



Transferable reinforcement learning for demand-responsive transit scheduling via offline pre-training and online fine-tuning

Downloaded from: <https://research.chalmers.se>, 2026-10-10 18:53 UTC

Citation for the original published paper (version of record):

Bian, J., Ma, X., Yan, H. et al (2027). Transferable reinforcement learning for demand-responsive transit scheduling via offline pre-training and online fine-tuning. *Transportation Research, Part C: Emerging Technologies*, 194. <http://dx.doi.org/10.1016/j.trc.2026.106010>

N.B. When citing this work, cite the original published paper.



Transferable reinforcement learning for demand-responsive transit scheduling via offline pre-training and online fine-tuning

Jing Bian^a, Xiaolei Ma^a, Haoyang Yan^a, Hesham El-Sayed^b, Kun Gao^c,
Xiaohan Liu^{c,*}

^a School of Transportation Science and Engineering, Beijing Key Laboratory for Cooperative Vehicle Infrastructure System and Safety Control, Beihang University, Beijing, 100191, China

^b College of Information Technology, United Arab Emirates University (UAEU), United Arab Emirates

^c Department of Architecture and Civil Engineering, Chalmers University of Technology, Göteborg, SE-412 96, Sweden

ARTICLE INFO

Keywords:

Demand-responsive transit
Offline pre-training
Online fine-tuning
Bus scheduling
Reinforcement learning

ABSTRACT

Demand-responsive transit (DRT) offers flexibility in adapting to dynamic travel demands and improving resource allocation efficiency. However, existing scheduling models suffer from limited generalization and the cold-start problem when deployed in new operational contexts. This study adapts reinforcement learning to cross-region DRT scheduling by combining offline imitation learning with online proximal policy optimization in a transferable framework (TR-PPO). The proposed approach follows a two-stage learning paradigm. In the first stage, behavior cloning is conducted on extensive expert datasets to learn transferable dispatching patterns and establish an initial policy prior. By using relative spatial representations rather than absolute coordinates, the framework reduces dependence on absolute coordinates and improves cross-region feature transfer. In the second stage, the pre-trained model is fine-tuned online in the target region using PPO to improve policy adaptation to local spatiotemporal dynamics under limited-data conditions. Case study findings from real-world datasets in China demonstrate that TR-PPO outperforms the tested scheduling baselines, achieving total system cost reductions of 11.6%–27.7%. Across the tested within-city, single-source cross-city, and multi-source cross-city transfer settings, the transfer-based policies also achieve competitive performance relative to the other baseline algorithms. TR-PPO accelerates convergence during online fine-tuning, reducing the number of training episodes by 57.8%. Moreover, TR-PPO demonstrates strong data efficiency during online fine-tuning, achieving competitive performance even with only 25% of the target-region training data. The results indicate that the proposed framework can improve the adaptation efficiency of DRT scheduling policies across the tested source-target regions with different demand patterns and network structures.

1. Introduction

Amid rapid urbanization and rising public transport demand, conventional transit scheduling systems are increasingly constrained by highly stochastic travel patterns. These traditional systems are prone to resource underutilization and service delays, as their static

* Corresponding author.

E-mail address: xiaohanl@chalmers.se (X. Liu).

or fixed schedules cannot effectively adapt to the real-time fluctuations inherent in modern urban mobility (Costa et al., 2021; Park et al., 2021). To overcome these limitations, DRT can effectively depart from the fixed schedules and rigid routes of conventional bus services, enabling adaptive responses to flexible passenger demands and reducing waiting times (Li and Quadrioglio, 2010). A key challenge in realizing efficient DRT operation lies in the scheduling problem, which has remained a primary focus of research throughout the past decade. Existing studies largely fall into two methodological streams: optimization-based approaches and data-driven learning approaches.

The optimization-based approaches commonly adopt mixed-integer linear formulations to solve static vehicle scheduling problems (Liu et al., 2021; Zhang et al., 2021; Zhang et al., 2023). To enhance the practical relevance, researchers typically set optimization objectives aimed at minimizing both operational costs and passengers' waiting time (Liu et al., 2021). These models explicitly account for vehicle routing, travel time constraints, and passenger-vehicle assignment constraints. To handle dynamic demand, insertion heuristics (Qiu et al., 2014) and hybrid optimization–heuristic frameworks (Zheng et al., 2021) have been introduced. However, as the problem scale increases and requests arrive in real time, these approaches often face substantial computational burdens, which limit their applicability in large-scale urban operations.

The application of reinforcement learning (RL) to DRT scheduling has attracted increasing attention, owing to its strong capability in handling sequential decision-making tasks. A recent study proposed an RL-based approach to plan DRT buses' routes in real-time, where the intelligent agents derive high-quality decisions by leveraging the current system state (Wu et al., 2024a). Multi-agent RL has also been employed in this problem field, where each intelligent agent is responsible for distinct scheduling decisions. Through inter-agent collaboration and information sharing, they collectively optimize the operational efficiency of the entire public transportation system (Wu et al., 2024b). Overall, the introduction of RL methods can enhance the dynamic adaptability of scheduling solutions while simultaneously expanding the scale of solvable problems and improving computational efficiency.

RL is broadly separable into two major categories: online and offline RL. These two approaches exhibit notable differences in their solutions to DRT scheduling problems. In general, online RL improves policies through real-time interaction with the environment, whereas offline RL learns from pre-collected datasets without additional online exploration (Levine et al., 2020). From a deployment perspective, both online and offline RL face important limitations in DRT scheduling. Online RL trained from scratch is often data-inefficient and slow to converge, because extensive target-domain exploration is required before the policy can learn stable and effective dispatching patterns, with many early interactions producing low-quality decisions. Offline RL avoids risky online exploration, but its performance depends heavily on the coverage and quality of pre-collected data and may generalize poorly to unseen regions (Prudencio et al., 2024). In addition, most of the RL models in existing DRT studies were trained for a single region scheduling scenario and verified in a specific simulation environment or on a single route (Liu et al., 2022; Tong et al., 2023; Yan et al., 2022). When deployed in a new region, these models often face a fundamental trade-off: offline-trained policies may generalize poorly to different demand patterns and road network structures, whereas online adaptation from scratch suffers from cold-start inefficiency, slow convergence, and a strong dependence on target-region data. Consequently, their effectiveness may be considerably reduced in new operational settings.

Therefore, a practical DRT scheduler for cross-region deployment should meet three requirements simultaneously: real-time responsiveness to dynamic requests, low-cost and reliable deployment in new operational contexts, and sufficient flexibility to adapt to local demand patterns and road network characteristics. Existing optimization-based and RL-based approaches do not fully satisfy these requirements at the same time. To address these challenges, this study adapts and integrates established transfer-RL components into a DRT-specific framework for cross-region scheduling. In the offline stage, behavior cloning (BC) is used to learn an initialization policy from expert trajectories collected in a source region. In the online stage, the pre-trained model is adapted to a target region through proximal policy optimization (PPO)-based fine-tuning. Our contributions can be outlined as follows.

First, this study identifies limited target-region data and real-time feasibility requirements as the key barriers to knowledge transfer in cross-region DRT scheduling and integrates established transfer-RL components to address these challenges. The framework combines offline behavior-cloning pre-training with online PPO fine-tuning to reduce cold-start exploration and adapt source-region dispatching knowledge to the target region. Second, the contribution lies in adapting and integrating existing transfer-RL techniques for the DRT scheduling context. The framework integrates relative vehicle–request representations, feasibility-aware action masking, and transfer-oriented fine-tuning to accommodate regional variations in demand patterns, network structures, and operational constraints. Third, experiments based on real-world DRT datasets from multiple regions demonstrate that TR-PPO reduces system cost, accelerates convergence, and improves target-region dispatching performance under limited target-region data compared with RL, heuristic, and optimization-based baselines.

The paper is organized as follows. Section 2 summarizes current research on the DRT scheduling problems and relevant RL algorithms. Section 3 introduces the DRT scheduling problem and defines key concepts. Section 4 details the proposed transferable RL framework and the corresponding transfer procedure. Section 5 reports the experiments and the findings from the case study. Section 6 concludes the study and identifies promising future research avenues.

2. Literature review

2.1. DRT: concepts and service models

Over the past decades, DRT has evolved into a flexible and dynamic extension of conventional public transport, driven by diversified travel demands and progress in intelligent transportation systems. Originally designed to serve special populations, DRT was established to provide mobility solutions for transportation-disadvantaged groups. Psaraftis (1980) first studied the real-time dial-

a-ride problem, which constitutes foundational research in early-stage DRT system optimization. Subsequent investigations established DRT as an innovative mobility alternative that effectively supplements conventional public transit systems, particularly in addressing the travel challenges inherent to low-density rural and suburban environments (Ryley et al., 2014). Specifically, the DRT service model can be categorized into three types: fixed routes with flexible stops (Liu et al., 2021; Quadrifoglio et al., 2007; Zheng and Li, 2019), demand-responsive feeder buses (Djavadian and Chow, 2017; Li and Quadrifoglio, 2010; Nickkar et al., 2022), and regional demand-responsive buses (Charisis et al., 2018; Lee et al., 2021, 2022; Yoo et al., 2024).

Flexible buses break the restrictions of the traditional buses' "fixed stops, fixed routes, and fixed schedules", and flexibly adjust the transport capacity and plan the routes according to passengers' demands and traffic flow. In such problems, fixed checkpoints and variable flexible stops are usually set for the vehicle routes. The vehicles will visit all the fixed checkpoints and then check whether to visit the flexible stops according to the set constraints. Fixed routes with flexible stops (Bruni et al., 2014; Zheng and Li, 2019) balance efficiency and flexibility by inserting real-time demands into pre-booked initial plans. However, this "static-plus-insertion" paradigm is inherently constrained by the backbone route, making it insufficient for large-scale scenarios where travel patterns are entirely random and lack a dominant corridor. Similarly, demand-responsive feeder buses focus on concentrated passenger flow and fixed service areas (Nickkar et al., 2022). The primary limitation of these studies lies in their regional dependency; the clustering and scheduling logic are often overfitted to specific spatial partitions, failing to provide a generalizable policy for cross-regional operations. In contrast, regional demand-responsive buses (Yoo et al., 2024) represent the most flexible mode, dynamically planning trajectories for many-to-many requests.

2.2. DRT scheduling approaches

Compared with conventional public transport, DRT offers superior flexibility in meeting personalized travel demands by optimizing service routes to minimize walking distances and waiting times. To clarify the research scope, the approaches for the bus scheduling problem are distinguished based on their operational environments: static or dynamic. Static scheduling assumes that all passenger demands are known a priori, allowing exact solution methods to derive the optimal solution (Fukasawa et al., 2006; Pecin et al., 2017). Their application to complex large-scale urban transit scenarios (e.g., those involving hundreds of pick-up and drop-off points) is often limited by prohibitive computational complexity and the inability to respond to real-time changes.

In contrast, modern DRT services are characterized by a fully dynamic and stochastic environment where requests emerge in real-time during vehicle operation. To handle this high degree of dynamism, heuristic algorithms and data-driven intelligent algorithms are employed to provide immediate, adaptive decision rules. While heuristic methods like dynamic insertion heuristics, genetic algorithms, and memetic algorithms (Du et al., 2024; Han et al., 2025; Li et al., 2022; Zhang et al., 2023; Zheng et al., 2021; Zheng et al., 2019) can efficiently navigate vast search spaces to generate high-quality solutions, they often lack theoretical guarantees for global optimality and are sensitive to hyperparameter tuning. Data-driven intelligent algorithms are capable of acquiring the intricate laws and decision-making patterns embedded in data through nonlinear mapping. RL has become an effective approach for these dynamic contexts (Habib and Filchenkov, 2022; Kalatzantonakis et al., 2023; Ng et al., 2024; Sun et al., 2025), as it can interact with the environment and adapt to uncertainty. However, existing RL models are typically region-bound and exhibit limited transferability when deployed in novel scheduling environments.

Exact solution algorithms primarily rely on mathematical modeling and optimization methods to find the optimal solution. Pecin et al. (2017) and Fukasawa et al. (2006) employed branch-and-bound integrated with column generation to tackle the vehicle routing problem with time windows. However, exact solution methods involve numerous constraints and high model complexity, leading to long solution times and limiting their scalability to large urban transportation scenarios.

To address the inefficiency of integer solvers in large-scale problems, heuristic methods have been developed to navigate complex search spaces more efficiently. Recent studies introduced the concept of meeting points (Zheng et al., 2019) and dynamic clustering (Zhang et al., 2023) to enhance system flexibility. Furthermore, sophisticated frameworks like the multi-population ant colony system (Du et al., 2024) have been proposed to balance operating costs with passenger satisfaction. To handle dynamic demands, two studies first collected static demands, used heuristic algorithms to generate initial route plans, and then inserted dynamic demands by adjusting the vehicle routes accordingly (Li et al., 2022; Zheng et al., 2021). Compared with exact solution methods, heuristic methods are more widely applicable and often navigate complex, large-scale search spaces more efficiently through stochastic or local search mechanisms. However, these metaheuristics lack theoretical guarantees for global optimality, and their performance is highly contingent upon hyperparameter configurations. This sensitivity often necessitates extensive parameter tuning for specific scheduling scenarios, limiting their robustness and direct transferability across diverse urban environments.

RL has become an effective approach to internalize intricate decision-making patterns within dynamic contexts. Recent research has focused on addressing convergence and adaptation challenges (Ng et al., 2024). For example, Habib and Filchenkov (2022) employed a two-phase training strategy: the first phase focuses on unconstrained graph navigation to minimize travel costs, while the second phase reintroduces operational constraints like vehicle capacity to refine the policy. Similarly, Kalatzantonakis et al. (2023) designed an integrated framework that combines RL with a variable neighborhood search metaheuristic. This approach utilizes an adaptive windowing algorithm to detect concept drift in the reward stream, allowing the model to dynamically adjust its operator selection strategy and generate high-quality planned routes in fluctuating environments. Nevertheless, existing RL models are typically region-bound, exhibiting limited cross-regional transferability due to their implicit coupling with specific road network structures.

2.3. Transfer learning and fine-tuning in RL-based mobility scheduling

Transfer learning has been widely studied to improve sample efficiency and accelerate adaptation in reinforcement learning. Existing transfer learning methods in RL can be broadly categorized according to the source-target relationship and the type of transferred knowledge. In agent-to-agent transfer, a source agent provides knowledge to a target agent that performs a similar task, usually with the same state representation, action space, and reward model. The transferred knowledge may take different forms, including shared experience samples, policy or parameter transfer, advice from an expert or another agent, learned representations, or meta-learned adaptation strategies. Experience sharing transfers agent-environment interactions to accelerate learning; policy transfer initializes the target agent with a previously trained policy; advice-based transfer provides action suggestions during learning; and meta-RL learns a policy or representation that can rapidly adapt to new tasks or environments.

Recent studies have increasingly explored knowledge transfer and online adaptation in reinforcement learning (RL), including autonomous driving, traffic signal control, multimodal mobility coordination, ride-sharing, and mobility-on-demand systems. In autonomous driving, RL-based fine-tuning has been used to improve the realism of simulated traffic participants and reduce collision risks (Peng et al., 2025). In traffic signal control, source-region data have been used to pre-train transferable policies, which are then adapted to target domains through online learning, such as the CrossLight framework (Sun, 2024). In multimodal mobility coordination, Hu et al. (2025) combine offline conservative Q-learning with online fine-tuning to improve the coordination between ride-sharing services and public transit.

More closely related to this study, recent ride-sharing and mobility-on-demand studies can also be interpreted through the above transfer-learning categories. Castagna and Dusparic (2021) presented an experience-transfer approach in multi-agent ride-sharing systems. In their setting, the source and target were different vehicle agents, and the transferred content consisted of selected agent-environment interaction samples, which were shared to accelerate learning in a large-scale ride-sharing simulation. Gammelli et al. (2023) presented a meta-RL and representation-transfer approach for cross-city AMoD control. They treated each city as a task and used a graph-based actor-critic framework to learn a policy that can rapidly adapt to unseen cities with limited target-city experience. Si et al. (2024) were more closely related to adaptive RL-based ride-pooling operations rather than transfer learning in the strict sense. They developed a multi-agent hierarchical RL framework for on-demand intercity ride-pooling, where idle vehicles were assigned to intercity lines at the upper level and vehicle routes were updated by an adaptive large neighborhood search heuristic at the lower level. Together, these studies show that transfer and adaptation in mobility systems can occur through shared experience, meta-learned policies, or hierarchical operational adaptation.

These studies indicate the potential of transferable RL for mobility systems, but request-level DRT scheduling introduces additional challenges that are less emphasized in generic transfer RL settings. First, DRT dispatching decisions must satisfy hard operational constraints, including vehicle capacity, pickup time windows, and maximum in-vehicle travel-time limits. Infeasible exploratory actions cannot simply be corrected through delayed reward penalties, because they may immediately violate service requirements. Second, cross-region DRT transfer involves changes in road network topology, spatial demand distribution, fleet availability, and active request patterns. These considerations motivate the design of TR-PPO, which combines source-region imitation initialization, relative feature encoding, feasibility-aware action masking, and target-region PPO fine-tuning for request-level DRT scheduling.

2.4. Summary

Existing studies have not fully addressed the need for DRT schedulers that are simultaneously real-time responsive, scalable, and transferable across regions. Optimization-based and heuristic methods remain constrained by computational burden or scenario-dependent parameterization, while RL-based methods are often tightly coupled with the data distribution and network structure of the training environment. Although transfer learning and offline-to-online adaptation have shown potential in other transportation domains, their application to DRT scheduling is still limited.

To address these gaps, TR-PPO is designed as a two-stage framework that first learns an initialization policy from source-region experience and then adapts that policy to a new region through online interaction. In this way, the method aims to improve not only cross-region transferability but also the practical efficiency of policy adaptation under limited target-region data and dynamic real-time operating conditions.

3. Problem definition

3.1. Service settings

DRT services are provided within a designated operating zone to meet flexible travel needs. In practice, demand-responsive bus services designate a set of fixed pick-up and drop-off points primarily to aggregate requests and standardize boarding and alighting locations, thereby improving routing efficiency and safety while keeping walking access within acceptable limits (Shaoting Public Transport Group Co., Ltd, 2025). Throughout this paper, we refer to these locations collectively as pick-up and drop-off points. Here, we assume that passenger requests are generated at pick-up points, and DRT buses pick up passengers at pick-up points. Each request can be described by the following attributes: pick-up point, drop-off point, service request time, maximum response waiting time, and maximum in-vehicle time. All DRT buses are assumed to depart from a single depot. As shown in Fig. 1, passengers arrive stochastically at different times throughout the day. They submit trip requests in real time and then wait for a DRT bus to be assigned in response. The travel time for each pair of consecutive points is derived from real-world travel data. Table 1 summarizes the

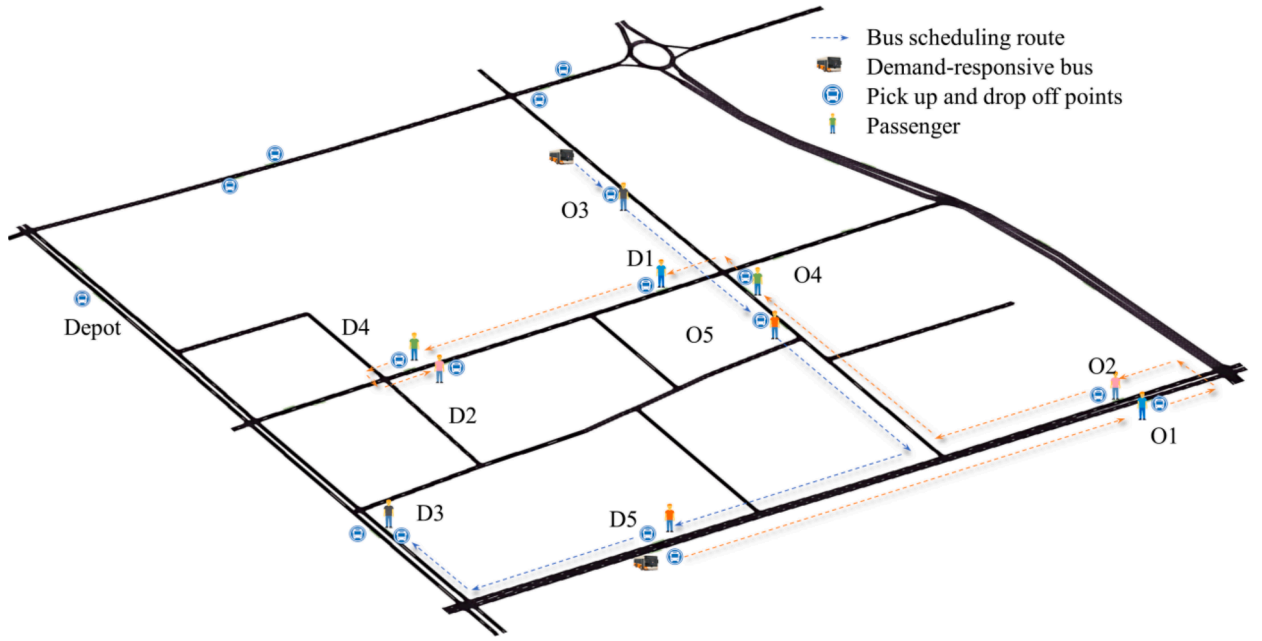


Fig. 1. DRT scheduling problem description.

Table 1
Definitions of key concepts.

Concepts	Definitions
Pick-up point	A predefined fixed pick-up location for DRT service.
Drop-off point	A predefined fixed drop-off location for DRT service.
Service request time	The time a passenger submits the request to the DRT system.
Request response time	The time when the DRT system responds to a request.
Pending request	A request is submitted by a passenger, but is not answered.
Maximum response waiting time	The maximum acceptable duration between the service request time and the request response time.
Removed request	A canceled request if the request is not responded to within the maximum response waiting time.
On-board request	A request that has been served by a DRT bus, with the passenger on board.
Unservd request	A request that has been assigned to a DRT bus but has not yet been picked up.
Sequence	An ordered list of pick-up and drop-off points for on-board and unservd requests assigned to a DRT bus.
Pickup waiting time	A passenger's pickup waiting time is defined as the duration from the service request time to the pickup time.

fundamental concepts defined in this study.

3.2. Scheduling mechanism

In the DRT dispatching system, let B , R , and T denote the set of DRT buses, passenger requests, and discrete time intervals, respectively. In this study, the length of each time interval is set to 1 second. A pending pool is maintained to store all requests that have arrived but have not yet been assigned to buses. The requests in the pending pool are ordered according to their service request times and are classified into two states: active requests and temporarily deferred requests.

Fig. 2 illustrates the flowchart of request assignment decision-making and bus route updating. At each time interval, the system first removes requests whose response waiting time exceeds the predefined threshold. Then, deferred requests are reactivated if their deferral time has reached 1 minute. Newly arriving requests are also added to the pending pool. If at least one active request is available, the controller evaluates the first active request in the pool according to the learned policy. Considering the computational cost of the insertion heuristic, at most one active request is evaluated and processed within each 1-second decision interval.

For each evaluated request, the controller selects one of two actions: assigning the request to a DRT bus or deferring the request. If the waiting action is selected, the request is temporarily deferred for 1 minute and excluded from policy evaluation during this period. This deferral mechanism prevents a difficult or temporarily unsuitable request from blocking the dispatching process, allowing the controller to continue serving other requests in subsequent decision intervals. If the request is assigned to a specific DRT bus, its origin and destination are inserted into the current unvisited stop sequence of the selected bus using an insertion-based heuristic (Jaw et al., 1986). Once assigned, the request remains fixed to that bus to ensure service consistency and avoid the computational complexity and passenger uncertainty associated with frequent reassignment.

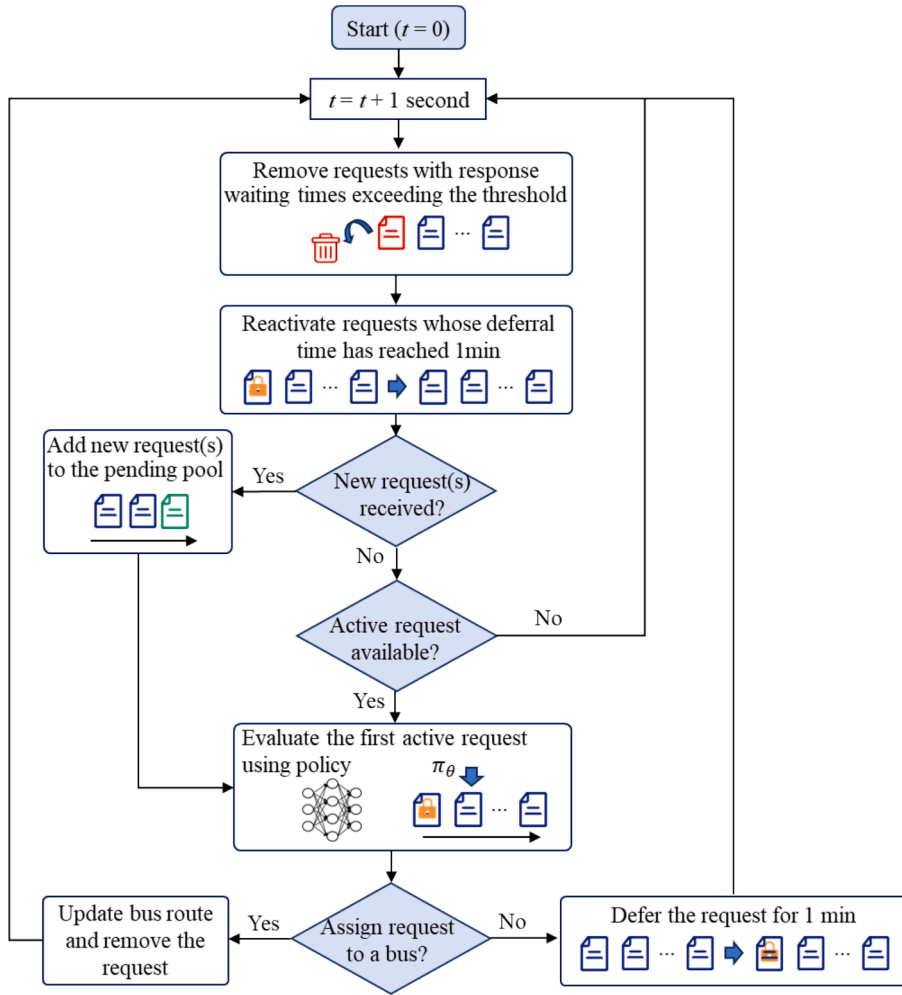


Fig. 2. Flowchart of request assignment decision and bus route update.

To ensure a clear and consistent mathematical formulation of the proposed TR-PPO framework, the key notations and symbols used throughout this section are summarized in [Appendix A](#).

3.3. System objective and operational constraints

The system objective is to minimize the total generalized cost, including vehicle travel cost, passenger travel cost, and penalties for requests that are removed. The objective function can be formulated as follows:

$$\min \alpha \sum_{b \in B} F_b + \sum_{d \in R_{acc}} (\beta_2 \omega_d + \beta_1 \delta_d) + \sum_{d \in R_{rem}} \beta_2 \Omega + \sum_{d \in R_{rem}} \gamma_d \quad (1)$$

$$n_{b,t} \leq v_b, \forall b \in B, t \in T \quad (2)$$

$$t_d^p < t_d^D, \forall d \in R_{acc} \quad (3)$$

$$\omega_d = t_d^p - t_d^{occ}, \forall d \in R_{acc} \quad (4)$$

$$\delta_d = t_d^D - t_d^p, \forall d \in R_{acc} \quad (5)$$

$$\delta_d \leq 1.5 \times T_d^{min}, \forall d \in R_{acc} \quad (6)$$

$$\omega'_d = t_d^{re} - t_d^{occ}, \forall d \in R \quad (7)$$

$$d \in \begin{cases} R_{acc}, & \text{if } \omega'_d \leq \Omega \\ R_{rem}, & \text{if } \omega'_d > \Omega \end{cases}, \forall d \in R \quad (8)$$

The first term in objective function (1) measures the vehicle travel cost. B denotes the DRT bus set. F_b signifies the total driving distance of bus b . Let α denote the unit vehicle travel cost. The second and third terms in objective function (1) represent the passenger travel cost, including all passengers' waiting time and in-vehicle time costs. Let R_{acc} and R_{rem} denote the set of responded requests and the set of removed requests, respectively. ω_d denotes the pickup waiting time, δ_d represents the in-vehicle travel time. Let Ω denote the maximum response waiting time. Let β_1 and β_2 denote the passenger in-vehicle time cost and waiting time cost, respectively. The last term introduces a penalty cost for removed requests, where γ_d denotes the penalty cost per request.

The operational constraints are defined as follows. Let $n_{b,t}$ denote the number of onboard passengers at time t of bus b . We use v_b to denote the rated passenger capacity of bus b . Constraint (2) requires that the number of onboard passengers must not exceed the rated capacity of bus b at any time. Let t_d^p and t_d^d denote the pick-up and drop-off time for request $d \in R_{acc}$, respectively. Constraint (3) indicates that for each responded request $d \in R_{acc}$, the pick-up location must be visited before its corresponding drop-off location. Constraint (4) defines the pickup waiting time of request d , where t_d^{oc} is the service request time of request d . Constraint (5) defines the in-vehicle time of request d . Let T_d^{min} denote the minimum in-vehicle travel time of request d . Constraint (6) indicates that for each responded request $d \in R_{acc}$, the in-vehicle time is restricted to within 1.5 times the minimum in-vehicle travel time. Constraint (7) defines the response waiting time of request d , denoted as ω'_d , where t_d^e denotes the request response time of request d . Constraint (8) states that request d is included in R_{acc} if its response waiting time is no greater than Ω ; otherwise, it is classified into R_{rem} .

4. Methodology

This section presents the technical details of the DRT scheduling framework proposed in this study. In the first part, we formulate the DRT service scheduling problem as a semi-Markov decision process (SMDP) model. We model the DRT scheduling system as an agent and propose a framework that is first trained on a source region and then fine-tuned on a target region using a small amount of local data. Finally, we introduce the implementation of a transferable RL framework in our work. It obtains the DRT scheduling system's decision pattern from offline data in the source region and utilizes a small amount of data from the target region to adjust the model. This approach allows the model to generalize and make reliable decisions even with limited data.

4.1. SMDP model

We formulate the DRT scheduling problem as a semi-Markov decision process (SMDP), in which a decision is made whenever the controller evaluates a request at a decision epoch. Although the system is updated at 1-second intervals, a decision epoch is defined only when the controller evaluates an active request. At each decision epoch, the agent observes the current system state, selects an action, and the system then evolves continuously until the next decision epoch.

4.1.1. State

At each decision epoch k associated with request d , the system state provides a snapshot of the current operating environment, which is represented as $S_k = (t_k, O_d, D_d, \{l_{b,k}, v_{b,k}, G_{b,k}, Y_{b,k}\}_{b \in B})$. Here, t_k denotes the system time at decision epoch k . Let O_d and D_d denote the origin and destination of request d . For bus $b \in B$, let $l_{b,k}$ represent the current location of bus b at decision epoch k . Let $v_{b,k}$ denote the remaining seat capacity at decision epoch k . To account for the vehicle's future service commitments, the capacity-related feature is defined as the normalized effective remaining capacity. Specifically, it is computed as the rated capacity of the bus minus the number of currently committed passengers. Here, currently committed passengers include both onboard requests and assigned but not yet served requests. This time-varying quantity is dynamically recomputed from the simulator state at each decision epoch, compressing the route workload into a standardized scalar feature for the network input. Let $Y_{b,k}$ denote the current planned visiting sequence of bus b at decision epoch k . To capture the service constraints of onboard passengers, $G_{b,k} = (g_1, g_2, \dots, g_{n_{b,k}})$ represents the remaining in-vehicle tolerance time of the $n_{b,k}$ passengers currently onboard bus b at decision epoch k . For onboard passenger i , the remaining tolerance is defined as follows.

$$g_i = 1.5 \times T_i^{min} - (t_k - t_i^p) \quad (9)$$

Where T_i^{min} denotes the minimum in-vehicle travel time of passenger i , and t_i^p represents the pick-up time for passenger i .

To encode the state, the origin and the destination of request d are included because they define the current decision to be made. By contrast, the variable-length route sequence $Y_{b,k}$ is not directly fed into the encoder; the service constraints of assigned passengers are summarized through their remaining in-vehicle tolerance times (Feng et al., 2021). This compact state representation reduces the complexity of the policy input while preserving the information most relevant to the current dispatching decision. To improve transferability across regions, the absolute coordinates of the origin and destination of request d are further transformed into relative spatial features $\Psi = \{\Delta dis_{bd}, \Delta \theta_b, \Delta \theta_d\}$ before being fed into the policy network. These features include the shortest network distance (Δdis_{bd}) between bus b and the current request d , the relative bearing ($\Delta \theta_b$) from the vehicle to its next scheduled stop, and the direction ($\Delta \theta_d$) of the OD pair of request d . If a bus has no scheduled stop, $\Delta \theta_b$ is set equal to $\Delta \theta_d$. The direction is discretized into eight

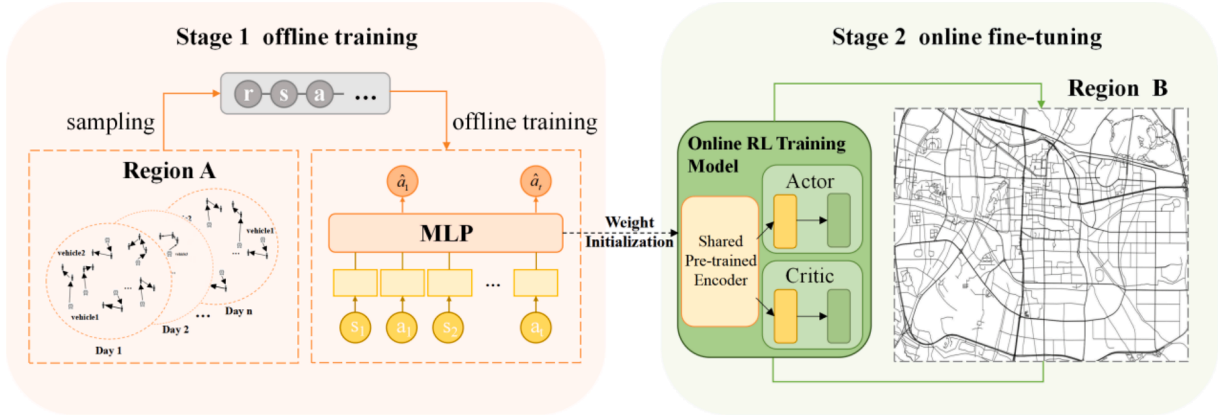


Fig. 3. TR-PPO framework.

sectors, each corresponding to a 45-degree interval. Finally, each bus-level state input is encoded by remaining capacity, remaining in-vehicle tolerance, and three relative spatial features.

4.1.2. Action

At each decision epoch k , the controller evaluates request d from the pending pool. The action space is defined as $A_k = \{0, 1, \dots, |B|\}$, where $a_k = b > 0$ denotes assigning the request to bus b , and $a_k = 0$ denotes deferring the request for later reconsideration. If the request d cannot be feasibly inserted into the route of any bus without violating capacity or in-vehicle time constraints, the only available action is to defer the request. Otherwise, the controller may either assign the request to a feasible bus or keep it in the pending pool.

In the case of an assignment action ($a_k > 0$), the system employs an insertion-based heuristic to evaluate the incremental distance of adding a new request to the existing route. Let $Y_{b,k} = (\varphi_0, \dots, \varphi_z)$ denote the current planned visiting sequence of bus b at decision epoch k . Here, φ_0 represents the current location of bus b , and z denotes the number of locations remaining to be visited. Let $\Phi_{b,k,d} = \{(i, j) : 0 \leq i < j \leq z, Y_{b,k}, O_d, D_d\}$ denote the feasible position set of insertions for request d , subject to the vehicle capacity constraints and in-vehicle time constraints of onboard passengers. Here, O_d is inserted after position i and D_d is inserted after position j . Let $\Delta_{b,k,d}(i, j)$ denote the incremental distance associated with inserting request d at positions (i, j) at decision epoch k . Then, the optimal insertion solution is determined by $(i^*, j^*) = \arg\min_{(i,j) \in \Phi_{b,k,d}} \Delta_{b,k,d}(i, j)$.

4.1.3. Transition

The SMDP transition $P(S_{k+1} = s', \tau_k = t_{k+1} - t_k | S_k = s, a_k = a)$ characterizes the joint probability of transitioning from state S_k to state S_{k+1} with sojourn time τ_k , given that action a_k is taken at decision epoch k . After action a_k is taken, the system evolves continuously until the next decision epoch $k + 1$. If request d is assigned, the corresponding bus route is updated using the insertion-based heuristic described above; if the request is deferred, the route remains unchanged, and the request stays in the pending pool. During the ensuing interval, each bus continues to move along its current route, and its location, occupancy, and the remaining in-vehicle time tolerance of onboard passengers are updated accordingly.

4.1.4. Reward

The reward received after taking action a_k at decision epoch k is formulated as below.

$$r_k = C_k - \alpha \cdot \sum_{b \in B} (dis_{b,k} - dis_{b,k-1}) - \beta_1 \left(\sum_{d \in R_k} it_{d,k} - \sum_{d \in R_{k-1}} it_{d,k-1} \right) - \beta_2 \left(\sum_{d \in R_k} wt_{d,k} - \sum_{d \in R_{k-1}} wt_{d,k-1} \right) \quad (10)$$

$$C_k = \begin{cases} 1, & a_k > 0 \\ 0, & a_k = 0 \end{cases} \quad (11)$$

Where C_k is a base reward; for each bus $b \in B$, $dis_{b,k}$ denotes the cumulative travel distance after action a_k at decision epoch k of bus b ; let R_k denote the set of requests present in the system after action a_k at decision epoch k ; $it_{d,k}$ represents the cumulative in-vehicle travel time of passenger $d \in R_k$ after action a_k at decision epoch k ; $wt_{d,k}$ denotes the cumulative pickup waiting time of passenger $d \in R_k$ after action a_k at decision epoch k . The base reward is defined as a conditional stimulus: $C_k = 1$ is awarded when the action results in a successful order assignment, whereas $C_k = 0$ is applied when the system selects a waiting action ($a_k = 0$).

The base reward C_k serves as a reward-shaping term that provides immediate feedback for successful service fulfillment, which helps improve exploration and stabilize policy learning (Ng et al., 1999). Therefore, the reward function is not intended to be an exact step-wise equivalent of the system objective in Equation (1), but rather an aligned training signal that combines service-fulfillment

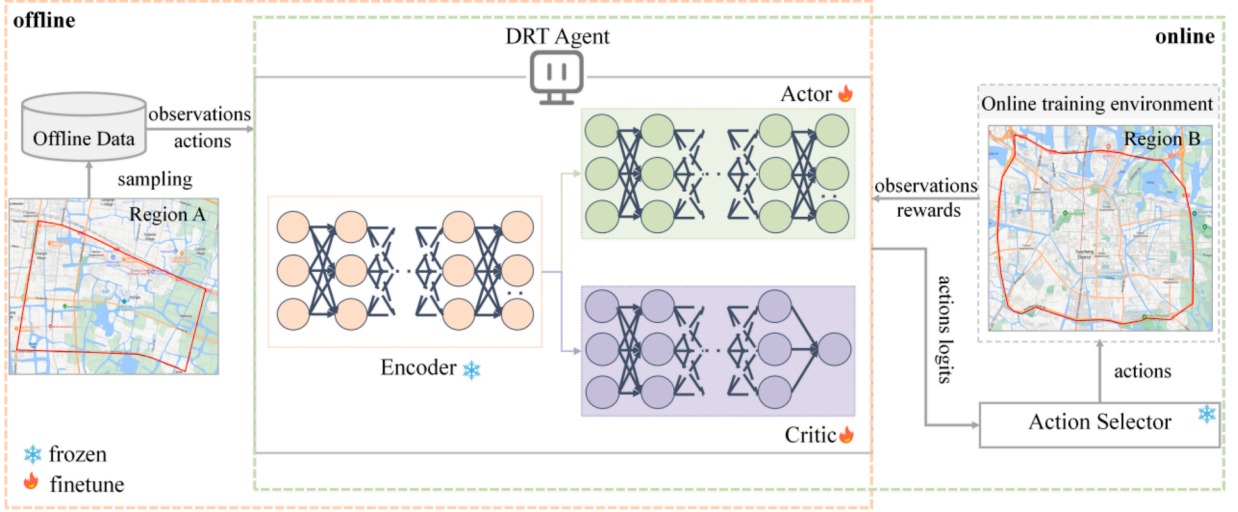


Fig. 4. Actor-Critic network architecture of TR-PPO.

incentives with incremental system-cost penalties. The weights α , β_1 , and β_2 are defined consistently with the system-level cost parameters used in Equation (1). In contrast, C_k is not an operating-cost parameter; rather, it is calibrated so that the positive reward for assigning a request is of the same order of magnitude as the typical marginal distance- and time-related costs incurred in serving that request.

4.2. TR-PPO framework

To solve the SMDP model in the cross-region DRT setting, we adapt and integrate behavior cloning with PPO fine-tuning into a DRT-specific transfer-RL framework, referred to as TR-PPO. As Fig. 3 illustrates, TR-PPO comprises two stages: an offline pre-training stage, where a state-action policy is learned from a pre-collected expert dataset in region A, and an online fine-tuning stage, where the pre-trained model from stage one provides the initialization for the online RL model.

4.2.1. Offline imitation learning via MLP-based behavior cloning

The pre-collected expert dataset is obtained from a simulation environment. The dataset is generated using a rule-based behavior policy π_β . This policy consists of a combination of a standard greedy insertion heuristic and stochastic exploration steps to ensure sufficient coverage of the state-action space. The behavior policy π_β is used only for data collection and initialization, rather than as the final target policy to be replicated. The subsequent learning process aims to improve upon this heuristic-driven policy by maximizing the expected cumulative reward. As part of the transfer framework, we use offline behavior cloning to initialize the policy before online PPO fine-tuning in the target environment. In the first step, we sample the expert data set from region A and abstract a series of trajectories $(s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_l, a_l, r_l)$. Here, s is the state of the DRT scheduling system, a denotes the DRT agent's action, r is the reward, and l denotes the episode horizon required to complete all scheduling instructions.

Subsequently, we employ a multi-layer perceptron (MLP) architecture comprising an encoder and a policy head as the function approximator for $\pi(a|s; \theta_{enc}, \theta_\pi)$. The encoder maps high-dimensional, redundant observations into a compact feature representation, while the policy head generates the corresponding action probability distribution. This decoupled design allows for independent component adjustment and feature reuse. During the pre-training process, BC is utilized to initialize the policy network by minimizing the Kullback–Leibler divergence between the current policy $\pi(a|s; \theta_{enc}, \theta_\pi)$ and the expert policy distribution $\pi_\beta(a|s)$, as formulated in Equation (12).

$$L_{BC}(\theta_{enc}, \theta_\pi) = \mathbb{E}_{s \sim d(s)} [D_{KL}(\pi_\beta(\cdot|s) \parallel \pi(\cdot|s; \theta_{enc}, \theta_\pi))] \quad (12)$$

Where $d(s)$ is the distribution of the state, and the notation $s \sim d(s)$ indicates that the state s is sampled from the distribution $d(s)$.

The network parameters are updated using the Adam optimizer, which provides superior convergence efficiency compared to standard stochastic gradient descent by dynamically adjusting learning rates based on first and second-order moments of the gradients. This stage yields a robust initial policy network, providing a warm start for the subsequent online fine-tuning phase.

4.2.2. TR-PPO for online fine-tuning

Building upon the knowledge captured during the offline pre-training stage, the model is subsequently transitioned to the online fine-tuning phase. While the offline policy provides a robust initialization of spatial-temporal embeddings, it may not fully capture the specific demand patterns and road network dynamics of the target region. Therefore, we adapt the framework using a specialized

online training procedure to ensure localized optimization and operational reliability.

During the online fine-tuning stage, an online reinforcement learning algorithm is deployed to interact with the stochastic and dynamic environment. We adopt Proximal Policy Optimization (PPO) due to its balance between training stability and sample efficiency (Foa et al., 2022). The optimization process is built upon the Actor-Critic framework, where the network architecture inherits specific components from a pre-trained BC model to leverage prior knowledge. Specifically, the actor network shares the identical spatial-temporal encoder and policy head structure. In the Critic network, a dedicated value component is integrated; it shares the same pre-trained encoder to extract high-level state representations but utilizes a newly initialized MLP head to output state-value estimations. To maintain the robustness of the learned representations while adapting to the target region, a selective freezing strategy is applied where the encoder's backbone remains frozen, while the policy and value heads are iteratively updated. The comprehensive structural transition and parameter mapping between the two stages are illustrated in Fig. 4.

The state space represents demand locations using the relative distance and heading of each request with respect to the vehicle, rather than absolute latitude and longitude coordinates. This relative spatial representation enhances the model's ability to generalize across different road network geometries. Given that the fundamental spatial-temporal patterns are captured during the offline stage, we employ a selective freezing strategy: the encoder's embedding layers are frozen to preserve the robust feature extraction capabilities, while the remaining policy and value heads are fine-tuned. This approach maintains a balance between knowledge retention and model flexibility, thereby facilitating faster adaptation to novel environments.

Following the data collection and RL optimization process, the model outputs the action logits Z , which represent the unnormalized preferences for each dispatching candidate. The logit vector is defined as follows:

$$Z = [z_0, z_1, \dots, z_{|B|}] \quad (13)$$

Where z_i represents the raw score assigned to each action within the full action space A .

To handle the high-dimensional action space and ensure operational feasibility, an action selector with a dual-masking strategy is implemented. At each decision epoch k , a binary mask vector $M_k = [1, m_1, \dots, m_{|B|}]$ is generated based on two primary constraints: (1) Capacity filter: vehicle b is masked if inserting the current request would violate the vehicle capacity constraint; and (2) Service guarantee filter: vehicle b is masked if inserting the current request would violate the passenger in-vehicle time constraint. The deferred action is always retained as a feasible fallback action. This masking process is integrated into the policy network by assigning negative infinity to the logits of invalid actions (denoted as $m_i = 0, \forall i \in B$) prior to the Softmax normalization. Consequently, the probability of selecting an infeasible action is reduced to zero.

$$\hat{z}_i = \begin{cases} z_i, & \text{if } m_i = 1 \\ -\infty, & \text{if } m_i = 0, \forall i \in B \cup \{0\} \end{cases} \quad (14)$$

Furthermore, while the offline pre-trained model provides a robust initialization, its performance may be sub-optimal when encountering the distribution shift inherent in online environments. During online fine-tuning, exploration is induced by stochastic action sampling from the masked PPO policy.

Conventional reward design for vehicle scheduling systems typically incorporates multiple metrics, including travel distance, duration, and overall cost. These metrics exhibit inherent correlations in their semantic representations. An excess of redundant evaluation metrics complicates the learning process and hinders convergence. The DRT system focuses on balancing passenger service quality with operational efficiency by optimizing a generalized cost-based reward function, as defined in Equation (10).

Within the Actor-Critic framework, the agent interacts with the environment through two synergistic components: the Actor, which employs a policy network $\pi(a|s; \theta_{enc}, \theta_{\pi})$ to map states to action distributions, and the Critic, which approximates the state-value function $V(s; \theta_{enc}, \theta_{vf})$ to evaluate the expected long-term return. Upon executing an action a_k , the environment transitions to the next state s_{k+1} and yields a scalar reward r_k .

The critic is optimized by minimizing the temporal-difference (TD) error loss, which quantifies the discrepancy between the estimated value and the observed transition:

$$L^V = \mathbb{E}_k \left[\left(r_k + \gamma V(s_{k+1}; \theta_{enc}, \theta_{vf}) - V(s_k; \theta_{enc}, \theta_{vf}) \right)^2 \right] \quad (15)$$

Where $\gamma \in [0, 1]$ is the discount factor; θ_{enc} is the frozen encoder parameters during online fine-tuning; θ_{vf} denotes the parameters of the value head. To refine the policy gradient while balancing bias and variance, we implement the generalized advantage estimation (GAE). The advantage function $\hat{A}_k$ is computed as a discounted weighted sum of future TD errors:

$$\hat{A}_k = \sum_{l=0}^{|K|-k-1} (\gamma \lambda)^l \delta_{k+l}, \delta_k = r_k + \gamma V(s_{k+1}; \theta_{enc}, \theta_{vf}) - V(s_k; \theta_{enc}, \theta_{vf}) \quad (16)$$

Where λ is the smoothing parameter. Finally, the actor is updated using the PPO clipping mechanism through gradient ascent. This objective function constrains the policy update step to prevent catastrophic performance collapse by clipping the probability ratio:

$$L^{CLIP} = \mathbb{E}_k \left[\min \left(\frac{\pi(a_k|s_k; \theta_{enc}, \theta_{\pi})}{\pi(a_k|s_k; \theta_{enc}, \theta_{\pi}^{old})} \hat{A}_k, \text{clip} \left(\frac{\pi(a_k|s_k; \theta_{enc}, \theta_{\pi})}{\pi(a_k|s_k; \theta_{enc}, \theta_{\pi}^{old})}, 1 - \epsilon_c, 1 + \epsilon_c \right) \hat{A}_k \right) \right] \quad (17)$$

Where ε_c is the hyperparameter of the clipping range; θ_π denotes the parameters of the new policy head; θ_π^{old} denotes the parameters of the old policy head, which is used at the beginning of the current data collection phase.

4.2.3. Overview of TR-PPO

Algorithm 1 illustrates the main components of the TR-PPO framework, which is structured into two sequential training stages: offline pre-training and online fine-tuning. During the offline stage, the trajectory data are collected from region A to construct a comprehensive dataset of transitions (s, a, r, s') . Utilizing the BC method, the policy network is trained to map these observations to action probabilities, effectively distilling expert dispatching logic into the model parameters θ . This pre-training phase provides a high-quality initial policy that achieves competent performance in real-time operational scheduling, thereby significantly expediting the subsequent reinforcement learning process by providing a robust starting point for exploration.

During the online stage, the pre-trained framework is adapted to the target region for localized fine-tuning. To handle the high-dimensional action space and ensure operational feasibility, the policy network employs a dual-masking strategy that filters out invalid actions based on vehicle capacity and spatiotemporal constraints. The architecture comprises a spatial-temporal encoder for generating latent embeddings and task-specific heads for action and value generation. To preserve the pre-trained representations, a selective freezing strategy is implemented: the low-level embedding layers of the encoder are frozen, while the policy and value heads remain fully trainable. The parameters of the Actor and Critic components are then iteratively updated using the PPO objective and TD-error as defined in Equations (15)–(17), allowing the model to specialize its decision-making logic for the target environment while maintaining stable gradient updates.

Algorithm 1: Offline pre-training and Online fine-tuning

Input:
 policy model π_θ ;
 source region data D_{source} ;
 environment E .
Output:
 fine-tuned reinforcement learning model parameters.

Step 1: Offline pre-training on source region data

1 **For** $epoch=1$ **to** $epochs$ **do**
 2 **For** $batch$ in $get_batches(D_{source})$ **do**
 3 $a_{probs} \leftarrow \pi_\theta(batch)$
 4 Compute policy network loss based on Equation (12).
 5 Update θ using the gradient descent method.
 6 **End**
 7 **End**

Step 2: Online fine-tuning in the target region

8 **Initialize** Actor-Critic network: Transfer pre-trained encoder weights to θ_{enc} and policy-head weights to θ_π ; randomly initialize the value-head θ_{vf} .
 9 $env \leftarrow SetupEnv(E, \theta_{old} \leftarrow \{\theta_{enc}, \theta_\pi\})$
 10 **For** each iteration $i = 1, 2, \dots$ **do**
 11 Clear buffer $\mathcal{D}$.
 12 **For** decision epoch $k = 1$ **to** $|K|$ **do**:
 13 Obtain state s_k and available mask M_k .
 14 Sample action a_k from masked policy $\pi_{\theta_{old}}(a_k | s_k, mask(\bullet))$.
 15 The DRT scheduling system performs the assigned action in environment env and collects the reward r_k based on Equation (10) and next state s_{k+1} .
 16 A tuple (s_k, a_k, r_k, s_{k+1}) is recorded.
 17 **End**
 18 Compute TD residuals δ_k and advantage estimates $\hat{A}_k$ for all decision epochs in the rollout buffer $\mathcal{D}$ according to Equation (16), and keep them fixed during PPO updates.
 19 **For** PPO update step $h = 1$ **to** H **do**:
 20 Sample a mini-batch of data from $\mathcal{D}$.
 21 Update θ_{vf} by gradient descent on the TD-error loss in Equation (15), and update θ_π by gradient ascent on the PPO clipped objective in Equation (17), while keeping θ_{enc} frozen.
 22 **End**
 23 Update old policy parameters $\theta_{old} \leftarrow \{\theta_{enc}, \theta_\pi\}$.
 24 **End**

5. Experiments

This section evaluates the proposed TR-PPO framework using real-world DRT datasets under multiple regional scheduling scenarios. We first describe the datasets and simulation settings, and then compare TR-PPO with heuristic and RL baselines in terms of service quality, operational cost, transferability, and data efficiency.

5.1. Data and simulation

We utilize SUMO 1.22.0 (Lopez et al., 2018) to construct a controlled environment for evaluating DRT dispatch policies. The initial road networks of the study areas are generated from OpenStreetMap (OSM) and subsequently inspected and manually corrected using

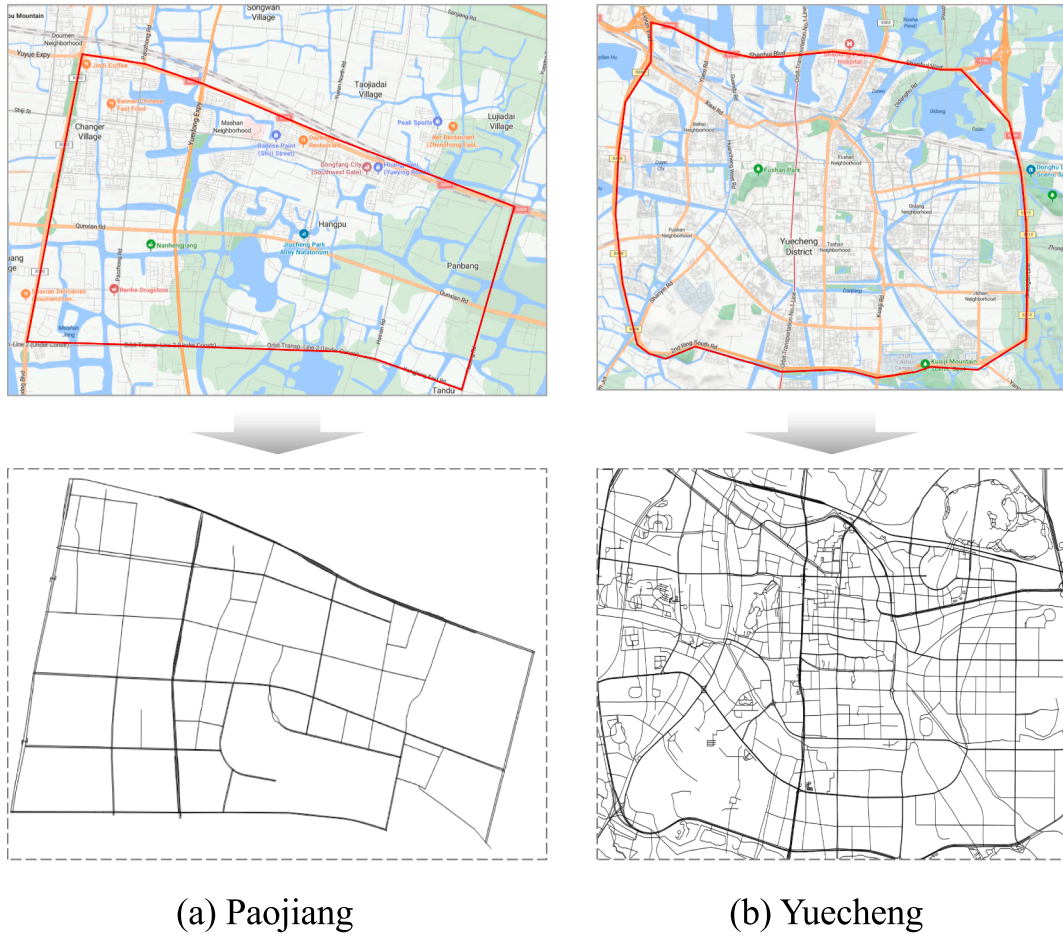


Fig. 5. Study areas and simulation networks: (a) Paojiang for offline pre-training and (b) Yuecheng for online fine-tuning and evaluation.

SUMO NetEdit. The corrections included disconnected links, road connectivity, lane attributes, turning connections, DRT stop locations, and feasible service routes.

The simulation explicitly represents DRT vehicles, passenger requests, background traffic, and traffic signals. DRT passenger requests are reconstructed from real-world operational records according to their observed request times and origin-destination stops. Background vehicle trips are generated from historical ride-hailing OD records obtained from the Amap (Gaode) platform. These records are used as a proxy for general background traffic demand. Road-level speed and congestion information from Amap is additionally used to parameterize background traffic conditions and link-speed and travel-time settings.

Traffic-signal locations are derived from the road-network data, and the signals operate according to fixed-cycle programs. Because field-observed signal timing plans were unavailable, the signal timings are not independently calibrated against empirical signal-control observations. The default SUMO car-following and vehicle-dynamics parameters are adopted, and no calibration against vehicle-trajectory-level observations was performed. Accordingly, the resulting simulation is intended to provide a consistent microscopic environment for comparing DRT dispatch policies under common traffic conditions. [Appendix B](#) provides a detailed classification of the simulation parameters and data sources.

The data used in this study are derived from real-world DRT operations in two districts of Shaoxing City, China: Paojiang District and Yuecheng District. The datasets include request submission times, passenger pick-up and drop-off timestamps, the latitude and longitude of pick-up and drop-off locations, stop IDs, and stop names. In the proposed transfer setting, the Paojiang dataset is used for offline pre-training, while the Yuecheng dataset is used for online fine-tuning and evaluation.

As shown in [Fig. 5](#), the Paojiang dataset covers an operating area of approximately $3.0 \text{ km} \times 6.4 \text{ km}$, whereas the Yuecheng dataset covers a larger area of approximately $7.5 \text{ km} \times 13.2 \text{ km}$. In the offline pre-training stage, we use two data periods from Paojiang, covering February 9–16, 2024, and March 11–15, 2024. The online dataset contains one working week of data from Yuecheng, covering the period from March 11 to March 15, 2024. The two datasets are provided by Hangzhou Public Transport Group. [Table 2](#) presents the details of the two datasets. Each DRT bus has a maximum passenger capacity of 9, and the total fleet size is 15 vehicles. Each passenger's maximum response waiting time is set to 20 minutes.

At the beginning of each simulated day, all 15 DRT buses depart from the same depot and operate continuously from 07:00 to

Table 2
Dataset descriptions.

Dataset	Paojiang	Yuecheng
Period	2024.02.09–2024.02.16 and 2024.03.11–2024.03.15	2024.03.11–2024.03.15
Operating hours	7:00–21:30	7:00–21:30
Number of average daily requests	271	455
Number of pick-up and drop-off points	164	292

21:30. Requests enter the system according to their actual submission times. The system is updated at 1-second intervals; expired requests are removed, deferred requests whose deferral time reaches 1 min are reactivated, newly arriving requests are added to the pending pool, and at most one active request is evaluated in each interval. The passenger boarding and alighting processes are governed by the native default parameters of SUMO's taxi device platform without custom overrides, which characteristically applies a 0-second pickup duration and a 60-second drop-off duration for each passenger batch across all evaluated scheduling methods. These values follow the default service-event parameters of the simulation platform and were not calibrated as empirical boarding or alighting times. Therefore, they should be interpreted as simulation assumptions rather than direct representations of real-world passenger boarding and alighting processes. The current simulation does not explicitly model detailed stop-level service behaviors, such as passenger walking, queuing, door operation, boarding, and alighting. Since the same pickup and drop-off duration settings are applied to all compared algorithms, the relative performance comparisons remain internally consistent.

5.2. Experimental settings

The TR-PPO framework adopts an encoder-actor-critic architecture. The encoder in TR-PPO employs a four-layer MLP. The policy head is a two-layer MLP that takes the encoded feature vector as input and outputs action logits. The value function is parameterized by a two-layer MLP, which processes an encoded vector and predicts a scalar evaluation value. The hidden layer size of each component is set to 256.

The encoder input is a 196-dimensional state vector, constructed by concatenating bus-level features and a system-time feature. Each of the 15 vehicles is represented by a 13-dimensional vector, including a 9-dimensional remaining in-vehicle tolerance vector, one remaining-capacity feature, and three relative spatial features. The remaining in-vehicle tolerance vector is padded according to the vehicle capacity of 9. Therefore, the final input dimension is $15 \times 13 + 1 = 196$.

The output of the policy is a 1-by-16 tensor, where the first entry corresponds to the deferral action and the remaining 15 entries correspond to the 15 DRT buses. The learning rate in the offline stage is 0.0008, and in the online stage is 2×10^{-6} . The clip parameter of PPO in the online stage is 0.3. Discount factor γ and GAE parameter λ are set as 0.99 and 0.95 (Schulman et al., 2017), respectively. We use Adam as the optimizer. The system uses an Intel Core i7-13700K CPU for simulation computations and an NVIDIA GeForce RTX 4070 GPU to accelerate parts of the training process. The training set and test set comprise 80% and 20% of the Yuecheng dataset, respectively.

To assess the model performance under congested demand conditions, the peak periods are defined as 07:00–09:00 and 17:00–19:00. For a small-sized demand-responsive bus with a maximum driving speed of 40 km/h, the operation cost α is US\$0.44 /km (Wu et al., 2022). The cost for passenger demand cancellations for a passenger γ_d is US\$ 7.05 (Xu et al., 2022). The waiting time cost is calculated based on passengers' average salary, with the cost β_2 brought by passengers' waiting time being US\$ 0.11 /min (Pei et al., 2019). The in-vehicle time cost β_1 is US\$ 0.09 /min (Wu et al., 2022).

5.3. Comparison of DRT system performance under different scheduling methods

The compared heuristic methods include the following: a large neighborhood search (LNS, Voigt et al., 2022) and the reactive anytime optimal method (RAO, Alonso-Mora et al., 2017). The algorithms described above employ the same objective as the approach proposed in this paper, namely, to minimize the total system cost involving DRT buses and passengers. In addition, PPO and Deep Q-Network (DQN) are included as representative RL baselines for single-agent scheduling problems, given their demonstrated effectiveness in vehicle routing applications (Ara et al., 2023). Graph Reinforcement Learning (Graph-RL) is also considered as a network-oriented RL baseline. It combines graph representation learning with a bi-level optimization framework, in which an RL policy specifies the desired next network state and a convex optimization problem determines the corresponding feasible control actions (Gammelli et al., 2023). A warm-start RL (WSRL) is further introduced to represent an offline-to-online transfer setting (Zhou et al., 2025). To examine the effect of online fine-tuning, we also include TR-PPO without fine-tuning (TR-PPO-noFT) for comparison. For fairness, all learning-based methods use the same routing heuristic for insertion evaluation and share the same state space, action space, and reward formulations.

To ensure a fair comparison, LNS and RAO are implemented with the same fleet, requests, operational constraints, routing module, and system-cost objective as TR-PPO. LNS is configured as a batch destroy-and-repair heuristic with a fixed removal ratio, a maximum iteration limit, and a no-improvement stopping rule. Candidate insertions are accepted only if they satisfy the pickup time window, maximum waiting time, delivery time window, vehicle capacity, and maximum in-vehicle time constraints. At each decision epoch, newly revealed requests within the current 30-second decision window are optimized as a batch. In our experiments, the destroy rate is set to 0.3, the maximum number of iterations is set to 100, and the search terminates early if no improvement is found for 10

Table 3
System performance comparison across different algorithms.

Algorithms	Total driving mileage (km)	On-time request completion rate		Service rate	Average pickup waiting time (minutes)	Average in-vehicle time (minutes)	Total system cost (\$)
		Peak Period	Off-peak Period				
LNS	2260.88	85.05%	88.38%	87.14%	6.13	8.53	2166.99
RAO	2067.21	85.05%	87.77%	86.76%	2.46	6.28	1792.35
PPO	1781.56	87.03%	91.44%	97.31%	8.50	8.82	1772.27
DQN	1546.68	87.03%	89.60%	96.16%	9.85	8.80	1782.87
WSRL	2316.65	80.99%	89.60%	97.70%	9.52	9.32	2076.36
Graph-RL	2209.90	82.47%	88.38%	96.16%	7.28	8.61	1918.84
TR-PPO-noFT	1978.69	85.52%	88.07%	95.97%	8.74	8.55	1904.29
TR-PPO	1477.89	94.00%	92.97%	97.70%	8.07	8.07	1566.94

consecutive iterations. RAO is implemented as a rolling-horizon batch optimizer rather than being triggered by each request arrival. At each decision interval, newly revealed and still unserved requests are optimized as a batch.

Table 3 compares the performance of the DRT system across different algorithms in the Yuecheng District. To illustrate the effectiveness of TR-PPO, we utilize the following evaluation metrics: (i) total driving mileage traveled by DRT buses during the operation period; (ii) on-time request completion rate during peak and off-peak periods, where a request is regarded as completed on time if the passenger's in-vehicle travel time does not exceed 1.5 times the shortest travel time between the request origin and destination; (iii) service rate, defined as the proportion of fulfilled requests among all submitted requests. (iv) average pickup waiting time. (v) average in-vehicle time; and (vi) total system cost. The total system cost consists of operational cost and passenger time cost. The service rate and the on-time completion rate measure different aspects of performance. The service rate counts requests that are accepted and eventually delivered, whereas the on-time completion rate is computed ex-post from the realized SUMO trajectory. The online feasibility check uses estimated network travel times to enforce Equation (6), but the realized in-vehicle time after simulation may differ from this estimate due to route execution effects, dwell times, and traffic signals. Therefore, a served request may still be excluded from the on-time count if its realized in-vehicle time exceeds 1.5 times the reference shortest-path travel time. The operational cost includes vehicle travel cost and penalties for removed requests, while the passenger time cost includes pickup waiting time cost and in-vehicle travel time cost.

To further analyze the operational mechanism behind the service-quality trade-off, we introduce three fleet-utilization metrics. The average occupancy is defined as the total passenger-vehicle time divided by the total vehicle operation time:

$$\text{Average occupancy} = \frac{\sum_t \sum_{b \in B} n_{b,t}}{\sum_t \sum_{b \in B} 1} \quad (18)$$

where $n_{b,t}$ denotes the number of onboard passengers in vehicle b at simulation step t . The occupied vehicle ratio is defined as the proportion of vehicle-time during which a vehicle carries at least one passenger:

$$\text{Occupied vehicle ratio} = \frac{\sum_t \sum_{b \in B} y_{b,t}}{\sum_t \sum_{b \in B} 1}, y_{b,t} = \begin{cases} 1, n_{b,t} > 0 \\ 0, n_{b,t} = 0 \end{cases} \quad (19)$$

$$\text{Average occupancy when occupied} = \frac{\text{Average occupancy}}{\text{Occupied vehicle ratio}} \quad (20)$$

A higher average occupancy indicates that vehicles carry more passengers on average, while a higher occupied vehicle ratio indicates that a larger proportion of the fleet is actively serving passengers rather than running empty or remaining idle. The average occupancy when occupied further captures the average onboard load conditional on a vehicle being occupied, thereby reflecting the extent of passenger consolidation among active service vehicles.

As shown in Table 3, TR-PPO achieves the lowest total system cost and driving mileage among all compared methods. It also obtains the highest peak-period and off-peak on-time request completion rates, reaching 94.00% and 92.97%, respectively. In terms of service rate, TR-PPO reaches 97.70%, which is tied with WSRL for the highest value among the compared methods. Although RAO achieves the shortest average pickup waiting time and average in-vehicle time, its service rate and total system cost are less competitive than those of TR-PPO. Compared with the RL baselines, TR-PPO reduces the total system cost by 11.59% to 24.53%, and compared with LNS, the reduction reaches 27.69%. However, the results also show that not all RL-based methods outperform RAO on every service-quality metric. For example, WSRL and Graph-RL have lower peak-period on-time completion rates than RAO. Overall, TR-PPO achieves the most favorable balance among operational cost, vehicle mileage, service coverage, and on-time completion, whereas the other RL baselines show metric-dependent performance variations.

The comparison between TR-PPO-noFT and TR-PPO helps isolate the additional contribution of the online reinforcement learning phase beyond imitation learning. TR-PPO-noFT represents the policy obtained after behavior cloning from expert trajectories, without subsequent PPO fine-tuning in the target region. As shown in Table 3, this policy provides a reasonable transferred initialization, but its

Table 4

Comparison of fleet-utilization indicators under different algorithms.

Algorithms	Average occupancy	Occupied vehicle ratio	Average occupancy when occupied
LNS	0.31	0.31	1.00
RAO	0.27	0.25	1.11
PPO	0.36	0.33	1.09
DQN	0.36	0.32	1.13
WSRL	0.34	0.30	1.13
Graph-RL	0.35	0.35	1.00
TR-PPO-noFT	0.32	0.29	1.10
TR-PPO	0.36	0.30	1.20

Table 5

Performance comparison of average runtime (s) per decision epoch under two problem sizes.

Algorithms	Case-500	Case-1000
LNS	0.07	0.89
RAO	0.03	0.78
PPO	0.02	0.03
DQN	0.04	0.04
WSRL	0.01	0.01
Graph-RL	0.01	0.01
TR-PPO	0.01	0.01

Table 6

Comparison of TR-PPO online fine-tuning across different module setups.

Method	Episode Return	Episode to Reach -980.00	Total Training Time to Reach -980.00 (hours)
TR-PPO-noFreeze	-964	1981	35.28
TR-PPO	-968	835	18.59

direct cross-region performance is limited under the target-region demand and network conditions. After PPO-based online fine-tuning, TR-PPO further reduces the total system cost by 17.7% and increases the peak-period on-time completion rate by 8.48 percentage points. These results indicate that imitation learning mainly provides an effective behavioral prior, while the online RL phase further adapts the policy to the target operating environment and improves dispatching performance.

This operational trade-off is further examined in Table 4 using vehicle-utilization indicators. As shown in Table 4, TR-PPO has an average occupancy comparable to PPO and DQN and a slightly lower occupied vehicle ratio, but it achieves the highest average occupancy when occupied. This suggests that TR-PPO does not improve performance by simply keeping more vehicles active. Instead, it uses vehicle capacity more selectively and consolidates compatible requests more effectively. Together with its lower total mileage and system cost, these results indicate that TR-PPO improves fleet efficiency through better vehicle-request matching rather than merely increasing fleet activity.

The computational efficiency of different algorithms is reported separately in Table 5. As shown in Table 5, the runtime of optimization-based methods such as LNS and RAO increases substantially under the larger Case-1000 setting, whereas TR-PPO maintains a near-real-time response. This suggests that the learned policy is more suitable for large-scale real-time deployment after training.

To further illustrate the operational logic of TR-PPO, Appendix C provides example scheduling trajectories extracted from the Yuecheng simulation environment during the morning peak period.

5.4. Ablation study on transferable RL framework

This subsection conducts an ablation study to analyze the impact of key modules within the TR-PPO framework, specifically focusing on the frozen module. The frozen module is designed to lock the parameters of pre-trained sub-networks to preserve pre-trained representations during the early stages of online fine-tuning. We compare two configurations: (i) TR-PPO without the frozen module (TR-PPO-noFreeze), where all parameters remain trainable; and (ii) the full TR-PPO model. The comparative results are summarized in Table 6.

The results in Table 6 show that the frozen module drastically accelerates the convergence process. While both setups eventually reach a comparable level of episode return, TR-PPO achieves the target episode return of -980.00 in only 835 episodes, whereas TR-PPO-noFreeze requires 1,981 episodes. This represents a reduction of approximately 57.85% in the required training iterations. Furthermore, the total training time is reduced from 35.28 hours to 18.59 hours. These results demonstrate that freezing pre-trained

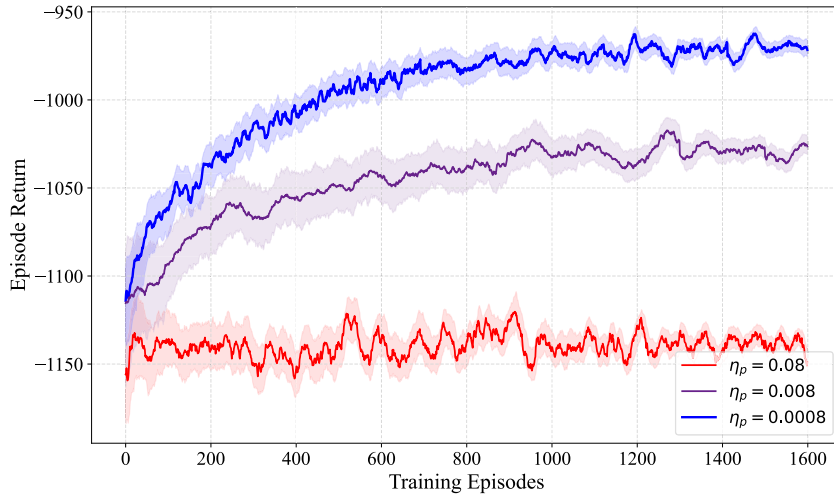


Fig. 6. TR-PPO under different learning η_p rates in offline pre-training (5 random seeds).

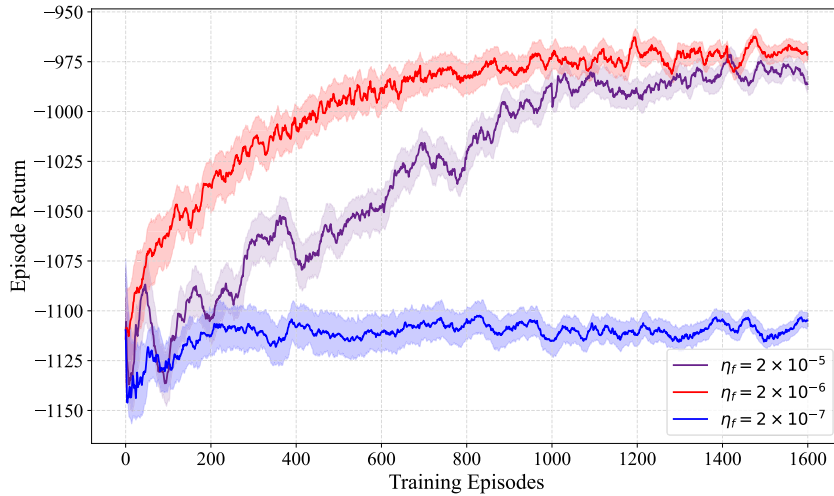


Fig. 7. TR-PPO under different learning rates η_f in online fine-tuning (5 random seeds).

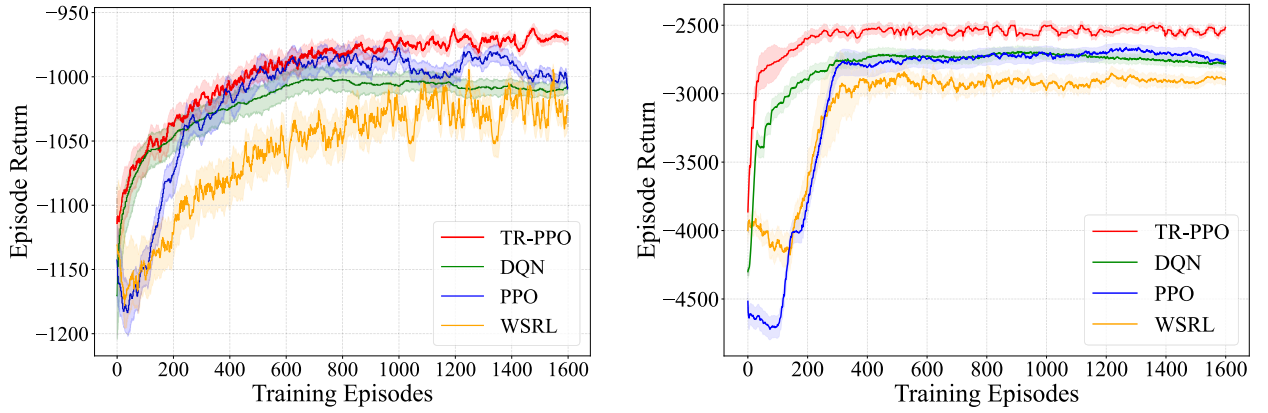
representations effectively prevents “catastrophic forgetting” and avoids redundant explorations in the early training phase. This ensures that the agent can adapt to the spatio-temporal dynamics of the new city with substantially higher computational efficiency while maintaining high solution quality.

5.5. Sensitivity analysis of learning rates

We conduct sensitivity analyses to assess the impact of offline pre-training learning rate η_p and online fine-tuning learning rate η_f , aiming to identify parameter ranges that support stable policy transfer. To ensure the statistical reliability of the results and minimize the impact of stochastic initialization, each learning-rate configuration is trained across five independent runs. The reported performance metrics for each instance represent the ensemble average of these five experiments.

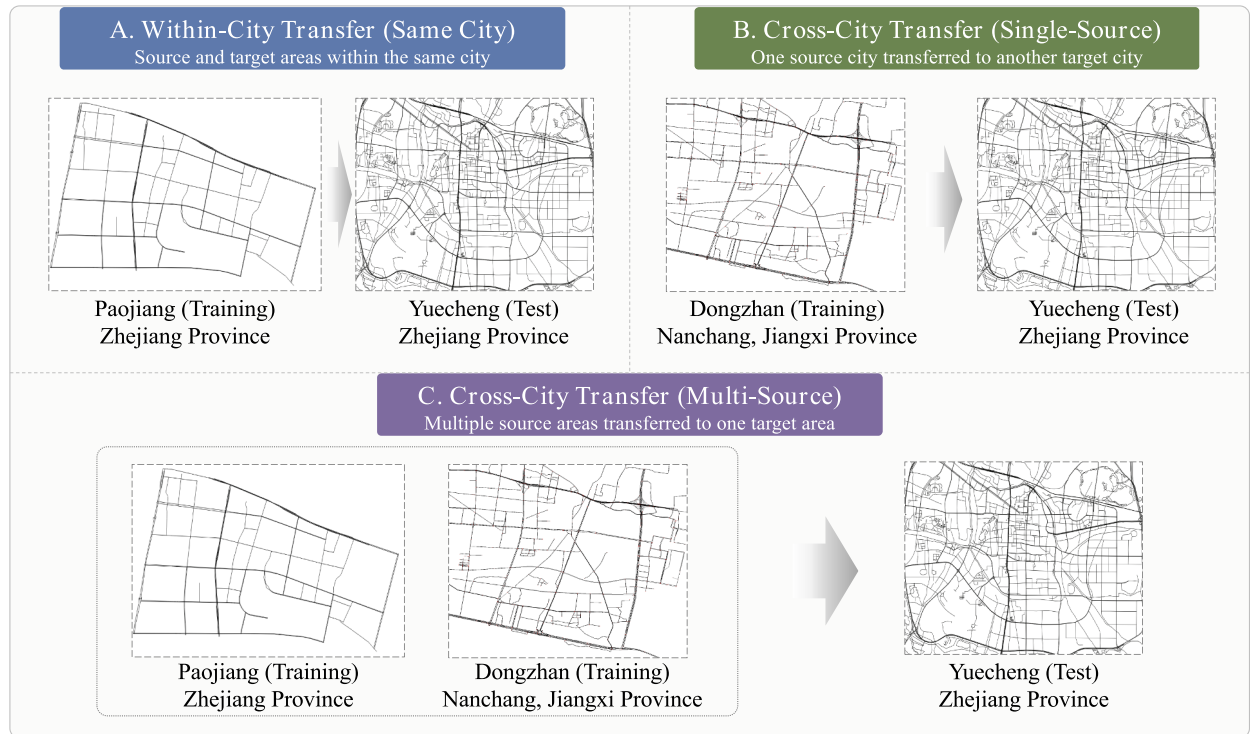
First, the impact of η_p is evaluated as shown in Fig. 6. The results indicate that the learning behavior is sensitive to the offline pre-training learning rate. Larger values, such as $\eta_p = 0.08$ and $\eta_p = 0.008$, produce more pronounced oscillations, suggesting that overly large update steps can destabilize behavior-cloning pre-training. Smaller values, such as $\eta_p = 0.0008$, reduce the update magnitude and achieve better final average performance among the tested settings, but the curve still shows periods of flatness and slower improvement. These results indicate a clear trade-off between training stability and convergence speed under different learning-rate settings.

Furthermore, we examine the sensitivity of the online fine-tuning learning rate η_f , with η_p fixed at 0.0008. As illustrated in Fig. 7, the results indicate that a higher learning rate ($\eta_f = 2 \times 10^{-5}$) leads to sub-optimal convergence with persistent variance even in the



(a) Yuecheng (500 daily requests on average)

(b) Xiongan (750–900 daily requests)

Fig. 8. Comparison of TR-PPO with other RL baselines (5 random seeds).**Fig. 9.** Illustration of the within-city, single-source cross-city, and multi-source cross-city transfer settings used for DRT dispatch policy evaluation.

late stages of training. Conversely, a very low rate ($\eta_f = 2 \times 10^{-7}$) results in insufficient policy adaptation. The best balance is found at $\eta_f = 2 \times 10^{-6}$. This suggests that a moderate fine-tuning learning rate is preferable because it supports target-region adaptation while preserving the pre-trained representations.

Overall, the sensitivity results show that learning rates affect convergence speed and final performance, but non-optimal learning-rate choices do not make the approach unusable. Across both offline pre-training and online fine-tuning, the tested learning rates cover a 100-fold range; most tested settings remain trainable, but some show oscillation or flatness. This suggests that TR-PPO remains usable under non-optimal learning-rate settings.

Table 7
System performance comparison across different transfer settings.

Transfer settings	Total driving mileage (km)	On-time request completion rate		Service rate	Average pickup waiting time (minutes)	Average in-vehicle time (minutes)	Total system cost (\$)
		Peak Period	Off-peak Period				
A	1477.89	94.00%	92.97%	97.70%	8.07	8.07	1566.94
B	1451.29	83.44%	93.58%	97.31%	8.86	8.64	1639.37
C	1513.68	83.58%	91.13%	98.08%	8.73	8.55	1630.07

5.6. Cross-region adaptability and source-domain transfer robustness

5.6.1. Target-domain adaptation under heterogeneous demand densities

To further evaluate the cross-region transferability of TR-PPO under different urban demand settings, we conduct comparative experiments in Yuecheng District and Xiongan New Area (A state-level new area in the Baoding area of Hebei, China). Fig. 8 reports the online training trajectories of TR-PPO and the three RL baselines, namely PPO, DQN, and WSRL. For the online training stage, we extract five consecutive days of travel data from Xiongan New Area in Hebei Province, spanning from November 10 to November 14, 2025. The dataset comprises more than 750 requests per day on average, with a maximum of approximately 900 daily requests, distributed across 261 distinct pick-up and drop-off points. The operational area covers approximately $10.02 \times 3.52 \text{ km}^2$. To ensure a controlled comparison, the vehicle service hours and other fundamental configurations are kept consistent with those used in Yuecheng. Several key observations can be drawn from the cross-regional comparison.

First, TR-PPO exhibits a clear advantage in training stability across both target regions. In Yuecheng (Fig. 8a), where the demand level is relatively lower, PPO and WSRL show larger fluctuations and slower improvement during the early training stage, whereas TR-PPO starts from a higher episode return level and converges more smoothly.

Second, the relative advantage of TR-PPO remains evident in the higher-demand Xiongan scenario (Fig. 8b). Although the denser demand provides more interaction samples for all methods, TR-PPO still reaches a stable performance level faster than PPO and DQN. In particular, TR-PPO converges within about 200 episodes, while PPO and DQN require approximately 300–400 episodes to stabilize.

Third, the WSRL (Warm-start RL) baseline shows a characteristic “negative transfer” dip in both regions, but the recovery is faster in Xiongan due to the higher volume of corrective online feedback, further validating that TR-PPO’s strategy refinement is essential for smoother transitions.

Overall, the results of TR-PPO in both low-density (Yuecheng) and high-density (Xiongan) regions indicate its cross-region transferability. It demonstrates that our framework is not only effective at mitigating data scarcity but also excels at achieving higher performance ceilings in data-rich urban environments.

5.6.2. Robustness to source-domain variations

To examine the robustness of the proposed transfer learning framework to different source-domain configurations, we further compare three transfer settings, as illustrated in Fig. 9. Setting A represents within-city transfer, where the policy is trained and tested on different areas within the same city. Setting B represents single-source cross-city transfer, where the policy is pretrained on one city and transferred to another city. Setting C represents multi-source cross-city transfer, where the policy is pretrained using source domains from multiple cities and then transferred to the target city.

The Dongzhan dataset contains 20 days of DRT request records, with 15,888 requests in total. On average, the dataset includes approximately 794 requests per day, with a maximum of 829 daily requests. These requests are distributed across 100 distinct pick-up and drop-off stops. The same target-domain setting as in the Yuecheng experiment is adopted for online fine-tuning and evaluation.

In the multi-source transfer setting, the source-domain data from Paojiang and Dongzhan are jointly used during the offline pre-training stage. Specifically, each source domain is first converted into the same state-action-reward representation, including vehicle states, request attributes, relative spatial features, and dispatching decisions. During pre-training, training episodes are sampled from both source domains, allowing the policy network to learn dispatching patterns from heterogeneous road networks and demand distributions. The model parameters learned from the combined source-domain experience are then used to initialize the target-domain policy. In the online fine-tuning stage, the pretrained policy is further updated using requests from the target domain, while the source-domain data are no longer used for policy evaluation. This design enables the model to acquire more generalizable dispatching representations before being adapted to the target operating environment. The system performance is summarized in Table 7.

As shown in Table 7, all three transfer settings achieve competitive performance relative to the benchmark algorithms reported in Table 3, demonstrating that the learned dispatching representations can be reused across spatial domains. Nevertheless, their performance varies considerably with the choice of source domain.

Setting A achieves the lowest total system cost, suggesting that within-city transfer provides a highly effective initialization when the source and target networks share similar spatial and demand characteristics.

The single-source cross-city setting, Setting B, obtains the lowest total driving mileage among all tested settings, with 1451.29 km. However, its peak-period on-time request completion rate decreases to 83.44%, indicating that a source domain from another city may help reduce vehicle mileage but may not always transfer well in terms of time-sensitive service quality. This may be caused by

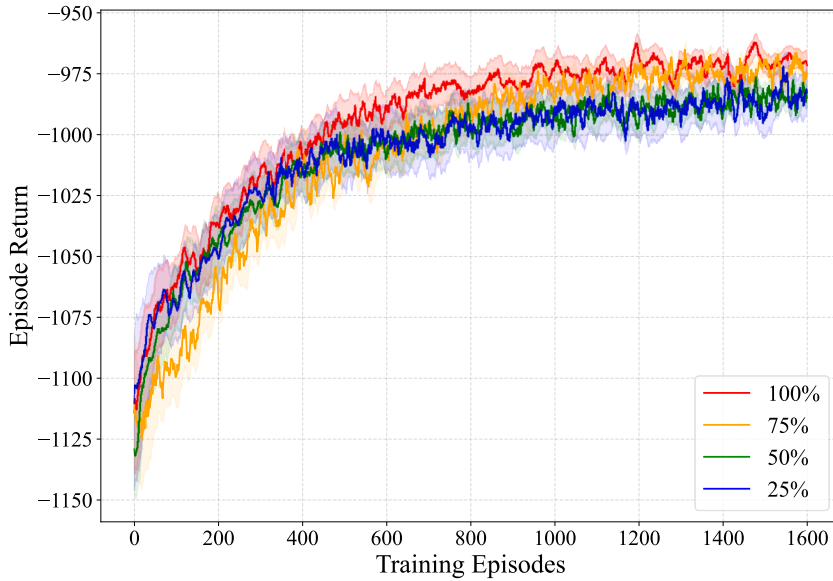


Fig. 10. Performance comparison of TR-PPO under different data volumes (5 random seeds).

differences in network topology, traffic conditions, and demand distribution between the source and target cities.

The multi-source cross-city setting, Setting C, provides a more balanced performance. Compared with Setting B, it slightly increases the total driving mileage but reduces the total system cost from 1639.37 to 1630.07 and improves the service rate to 98.08%, the highest among all settings. This suggests that exposure to heterogeneous source domains during offline pre-training may improve the generalizability of the learned policy, although its performance still depends on the similarity between the source and target domains.

Overall, the results show that source-domain selection affects both operational efficiency and service reliability. Within-city transfer achieves the best overall cost performance, whereas multi-source transfer offers a more balanced alternative when a closely matched source region is unavailable. These findings are consistent with the intended role of relative spatial representations and the coordinate-independent feature encoder, which are designed to reduce dependence on absolute geographic coordinates.

5.7. Data efficiency and robustness under limited online training data

To evaluate the data efficiency and robustness of the TR-PPO framework, we conduct a sensitivity analysis by varying the amount of available online training data. Specifically, the training requests from the Yuecheng dataset are randomly subsampled to 25%, 50%, 75%, and 100% of the original data volume.

As presented in Fig. 10, even with only 25% of the training data, TR-PPO maintains a stable upward trajectory and achieves a performance level close to the full-data TR-PPO configuration. While the final episode return under 25% data is slightly lower than that of the 100% configuration, the model does not exhibit severe performance collapse under the limited-data setting. This demonstrates that the latent scheduling knowledge captured during the offline pre-training phase provides an effective policy initialization, thereby reducing the need for extensive online exploration.

As the data volume increases from 50% to 100%, the performance gap gradually narrows. The 100% data configuration yields the highest return and the smoothest convergence, benefiting from more sufficient samples for policy updates. However, the 75% data setting performs almost comparably to the full-data setting in the later training episodes. This suggests that TR-PPO reaches a near-saturation point before using the complete dataset, demonstrating strong sample efficiency. The initial “jumpstart” advantage remains robust even at the 25% level, indicating that the transfer of pre-trained weights effectively bypasses the high-variance exploration phase that usually demands large amounts of interaction data.

5.8. Impact of DRT fleet size

To assess the impact of DRT fleet size on system service performance, we conduct a comparative performance evaluation of the proposed model in Yuecheng District. The system performance is evaluated using three metrics: average passenger pickup waiting time, system service rate, and total bus driving distance. The detailed results are presented in Fig. 11, and the model configuration parameters are identical to those described in Section 5.2. In Fig. 11, as the fleet size increases, the average passenger pickup waiting time generally decreases and the service rate increases, whereas total driving mileage varies non-monotonically across the tested fleet size. This result can be interpreted as follows. First, the average passenger pickup waiting time decreases because a larger DRT bus fleet reduces detours and increases the available capacity for pick-ups. The decrease in total DRT bus travel distance is mainly due to the larger feasible routing space provided by a larger DRT bus fleet when serving a fixed number of requests, which allows routes to be

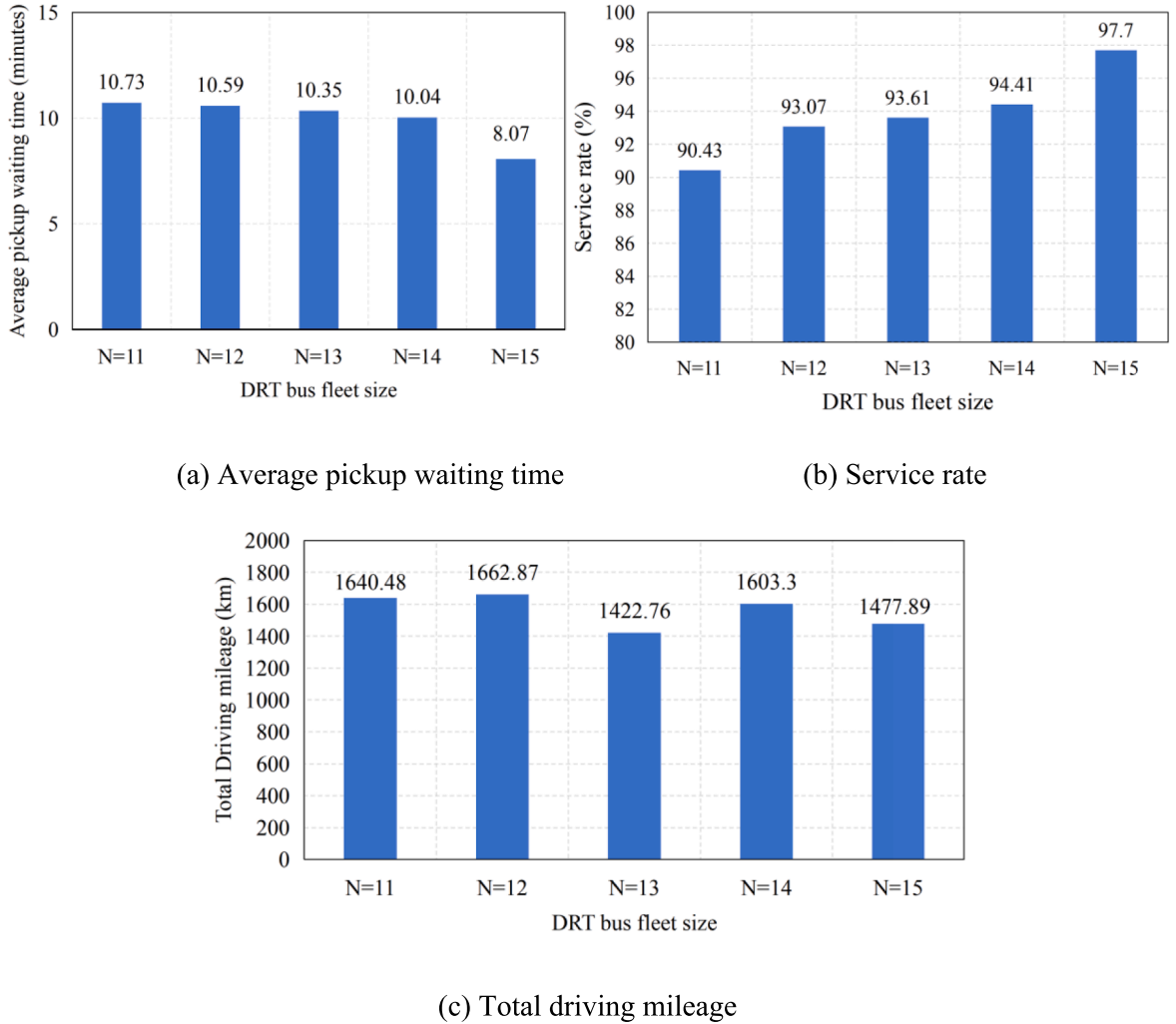


Fig. 11. TR-PPO performance under different DRT bus fleet sizes.

coordinated more efficiently. Similarly, the service rate improves as the fleet size grows. While increasing the fleet size can improve system performance and thereby enhance the service attractiveness, it also entails higher upfront investment costs for the DRT service operator. Accordingly, determining an appropriate balance between service quality and initial fleet investment is an important research topic. This calls for quantitative decision-support tools that can explicitly capture this trade-off. The proposed model can be used not only for real-time operation of DRT buses, but also can provide preliminary operational evidence for fleet-size planning to maximize benefits for DRT service operators.

To examine the consistency between the learned policy and the fleet-size mask, we analyze the raw network outputs before masking. As shown in Fig. 12, the pre-mask probability mass assigned to unused vehicle slots remains close to zero across all tested fleet sizes. The vertical axis represents the total pre-mask probability mass assigned to unused vehicle slots. A lower value indicates that the policy assigns less preference to vehicles that are unavailable under the current fleet-size setting. Because these values are mostly close to zero, a symmetric logarithmic scale is used for visualization. Although a few outliers appear for fleet sizes of 13 and 14, they do not become the policy's preferred action. These findings indicate that the fleet-size mask mainly guarantees action validity under variable fleet sizes, rather than overriding the learned vehicle-selection policy.

6. Conclusion

This paper adapts and integrates established transfer reinforcement learning components into a DRT-specific scheduling framework, TR-PPO, for demand-responsive transit scheduling in newly deployed or data-scarce urban regions. The framework combines offline pre-training in a source region with online fine-tuning in a target region, with the aim of improving cross-region adaptability in dynamic DRT operations.

The experimental results demonstrate that in the main Yuecheng evaluation, TR-PPO outperforms the compared heuristic and RL

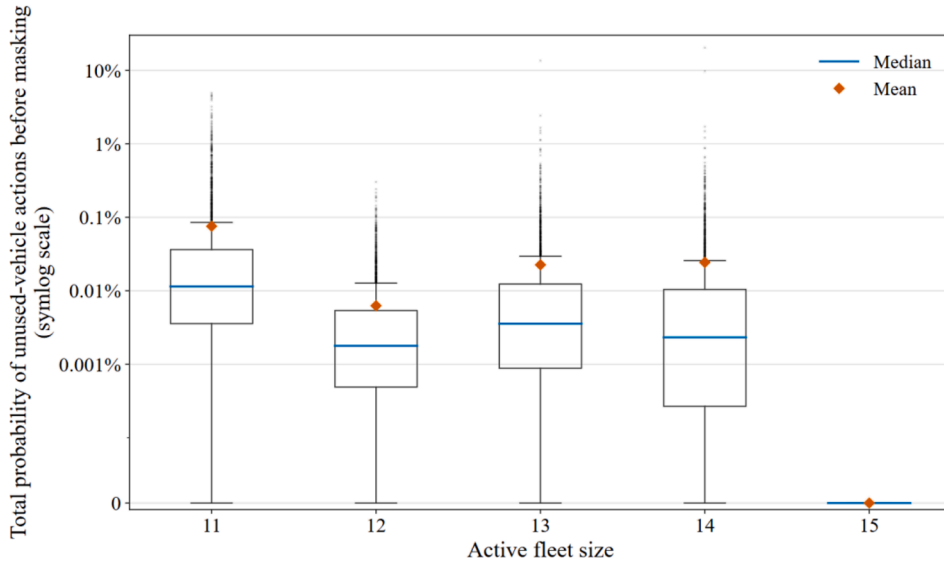


Fig. 12. Mask consistency under different active fleet sizes.

baselines in the target-region scheduling tasks. In the Yuecheng case, TR-PPO reduces the total system cost by up to 27.69% compared with the LNS heuristic and by 11.59%–24.53% compared with the RL baselines. In addition, the selective freezing strategy substantially improves fine-tuning efficiency, reducing the number of episodes required to reach the target return by 57.85%. The cross-region experiments in Yuecheng and Xiongan further show that the proposed framework achieves more stable and faster adaptation under both lower- and higher-demand settings.

The results also indicate that the combination of offline pre-training and online fine-tuning improves data efficiency in the target region. Even when only a limited amount of target-region data is available, the pre-trained model provides an effective initialization that reduces the need for extensive random exploration and accelerates policy adaptation. These findings suggest that TR-PPO is a practical approach for improving the deployability of DRT scheduling policies across the tested regions with different demand patterns and network structures.

This study has several limitations. The current framework uses a fixed-size homogeneous fleet of 15 buses because its MLP input and action dimensions are predefined, although padding and action masking accommodate temporarily unavailable vehicles. Future work will investigate fleet-size-agnostic architectures using shared encoders, attention, or graph neural networks to support variable fleet sizes. Second, the current model focuses on single-mode DRT operations. Future extensions could incorporate heterogeneous fleets and multimodal traffic coordination to improve applicability in more diverse urban mobility settings. Third, passenger service events are simplified in the SUMO-based simulation, where the pickup and drop-off durations follow SUMO's native taxi-device default settings rather than calibrated empirical boarding or alighting times. Future work will incorporate calibrated non-zero service-event durations and conduct sensitivity analysis to evaluate their effects on DRT service quality.

Finally, although TR-PPO should not be interpreted as a transit foundation model, it can be viewed as an early-stage step toward more general transferable learning frameworks for transit operations. Moving toward a genuine transit foundation model would require progress along several dimensions. First, the data scale would need to expand substantially beyond the limited set of DRT regions considered in this study, covering multiple cities, road-network structures, fleet sizes, demand regimes, and transit modes. Second, the model architecture would need to move beyond a fixed-size MLP policy toward higher-capacity and more flexible representations, such as graph neural networks, attention-based encoders, or transformer-style architectures that can process variable-size vehicle fleets, request sets, and road networks. Third, the task scope would need to include a broader set of transit operations, such as demand forecasting, DRT dispatching, vehicle rebalancing, multimodal coordination, depot or charging scheduling, and disruption response. Finally, standardized generalization benchmarks would be required to evaluate whether a model can transfer across cities, operating regions, fleet sizes, demand intensities, and network structures. In this sense, the present study should be interpreted as a DRT-specific transfer-learning precursor toward this broader vision.

CRedit authorship contribution statement

Jing Bian: Writing – original draft, Methodology, Formal analysis. **Xiaolei Ma:** Writing – review & editing, Funding acquisition. **Haoyang Yan:** Writing – review & editing, Methodology. **Hesham El-Sayed:** Writing – review & editing, Funding acquisition. **Kun Gao:** Writing – review & editing. **Xiaohan Liu:** Writing – review & editing, Validation, Supervision, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. T2588101) and the Beihang-UAEU Joint Collaboration Proposal (G00005366).

Appendix A

Summary of notation and model parameters.

Notation	Definition
Sets (indices)	
$T(t)$	Set of 1-second discrete time intervals
$K(k)$	Set of decision epochs
$R_{acc}(d)$	Set of responded requests
$R_{rem}(d)$	Set of removed requests
$R(d)$	Set of requests ($R = R_{acc} \cup R_{rem}$)
$R_k(d)$	Set of requests present in the system after action a_k is taken at decision epoch k
$B(b)$	Set of DRT buses
$\Phi_{b,k,d}(i,j)$	Feasible position set of insertions of request d for bus b at decision epoch k
States	
S_k	System state at decision epoch k (vector)
t_k	System time at decision epoch k (scalar)
O_d	Origin of request d that is associated with decision epoch k (vector)
D_d	Destination of request d that is associated with decision epoch k (vector)
$l_{b,k}$	Current location of bus b at decision epoch k (vector)
$\nu_{b,k}$	Remaining seat capacity of bus b at decision epoch k (scalar)
$Y_{b,k}$	Planned visiting sequence of bus b at decision epoch k (vector)
$G_{b,k}$	Remaining in-vehicle tolerance time of onboard passengers of bus b at decision epoch k (vector)
Ψ	Relative spatial feature (vector)
Δdis_{bd}	Shortest network distance between bus b and the origin of the current request d (scalar)
$\Delta \theta_b$	Relative bearing from the bus b to its next scheduled waypoint (scalar)
$\Delta \theta_d$	Direction of the origin-destination pair of request d (scalar)
Actions	
A_k	Action space at decision epoch k (vector)
a_k	Selected action at decision epoch k (scalar)
Transition	
$P(S_{k+1}, \tau_k S_k, a_k)$	Joint probability of transitioning from state S_k to state S_{k+1} with sojourn time τ_k , given that action a_k is taken at decision epoch k .
Reward	
r_k	Reward received after action a_k at decision epoch k
C_k	base reward;
$dis_{b,k}$	Cumulative travel distance after action a_k at decision epoch k of bus b
$it_{d,k}$	Cumulative in-vehicle travel time of passenger $d \in R_k$ after action a_k at decision epoch k
$wt_{d,k}$	Cumulative pickup waiting time of passenger $d \in R_k$ after action a_k at decision epoch k
Other variables	
F_b	F_b signifies the total driving distance of bus b
t_d^{occ}	The service request time of request d
t_d^{re}	Request response time of request d
t_d^p	Pick-up time for request $d \in R_{acc}$
t_d^d	Drop-off time for request $d \in R_{acc}$
$n_{b,t}$	Number of onboard passengers of bus b at time t
T_d^{min}	Minimum in-vehicle travel time of request d
ω_d	Pickup waiting time of request d
δ_d	In-vehicle travel time of request d
ω'_d	Response waiting time of request d
$\Phi_{b,k,d}$	Feasible position set of insertions of request d for bus b at decision epoch k
$\Delta_{b,k,d}(i,j)$	Incremental distance associated with inserting request d at positions $(i,j) \in \Phi_{b,k,d}$ for bus b at decision epoch k
Parameters	
α	Distance weight parameter
β_1	In-vehicle time weight parameter
β_2	Waiting time weight parameter
θ	Parameter of the policy network

(continued on next page)

(continued)

Notation	Definition
ω	Parameter of the value network
γ	Discount factor
λ	Smoothing parameter
ε_c	The hyperparameter of the clipping range
θ_{π}^{old}	The old policy-head parameters, which is used at the beginning of the current data collection phase
θ_{enc}	Pre-trained encoder parameters
θ_{π}	Policy-head parameters
θ_{vf}	Value-head parameters
η_p	Offline pre-training learning rate
η_f	Online fine-tuning learning rate
H	Number of PPO update steps for each rollout buffer.

Appendix B

Classification of simulation parameters and data sources.

Parameter	Specific Content	Source Category	Data Source	Processing Method
Simulation platform	Microscopic traffic simulation environment	SUMO default	SUMO 1.22.0	Used to construct a controlled simulation environment
Car-following model	Vehicle-following behavior	SUMO default	SUMO default car-following model	Default parameters were adopted
Vehicle dynamics parameters	Acceleration, deceleration, and other basic vehicle-operation parameters	SUMO default	SUMO default vehicle parameters	Default settings were adopted
Initial road network	Road topology, links, nodes, and intersections	OSM	OpenStreetMap	Initial SUMO road network generated from OSM
Traffic-signal locations	Signalized intersections	OSM / road-network data	OSM and generated SUMO network	Extracted from the road-network data
Background traffic OD	Background vehicle travel demand	Amap (Gaode)	Historical ride-hailing OD records from Amap	Used to generate background vehicle trips
Road speed	Link-level speed settings	Amap (Gaode)	Road-level speed information from Amap	Used to set or adjust link speeds
DRT passenger requests	Request time, origin stop, and destination stop	Bus company	Real-world DRT operational records	Reconstructed according to observed request times and OD stops
DRT stops	Passenger boarding and alighting locations	Bus company / manual correction	DRT stop records and SUMO NetEdit	Stops were placed according to operational records and manually checked in the network
DRT vehicles	Vehicle fleet and service-vehicle definitions	Bus company / simulation setting	DRT operational information and SUMO vehicle definitions	Explicitly represented in the simulation
Disconnected links	Broken or disconnected road links	Manual correction	SUMO NetEdit	Manually inspected and corrected
Road connectivity	Node-link connections and road accessibility	Manual correction	SUMO NetEdit	Manually inspected and corrected
Feasible DRT service routes	Routes available for DRT vehicle operations	Manual correction	SUMO NetEdit	Manually checked for route feasibility and accessibility
Traffic-signal timing	Fixed-cycle signal programs	SUMO default / simulation setting	SUMO signal-control programs	Fixed-cycle programs were adopted

Appendix C

To verify the practical decision-making logic of the TR-PPO agent, we present a series of real-world operational snapshots extracted from the Yuecheng simulation environment. These visualizations provide a microscopic view of how the trained model handles complex urban mobility tasks. The results are illustrated in Fig. C1 and Fig. C2.

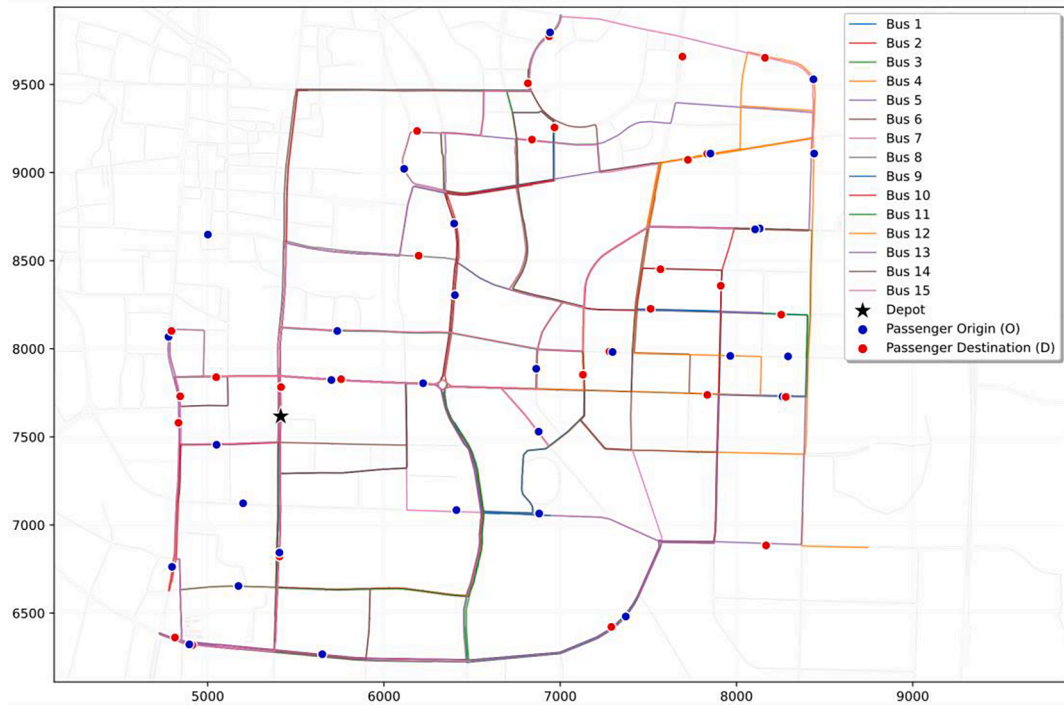


Fig. C1. Real-world scheduling trajectory examples during 7:00–9:00 (15 buses).

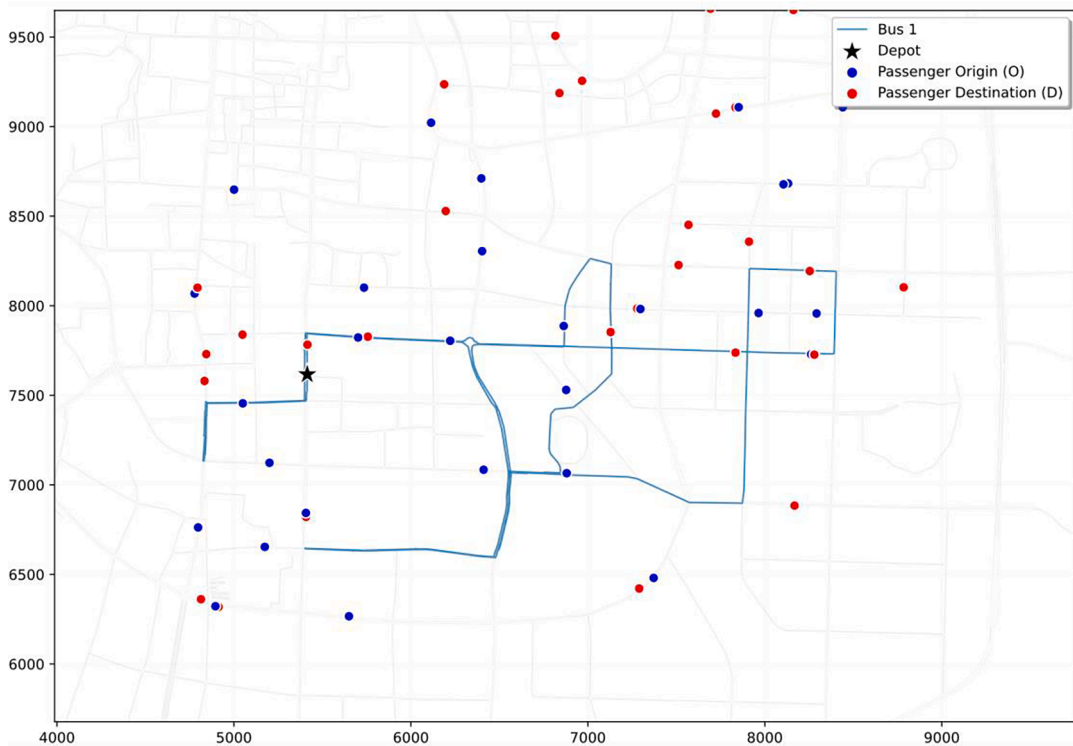


Fig. C2. Real-world scheduling trajectory examples during 7:00–9:00 (bus 1).

Data availability

Data will be made available on request.

References

- Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., Rus, D., 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *PNAS* 114 (3), 462–467.
- Ara, S., Mostafa Kamal Akib, Md.M., Shariar Rahman Oion, M., Rahman Shohel, Md.H., Noman Faysal Ridoy, Md., Kabita, F.A., Shahiduzzaman, M., 2023. Vehicle Routing Problem Solving Using Reinforcement Learning. In: 2023 26th International Conference on Computer and Information Technology (ICCIT). IEEE, pp. 1–6.
- Bruni, M.E., Guerriero, F., Beraldi, P., 2014. Designing robust routes for demand-responsive transport systems. *Transportation Research Part E: Logistics and Transportation Review* 70, 1–16.
- Castagna, A., Dusparic, I., 2021. Multi-agent transfer learning in reinforcement learning-based ride-sharing systems. In: 14th International Conference on Agents and Artificial Intelligence (ICAART), pp. 120–130.
- Charisis, A., Iliopoulou, C., Kepaptsoglou, K., 2018. DRT route design for the first/last mile problem: model and application to Athens, Greece. *Public Transp.* 10, 499–527.
- Costa, P.C., Cunha, C.B., Arbex, R.O., 2021. A simulation-optimization model for analyzing a demand responsive transit system for last-mile transportation: A case study in São Paulo, Brazil. *Case Studies on Transport Policy* 9, 1707–1714.
- Djavadian, S., Chow, J.Y.J., 2017. An agent-based day-to-day adjustment process for modeling ‘Mobility as a Service’ with a two-sided flexible transport market. *Transp. Res. B Methodol.* 104, 36–57.
- Du, K.-J., Yang, J.-Q., Wang, L., Han, X., Wang, H., Zhan, Z.-H., 2024. Multi-objective Demand Responsive Transit Scheduling in Smart City: A Multiple Populations Ant Colony System Approach. In: 2024 16th International Conference on Advanced Computational Intelligence (ICACI). IEEE, pp. 197–205.
- Feng, J., Gluzman, M., Dai, J.G., 2021. Scalable deep reinforcement learning for ride-hailing. *IEEE Control Syst. Lett.* 5, 2060–2065.
- Foa, S., Coppola, C., Grani, G., Palagi, L., 2022. Solving the vehicle routing problem with deep reinforcement learning. *arXiv preprint arXiv.2208.00202*. <https://doi.org/10.48550/arXiv.2208.00202>.
- Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M.P.D., Reis, M., Uchoa, E., Werneck, R.F., 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Math. Program.* 106, 491–511.
- Gammelli, D., Harrison, J., Yang, K., Pavone, M., Rodrigues, F., Pereira, F.C., 2023. Graph Reinforcement Learning for Network Control via Bi-Level Optimization. In: Proceedings of the 40th International Conference on Machine Learning. Presented at the International Conference on Machine Learning, pp. 10587–10610.
- Habib, Y., Filchenkov, A., 2022. Multi-agent reinforcement learning for multi vehicles one-commodity vehicle routing problem. *Procedia Comput. Sci.* 212, 418–428.
- Han, J., Xue, Q., Yuan, Y., Yang, X., Wu, J., 2025. Integrated optimization of train stopping plans and timetabling for urban rail transit based on dynamic demand. *Digital Transport. Safety* 4 (1), 42–49.
- Hu, Y., Dong, T., Li, S., 2025. Coordinating ride-pooling with public transit using Reward-Guided Conservative Q-Learning: An offline training and online fine-tuning reinforcement learning framework. *Transp. Res. Part C Emerging Technol.* 174, 105051.
- Jaw, J.-J., Odoni, A.R., Psarafitis, H.N., Wilson, N.H.M., 1986. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transp. Res. B Methodol.* 20, 243–257.
- Kalatzantonakis, P., Sifaleras, A., Samaras, N., 2023. A reinforcement learning-Variable neighborhood search method for the capacitated Vehicle Routing Problem. *Expert Syst. Appl.* 213, 118812.
- Lee, E., Cen, X., Lo, H.K., 2022. Scheduling zonal-based flexible bus service under dynamic stochastic demand and Time-dependent travel time. *Transportation Research Part E: Logistics and Transportation Review* 168, 102931.
- Lee, E., Cen, X., Lo, H.K., Ng, K.F., 2021. Designing zonal-based flexible bus services under stochastic demand. *Transp. Sci.* 55, 1280–1299.
- Levine, S., Kumar, A., Tucker, G., Fu, J., 2020. Offline reinforcement learning: tutorial, review, and perspectives on open problems. *arXiv preprint arXiv.2005.01643*. <https://doi.org/10.48550/arXiv.2005.01643>.
- Li, J., He, Z., Zhong, J., 2022. The multi-type demands oriented framework for flex-route transit design. *Sustainability* 14, 9727.
- Li, X., Quadrifoglio, L., 2010. Feeder transit services: Choosing between fixed and demand responsive policy. *Transp. Res. Part C Emerging Technol.* 18, 770–780.
- Liu, X., Qu, X., Ma, X., 2021. Improving flex-route transit services with modular autonomous vehicles. *Transportation Research Part E: Logistics and Transportation Review* 149, 102331.
- Liu, Y., Wu, F., Lyu, C., Li, S., Ye, J., Qu, X., 2022. Deep dispatching: A deep reinforcement learning approach for vehicle dispatching on online ride-hailing platform. *Transport. Res. Part E: Logistics Transport. Rev* 161, 102694.
- Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., Wiessner, E., 2018. Microscopic Traffic Simulation using SUMO. In: In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, pp. 2575–2582.
- Ng, A. Y., Harada, D., Russell, S., 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In: Proceedings of the Sixteenth International Conference on Machine Learning 99, 278–287.
- Ng, M.T.M., Engelhardt, R., Dandl, F., Mahmassani, H.S., Bogenberger, K., 2024. Semi-on-demand transit feeders with shared autonomous vehicles and reinforcement-learning-based zonal dispatching control. In: In: 2024 IEEE 27th International Conference on Intelligent Transportation Systems (ITSC). IEEE, pp. 2863–2868.
- Nickkar, A., Lee, Y.-J., Meskar, M., 2022. Developing an optimal algorithm for demand responsive feeder transit service accommodating temporary stops. *J. Public Transp.* 24, 100021.
- Park, K.-J., Jung, H.-Y., Jeon, S.-M., 2021. Feasibility analysis of demand response transport (DRT) over route buses using operating cost functions. *KSCE J. Civ. Eng.* 25, 4417–4424.
- Pecin, D., Contardo, C., Desaulniers, G., Uchoa, E., 2017. New Enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS J. Comput.* 29, 489–502.
- Pei, M., Lin, P., Du, J., Li, X., 2019. Operational design for a real-time flexible transit system considering passenger demand and willingness to pay. *IEEE Access* 7, 180305–180315.
- Peng, Z., Luo, W., Lu, Y., Shen, T., Gulino, C., Seff, A., Fu, J., 2025. Improving Agent Behaviors with RL Fine-Tuning for Autonomous Driving. In: Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., Varol, G. (Eds.), *Computer Vision – ECCV 2024, Lecture Notes in Computer Science*. Springer Nature Switzerland, Cham, pp. 165–181.
- Prudencio, R.F., Maximo, M.R.O.A., Colombini, E.L., 2024. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Trans. Neural Networks Learn. Syst.* 35, 10237–10257.
- Psarafitis, H.N., 1980. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transp. Sci.* 14, 130–154.
- Qiu, F., Li, W., An, C., 2014. A google maps-based flex-route transit scheduling system. In: In: 14th COTA International Conference of Transportation Professionals, pp. 247–257.
- Quadrifoglio, L., Dessouky, M.M., Palmer, K., 2007. An insertion heuristic for scheduling Mobility Allowance Shuttle Transit (MAST) services. *J. Sched.* 10, 25–40.
- Ryley, T.J., Stanley, P.A., Enoch, M.P., Zanni, A.M., Quddus, M.A., 2014. Investigating the contribution of Demand Responsive Transport to a sustainable local public transport system. *Res. Transp. Econ.* 48, 364–372.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv.1707.06347*. <https://doi.org/10.48550/arXiv.1707.06347>.
- Shaoxing Public Transport Group Co., Ltd, 2025. Dynamic Station, <https://sxbuszp.com/station>.

- Si, J., He, F., Lin, X., Tang, X., 2024. Vehicle dispatching and routing of on-demand intercity ride-pooling services: A multi-agent hierarchical reinforcement learning approach. *Transport. Res. Part E: Logistics Transport. Rev.* 186, 103551.
- Sun, Q., 2024. CrossLight: Offline-to-online reinforcement learning for cross-city traffic signal control. In: *In: KDD '24: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2765–2774.
- Sun, Z., Jia, X., Cai, Y., Ji, A., Lin, X., Liu, L., Wang, W., Tu, Y., 2025. Joint control of traffic signal phase sequence and timing: a deep reinforcement learning method. *Digital Transport. Safety* 4, 118–126.
- Tong, Y., Shi, D., Xu, Y., Lv, W., Qin, Z., Tang, X., 2023. Combinatorial optimization meets reinforcement learning: Effective taxi order dispatching at large-scale. *IEEE Trans. Knowl. Data Eng.* 35, 9812–9823.
- Voigt, S., Frank, M., Fontaine, P., Kuhn, H., 2022. Hybrid adaptive large neighborhood search for vehicle routing problems with depot location decisions. *Comput. Oper. Res.* 146, 105856.
- Wu, B., Zuo, X., Chen, G., Ai, G., Wan, X., 2024b. Multi-agent deep reinforcement learning based real-time planning approach for responsive customized bus routes. *Comput. Ind. Eng.* 188, 109840.
- Wu, M., Yu, C., Ma, W., An, K., Zhong, Z., 2022. Joint optimization of timetabling, vehicle scheduling, and ride-matching in a flexible multi-type shuttle bus system. *Transp. Res. Part C Emerging Technol.* 139, 103657.
- Wu, W., Zhu, Y., Liu, R., 2024a. Dynamic scheduling of flexible bus services with hybrid requests and fairness: Heuristics-guided multi-agent reinforcement learning with imitation learning. *Transp. Res. B Methodol.* 190, 103069.
- Xu, K., Saberi, M., Liu, W., 2022. Dynamic pricing and penalty strategies in a coupled market with ridesourcing service and taxi considering time-dependent order cancellation behaviour. *Transp. Res. Part C Emerging Technol.* 138, 103621.
- Yan, L., Shen, H., Kang, L., Zhao, J., Zhang, Z., Xu, C., 2022. MobiCharger: Optimal scheduling for cooperative EV-to-EV dynamic wireless charging. *IEEE Trans. Mob. Comput.* 1–17.
- Yoo, S., Kim, H., Lee, J., 2024. Combining reinforcement learning with genetic algorithm for many-to-many route optimization of autonomous vehicles. *IEEE Access* 12, 26931–26942.
- Zhang, J., Li, W., Wang, G., Yu, J., 2021. Feasibility study of transferring shared bicycle users with commuting demand to flex-route transit—A case study of Nanjing City. *China. Sustainability* 13, 6067.
- Zhang, J., Li, W., Zheng, Y., Guo, R., 2023. Dynamic Clustering meeting points strategy to improve operational service capability of flex-route transit. *Journal of Transportation Engineering, Part A: Systems* 149, 04023038.
- Zheng, Y., Gao, L., Li, W., 2021. Vehicle Routing and scheduling of flex-route transit under a dynamic operating environment. *Discret. Dyn. Nat. Soc.* 2021, 1–10.
- Zheng, Y., Li, W., Qiu, F., Wei, H., 2019. The benefits of introducing meeting points into flex-route transit services. *Transp. Res. Part C Emerging Technol.* 106, 98–112.
- Zheng, Y., Li, W., 2019. Flex-Route Transit Service with Different Degree of Dynamism. In: *CICTP 2019. Presented at the 19th COTA International Conference of Transportation Professionals, American Society of Civil Engineers*. pp. 4369–4380.
- Zhou, Z., Peng, A., Li, Q., Levine, S., Kumar, A., 2025. Efficient Online Reinforcement Learning Fine-Tuning Need Not Retain Offline Data. In: *International Conference on Learning Representations 2025 (ICLR)*, 32343–32368.