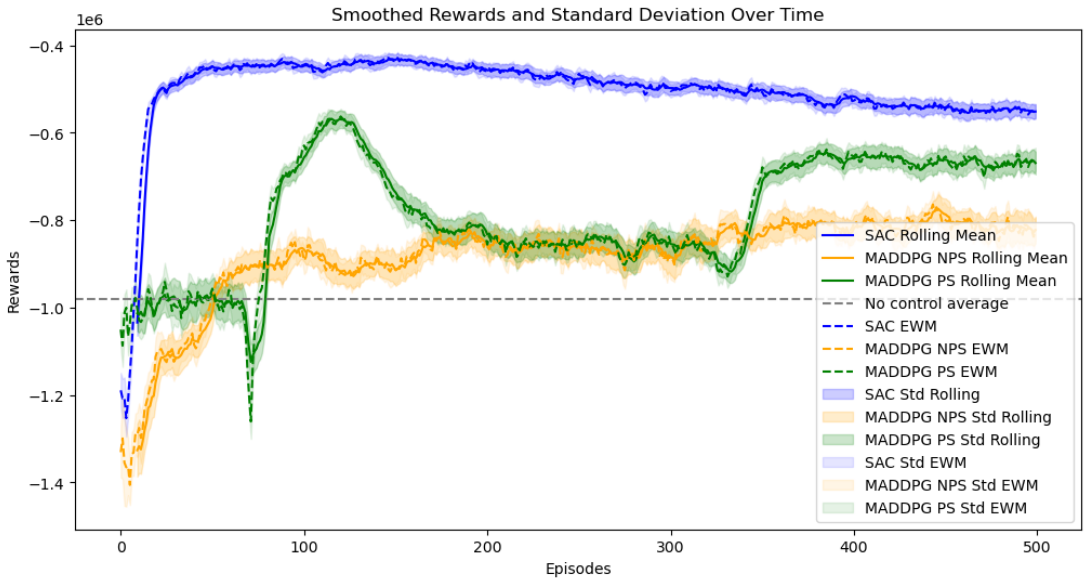

Figure 5: Smoothed training rewards and standard deviations of Embedding-SAC and MADDPG variants. Solid lines: 10-episode rolling means. Dashed lines: exponentially weighted moving averages (EWM, $\alpha = 0.3$). Shaded areas: ± 1 standard deviation over 15 evaluation rollouts. Gray dashed line: average reward under uncontrolled dispatch.

(2) Asymptotic Performance Embedding-SAC consistently maintains the highest mean reward throughout training, plateauing near ~ -430k. MADDPG-PS stabilizes around ~ -530k and MADDPG-NPS around ~ -580k. The uncontrolled baseline (gray dashed line) sits near ~ -980k, indicating that all methods improve upon the no-control scenario, but SAC achieves a substantially larger reward margin.

(3) Training Stability Shaded regions around each curve reflect the standard deviation of cumulative rewards from 15 independent evaluation episodes. Embedding-SAC exhibits the narrowest bands, especially in later stages, confirming strong policy robustness against environmental randomness. MADDPG variants suffer from wider fluctuations, with MADDPG-NPS showing instability and degradation after mid-training.

(4) Parameter Sharing Effects MADDPG-PS benefits from shared parameters, leading to faster early-stage improvement compared to MADDPG-NPS. However, its inability to capture agent heterogeneity under bidirectional and timetable-driven deployment prevents it from achieving SAC-level rewards.

(5) Variance-Sensitive Architecture Embedding-SAC's success lies in its single-agent architecture enhanced with categorical embeddings for vehicle ID, station ID, time period, and direction. This encoding strategy compensates for uneven exposure to training samples, particularly during peak-hour imbalances, and facilitates generalization across dynamic spatiotemporal contexts.

(6) Rationale for Not Using Waiting Time Passenger waiting time, though common in public transit metrics, is excluded here. As emphasized by Ceder [**?**], waiting time is more influenced by the timetable-demand alignment than by holding interventions. Since our method focuses on intra-trip headway stabilization, cumulative reward and headway-based variance are more indicative of control quality.

(7) Realistic Data Generation Simulated demand and road conditions are derived from real-world patterns. OD-based passenger arrivals follow Poisson processes updated hourly, while segment-wise travel speeds fluctuate according to time-varying Gaussian distributions, consistent with [27].

Conclusion: Embedding-SAC outperforms all MADDPG baselines and the no-control benchmark by a large margin. Its reward gain, stability, and convergence efficiency highlight the strength of combining soft actor-critic methods with categorical feature embedding in realistic, stochastic transit environments.

5.3 Empirical Evidence of Bus Bunching and Control Effectiveness

To assess the effectiveness of our proposed SAC-based holding strategy and to provide visual evidence of how bus bunching occurs and propagates, we present a series of trajectory-based and statistical analyses.

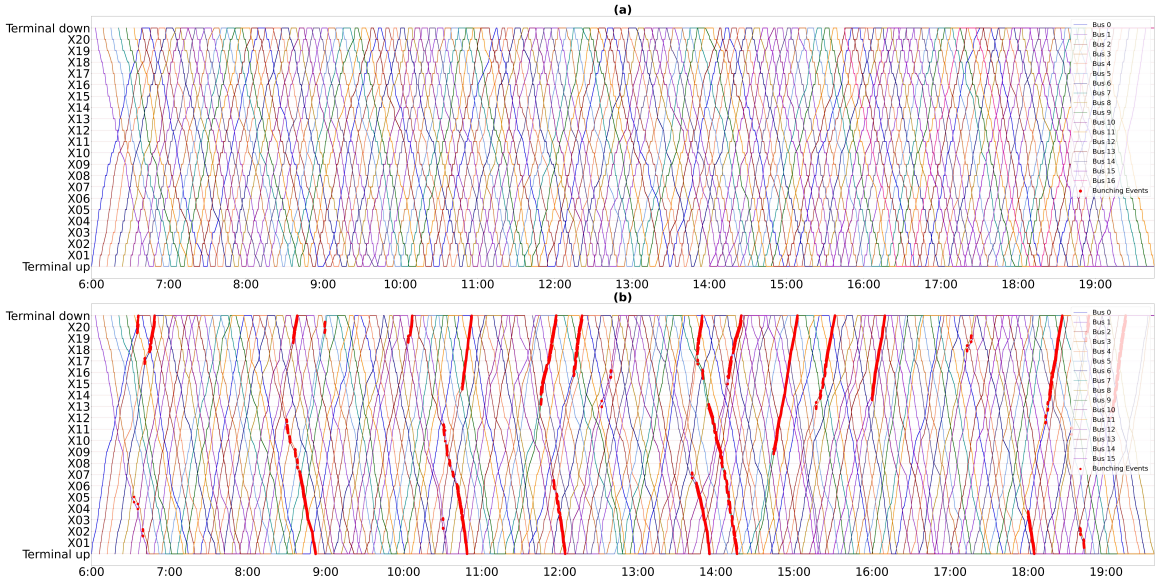

Figure 6: Bus trajectory visualization with and without SAC-based control. Top: (a) Bus trajectories with SAC control. Bottom: (b) Bus trajectories without control. Red markers indicate detected bunching events.

Figure 6 visualizes the complete operational period (*06:00–19:00*) for both the controlled and uncontrolled scenarios. Each colored line depicts the trajectory of an individual bus as it travels between terminals, running between all 22 stops. In the uncontrolled scenario (b, bottom panel), numerous bunching events are evident—highlighted by *red* segments—indicating intervals where two or more buses operate in close succession. These bunching occurrences are most prevalent during the early morning (*06:30–08:30*) and late afternoon (*16:00–18:00*) periods, immediately preceding the peak demand hours.

By contrast, the controlled scenario (a, top panel), where SAC governs holding decisions at stops, exhibits *no bunching events* throughout the episode. Bus trajectories remain well-spaced, confirming the robustness and effectiveness of the proposed control strategy under stochastic demand and travel speed conditions.

Figure 7 provides a fine-grained summary of the spatio-temporal distribution of bunching events based on the uncontrolled case. Subplots (a) and (b) identify the top 7 stops most frequently involved in bunching, categorized by travel direction. These are typically downstream stops near the end of each route, where accumulated delays due to traffic and passenger load cause buses to converge. Subplots (c) and (d) show hourly bunching frequencies in both directions. The risk of bunching peaks during transitional hours—especially *07:00–09:00* and *16:00–18:00*—which precede

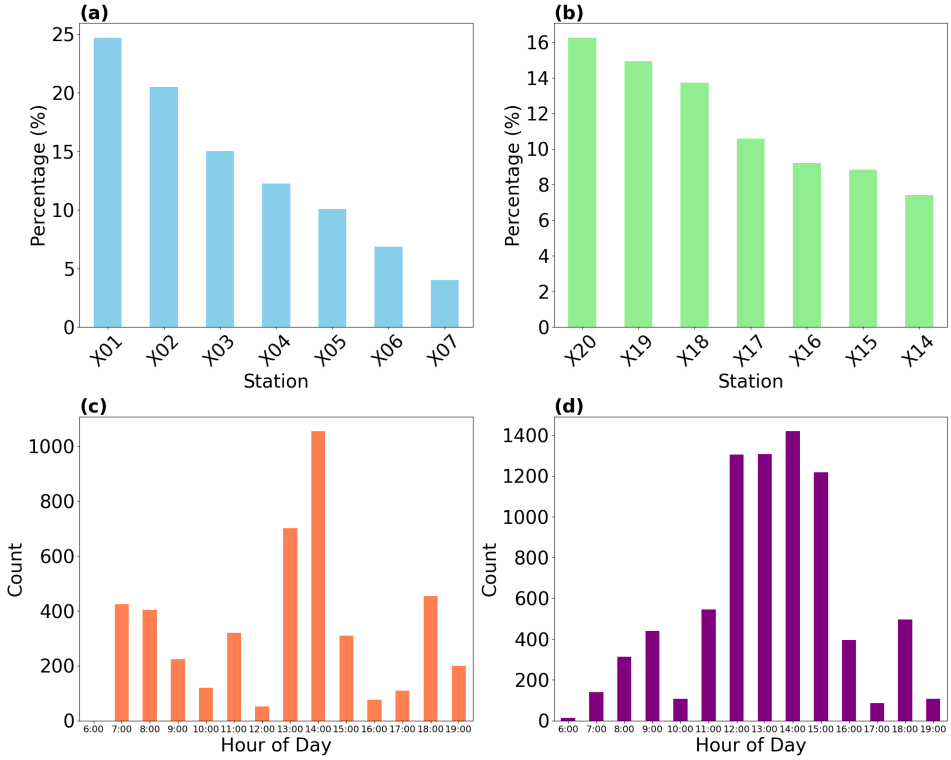

Figure 7: Spatio-temporal analysis of bunching events. (a) Top 7 bunching-prone stations (down direction). (b) Top 7 bunching-prone stations (up direction). (c) Hourly bunching frequency (down direction). (d) Hourly bunching frequency (up direction).

the full onset of rush hour demand. This supports our earlier hypothesis that bunching is not strictly a peak-hour phenomenon, but rather the result of demand-speed imbalances building up in the periods preceding peak load.

Together, these findings reinforce the key design motivations of our work:

- Bunching tends to arise in *pre-peak and transitional periods* due to sudden local demand or speed perturbations;
- Spatially, bunching accumulates *near the end* of each trip when the bus has been exposed to long sequences of stochastic disturbances;
- A *control strategy that integrates spatio-temporal context*—as done via SAC with categorical embedding—can effectively eliminate bunching under high variability.

These results validate the proposed modeling and control approach, and demonstrate that a well-trained single-agent RL policy can generalize across dynamic fleet states and operational conditions.

6 Conclusion

This paper presents a novel single-agent reinforcement learning framework for mitigating bus bunching in realistic, bidirectional, timetabled transit systems. Departing from conventional multi-agent approaches designed for idealized loop-line environments, our method explicitly incorporates categorical embeddings—such as vehicle ID, station ID, and trip identifiers into a SAC policy. This enables a single agent to generalize across heterogeneous agents and spatio-temporal contexts, overcoming the limitations of data imbalance and agent-specific training instability inherent to traditional MARL settings.

To emulate operational realism, we construct a bidirectional bus simulation environment based on empirically derived passenger demand and route speed profiles. The environment features a timetable-driven dispatch mechanism and stochastic variability in both road conditions and boarding/alighting dynamics. Such setup extending and refining the methodology proposed by Zheng et al. [27]—supports more faithful modeling of real-world operational challenges.

Through extensive experiments, we demonstrate that Embedding-SAC achieves better performance in terms of cumulative mean reward value, asymptotic stability, and variance of cumulative reward value compared to both parameter-sharing and non-sharing version of MADDPG. Notably, our control policy eliminates all bunching events across the whole simulation period, as confirmed by trajectory-level visualizations and spatio-temporal heatmaps. These results validate that a well-structured single-agent policy, when used appropriate embedded features, can effectively accomplish holding decisions under uncertainty.

Beyond technical contributions, this study offers practical meaning: bunching always not happen in peak hours, but in transitional periods preceding them. Especially when schedule delays and passenger demand begin to accumulate at the same time. As this kind of accumulation growing, the bunching happens closer to the terminal segments, as the service close to the end of the day, which highlighting the importance of control strategies to prevent the bunching from occurring more frequently as the day progresses.

In future work, we plan to extend this framework to incorporate non-stationary demand distributions, transfer learning across corridors, and integration with upstream scheduling modules. We also intend to explore hierarchical and latent-graph models to further disentangle causal influences in bus dynamics. Ultimately, we believe that robust, scalable, and interpretable RL-based bus control systems can serve as critical enablers of smart city mobility.

References

[1] "Bus bunching - wikipedia," https://en.wikipedia.org/wiki/Bus_bunching, accessed: 2025-07-29.

[2] M. Rezazada, N. Nassir, E. Tanin, and A. A. Ceder, "Bus bunching: a comprehensive review from demand, supply, and decision-making perspectives," *Transport Reviews*, vol. 44, no. 4, pp. 766–790, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0144164724000035

[3] W. Wu, R. Liu, and W. Jin, "Modelling bus bunching and holding control with vehicle overtaking and distributed passenger boarding behaviour," *Transportation Research Part B: Methodological*, vol. 104, pp. 175–197, 2017. [Online]. Available: https://doi.org/10.1016/j.trb.2017.06.019

[4] J. Wang and L. Sun, "Reducing bus bunching with asynchronous multi-agent reinforcement learning," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*, 2021.

[5] ——, "Dynamic holding control to avoid bus bunching: A multi-agent deep reinforcement learning framework," *Transportation Research Part C: Emerging Technologies*, vol. 116, p. 102661, 2020.

[6] A. Ceder, *Public Transit Planning and Operation: Modeling, Practice and Behavior*, 2nd ed. CRC Press, 2016.

[7] B. Eysenbach, S. Levine, and C. Finn, "Maximum entropy rl (provably) solves some robust rl problems," in *International Conference on Learning Representations (ICLR)*, 2022.

[8] O. Cats, Y. Shiftan, and E. Ben-Elia, "Bus-holding control strategies: Simulation-based evaluation and guidelines for implementation," *Transportation Research Record*, vol. 2274, no. 1, pp. 100–108, 2012.

[9] Y. Bie, S. Zhang, and S. Gao, "Dynamic headway control for high-frequency bus lines based on speed guidance and intersection signal adjustment," *Computer-Aided Civil and Infrastructure Engineering*, vol. 35, no. 5, pp. 411–426, 2020.

[10] Q. Nie, J. Ou, H. Zhang, J. Lu, S. Li, and H. Shi, "A robust integrated multi-strategy bus control system via deep reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 133, p. 107986, 2024.

[11] C. E. Cortés, G. A. Pineda, M. Gutiérrez, V. Vargas, D. E. Delgado, and M. Paredes, "Hybrid predictive control for real-time optimization of public transport systems' operations based on evolutionary multi-objective optimization," in *Transportation Research Board 89th Annual Meeting*, 2010.

[12] O. Cats and J.-D. Glück, "Frequency and vehicle capacity determination using a dynamic transit assignment model," *Public Transport*, vol. 11, pp. 505–525, 2019.

[13] K. Menda, Y.-C. Chen, J. Grana, J. W. Bono, B. D. Tracey, M. J. Kochenderfer, and D. Wolpert, "Deep reinforcement learning for event-driven multi-agent decision processes," in *AIAA Scitech 2019 Forum*, 2019.

[14] M. Yu, T. Yang, C. Li, Y. Jin, and Y. Xu, "Mitigating bus bunching via hierarchical multi-agent reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 8, pp. 9675–9680, 2024.

[15] Y. Tang, A. Qu, X. Jiang, B. Mo, S. Cao, J. Rodriguez, H. N. Koutsopoulos, C. Wu, and J. Zhao, "Robust reinforcement learning strategies with evolving curriculum for efficient bus operations in smart cities," *Smart Cities*, vol. 7, no. 6, pp. 3658–3677, 2024.

[16] Y. Chen, X. Qian, and J. Zhao, "Robust nonlinear decision approach for online bus speed control under uncertainties," *Transportation Research Part C: Emerging Technologies*, vol. 118, p. 102735, 2020.

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[18] C. Tang, H. Shi, and T. Liu, "Optimization of single-line electric bus scheduling with skip-stop operation," *Transportation Research Part D: Transport and Environment*, vol. 117, p. 103652, 2023.

[19] C. Tang, J. Liu, A. Ceder, and Y. Jiang, "Optimisation of a new hybrid transit service with modular autonomous vehicles," *Transportmetrica A: Transport Science*, 2023.

[20] C. Tang, Y.-E. Ge, H. Xue, A. Ceder, and X. Wang, "Optimal selection of vehicle types for an electric bus route with shifting departure times," *International Journal of Sustainable Transportation*, 2023.

[21] C. Tang, X. Li, A. Ceder, and X. Wang, "Public transport fleet replacement optimization using multi-type battery-powered electric buses," *Transportation Research Record*, vol. 2675, no. 12, pp. 1422–1431, 2021.

[22] C. Tang, A. Ceder, Y.-E. Ge, and T. Liu, "Optimal operational strategies for single bus lines using network-based method," *International Journal of Sustainable Transportation*, 2020.

[23] L. Zheng, J. Bao, and Z. Tan, "Robust simulation-based optimization for multiobjective problems with constraints," *Annals of Operations Research*, 2024, online first.

[24] L. Zheng, J. Bao, C. Xu, and Z. Tan, "Biobjective robust simulation-based optimization for unconstrained problems," *European Journal of Operational Research*.

[25] C. F. Daganzo and J. Pilachowski, "Reducing bunching with bus-to-bus cooperation," *Transportation Research Part B: Methodological*, vol. 45, no. 2, pp. 267–277, 2011. [Online]. Available: https://doi.org/10.1016/j.trb.2010.06.005

[26] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, ser. PMLR, vol. 119, 2020, pp. 4912–4922. [Online]. Available: https://proceedings.mlr.press/v119/haarnoja20a.html

[27] S. Zheng, B. Chen, S. Wang, R. Liu, H. Tian, and X. Li, "Robust nonlinear decision mapping approach for online bus speed control under uncertainty," *Computer-Aided Civil and Infrastructure Engineering*, vol. 38, no. 9, pp. 1120–1139, 2023.