Article:
Wu, W., Zhu, Y. and Liu, R. orcid.org/0000-0003-0627-3184 (2024) Dynamic scheduling of flexible bus services with hybrid requests and fairness: Heuristics-guided multi-agent reinforcement learning with imitation learning. Transportation Research Part B: Methodological, 190. 103069. ISSN: 0191-2615

https://doi.org/10.1016/j.trb.2024.103069

Please cite the paper as:

Wu W, Zhu Y and Liu R (2024) Dynamic scheduling of flexible bus services with hybrid requests and fairness: Heuristics-guided multi-agent reinforcement learning with imitation learning. Transportation Research Part B. 190, 103069.
https://doi.org/10.1016/j.trb.2024.103069

Dynamic scheduling of flexible bus services with hybrid requests and fairness: Heuristics-guided multi-agent reinforcement learning with imitation learning

Weitiao Wu[a1], Yanchen Zhu[a], Ronghui Liu[b]

a. School of Civil Engineering and Transportation, South China University of Technology, Guangzhou 510641, China
b. Institute for Transport Studies, University of Leeds, Leeds, LS2 9JT, United Kingdom

Abstract: Flexible bus is a class of demand-responsive transit that provides door-to-door service. It is gaining popularity now but also encounters many challenges, such as high dynamism, immediacy requirements, and financial sustainability. Scientific literature designs flexible bus services only for reservation demand, overlooking the potential market for immediate demand that can improve ride pooling and financial sustainability. The increasing availability of historical travel demand data provides opportunities for leveraging future demand prediction in optimizing fleet utilization. This study investigates prediction failure risk-aware dynamic scheduling flexible bus services (FBS) with hybrid requests allowing for both reservation and immediate demand. Equity in request waiting time for immediate demand is emphasized as a key objective. We model this problem as a multi-objective Markov decision process to jointly optimize vehicle routing, timetable, holding control and passenger assignment. To solve this problem, we develop a novel heuristics-guided multi-agent reinforcement learning (MARL) framework entailing three salient features: 1) incorporating the demand forecasting and prediction error correction modules into the MARL framework; 2) combining the benefits of MARL, local search algorithm, and imitation learning (IL) to improve solution quality; 3) incorporating an improved strategy in action selection with time-related information about spatio-temporal relationships between vehicles and passengers to enhance training efficiency. These enhancements are general methodological contributions to the artificial intelligence and operations research communities. Numerical experiments show that our proposed method is comparable to prevailing benchmark methods both with respect to training stability and solution quality. The benefit of demand prediction is significant even when the prediction is imperfect. Our model and algorithm are applied to a real-world case study in Guangzhou, China. Managerial insights are also provided.

Keywords: Flexible bus; Multi-agent reinforcement learning; Imitation learning; Local search; Demand prediction

[1] Corresponding author, E-mail: ctwtwu@scut.edu.cn

1. Introduction

Public transportation plays an important role in urban mobility and development. Its operation usually compromises between the level of service and economic viability. Although conventional fixed-line bus services are cost-efficient in high-demand areas, it is not affordable in low-density demand areas. Recently, the advent of information technology has enabled demand-responsive mobility services and spawned a new generation of the travel industry, such as Uber, Lyft, and Didi Chuxing. Despite being convenient and offering a direct door-to-door service, taxis and ride-hailing have high operation and environmental costs. With a larger capacity and ease of sharing rides, flexible bus services bridge the transportation gap, providing customized services to passengers at a relatively low price.

Successful implementation of flexible bus services (FBS) encounters many challenges. In addition to high dynamism and immediacy requirements, FBS are facing the barriers of low occupancy numbers and strong dependence on public funding (Schasché et al., 2022). Nowadays, FBS is growing in a competitive market with car-hailing as their main rival. FBS usually operate under a reservation request mode, where time windows should be specified in advance. Customers today, however, have higher expectations, preferring easy-to-use service delivery allowing for issuing riding requests in real-time and getting aboard as soon as possible, similar to the immediate demand serviced by taxis and car-hailing. Therefore, it becomes essential to raise the user acceptance of FBS and increase ride pooling because it affects their profitability and long-term survival.

In an attempt to improve the utilization of public transport, integrated passenger-freight transport was recently presented as a solution (Cavallaro and Nocera, 2023; Fehn et al., 2023; Donne et al., 2023; He et al., 2023). Inspired by the integrated passenger-freight transport research, the hybrid request mode allowing for both reservation and immediate demand is one appealing solution for raising user acceptance and improving ride pooling for FBS. Under the hybrid request mode, customers can choose either the reservation request pattern or the immediate request pattern. Different to the traditional reservation request, the immediate request does not need to specify time windows, similar to taxi-like ad hoc services. Such a hybrid request mode can attract more users, which will enable service improvement and initiate a virtuous cycle further attracting new users. Realizing the potential benefits of this request option, a few on-demand transit platforms offered immediate request options, enabling trip behavior to book ride services in real-time. For example, an emerging shuttle service platform in Guangzhou, called Cheying Net, has launched services with hybrid requests by making the running states of en-route buses transparent to the users on the App interface.

Scientific literature focuses on the FBS optimization problem with reservation requests, overlooking the potential hybrid request mode that can optimize their utilization. This study envisions an emerging FBS with such hybrid requests. The problem jointly optimizes vehicle routing, timetable, holding control, and passenger assignment. Seeking effective solutions for this problem is difficult in two ways. First, the service provider wants to optimize both the service level and operation cost. However, in this hybrid request mode, the minimum-cost operation can raise the unfairness issue for immediate requests. For example, vehicles are more likely to serve the high-demand area under minimum-cost operation, such that the waiting time for requests in the low-demand area will be relatively long, leading to passengers' dissatisfaction and degradation of the level of service, among other disappointing consequences. As such, a fundamental question for FBS management with hybrid requests is how to make a trade-off between the economic viability of the system and maintaining good service for passengers, and the trade-off between different groups of passengers. Second, since the immediate requests are highly stochastic, this problem often handles travel demand that reveals over time and requires decision-making considering stochastic future requests. The service provider must make a decision for already known demand, and anticipate its impact on the future for

which requests are not yet accessible. Therefore, an efficient approach that can maximize long-term rewards and that can make dynamic decisions is highly desired for dynamic FBS.

Traditionally, dynamic FBS optimization problems are usually addressed by the repetitive one-period static model and construction insertion heuristics, without leveraging historical demand data. Unfortunately, traditional methods are generally sub-optimal because they fail to consider the future dynamics and measures of the whole system. For the dynamic FBS services with hybrid requests, it is significant to find high-quality solutions promptly in response to the time-varying environment considering stochastic future requests. An emerging technique to address complex sequential decision-making problems is reinforcement learning (RL). Despite its outstanding exploration ability and generalizability, there are significant drawbacks when applying traditional RL methods to solve our problem: (a) The training process is usually time-consuming; (b) Despite the capability of solving large-scale dynamic optimization problems, the solution quality is unstable and still not comparable to those of the exact algorithms and the state-of-the-art heuristic algorithms; (c) They do not leverage historical demand data for future demand prediction to further improve foresight in the planning scheme. These issues, while secondary to the RL community, are central to the operations research and transportation science community that has been widely investigating them. Imitation learning (IL) is a supervised learning method for learning skills via observations of an expert demonstration, which avoids complicated reward design and large-scale random exploration (Delgado and Oyedele, 2022). In the policy evaluation process, the actions are based on the observations, which can not only enhance the training stability but also avoid the policy evaluation outside the observations, so that the best policy in the behavioral strategy library can be learned. However, the IL is inherently flawed since the actions are taken within the observations, which limits the generalizability and is difficult to outperform the original behavioral policy. As such, performance dramatically decreases when deployed to unseen environments and is inconsistent against varying conditions.

Unlike traditional model-driven approaches relying on many heuristic assumptions and equations, we formulate the dynamic FBS optimization problem as a Markov decision process in an unknown environment (i.e., model-free). The problem is solved by a multi-agent deep reinforcement learning (MARL) approach algorithm that learns and responds to the prevailing traffic conditions without requiring an underlying system model. To overcome the drawbacks of traditional RL approaches, we propose a novel MARL approach integrating IL and local search heuristics to combine the benefits of each algorithm. The local search algorithm for expert demonstration is devised based on the characteristics of the problem. Furthermore, an improved strategy to enhance exploration efficiency for agents in the training process is also proposed. Our approach not only improves the solution quality by the local search algorithm but also enables RL to learn the better solution samples obtained by the local search algorithm with the mechanism of imitation learning, thus essentially improving the performance of RL. Commendably, by leveraging the historical travel demand data in future demand prediction, we embed a look-ahead policy and prediction error correction mechanism into the newly developed MARL framework for moving beyond FBS into a prediction failure risk-aware operation mode. Our model and algorithm are applied to a real-world case study in Guangzhou, China. Managerial insights are also provided.

2. Literature review

In this section, we summarize the literature regarding demand-responsive transit. In particular, we compare with existing studies on solution approaches, to further highlight the contributions of this paper.

2.1. The demand-responsive transit problem

The demand-responsive transit (DRT) problems have drawn much research attention in the field of shared mobility and public transportation. The first trial of on-demand flexible buses, called dial-a-ride services, was provided to the elderly and disabled in the USA in 1970 and was first formulated by Psaraftis in 1980 (Psaraftis, 1980). Later, its application was extended to other areas, such as patient transportation (Lim et al., 2017), first-mile feeder services (Ren et al., 2022), and last-mile feeder services (Shehadeh et al., 2021). For an overview of research on DRT problems, see Ho et al. (2018) and Vansteenwegen et al. (2022). The DRT problem is a variant of vehicle routing problems with pickup and delivery. Compared to ride-hailing services, the DRT operates with the following unique features: (a) fixed terminals (depots) are processed; (b) the stopping locations are selected from candidate stop locations, and (c) the vehicle trajectory is closed-loop to pull out and pull in the vehicle. According to request availability, the DRT problem can be categorized into static and dynamic versions. For the dynamic version, the vehicle routes are dynamically modified based on the en-route pop-up passenger requests. According to the demand distribution, the DRT service network includes many-to-one (Jiang et al., 2020), one-to-many (Wang, 2019), and many-to-many (Daganzo, 1978) patterns. The many-to-one and one-to-many patterns correspond to the feeder service for first-mile and last-mile commuters, respectively (Montenegro et al., 2021). In comparison, the many-to-many pattern is more generalized in that each node can act as both the pick-up and drop-off point.

Early studies on DRT feature homogeneous vehicles (Psaraftis, 1980; Desrosiers et al., 1986). Subsequent research has extended to considering more realistic and complex factors. To meet diverse travel demands (e.g., patient transportation, and wheelchair requirements), some researchers investigated the dial-a-ride problem with heterogeneous vehicles (Detti et al., 2017; Braekers et al., 2014; Masmoudi et al., 2016). Masson et al. (2014) presented a dial-a-ride problem with transfers. They showed that introducing transfer points can save operation costs by up to 8%. Posada et al. (2017) integrated the dial-a-ride service with fixed bus routes, where passengers may change vehicles during their journey. To improve the service quality, Braekers and Kovacs (2016) presented a multi-period dial-a-ride problem considering driver consistency. Molenbruch et al. (2017) introduced horizontal cooperation in the dial-a-ride service, in which passenger requests can be exchanged between providers to jointly plan vehicle routes. Tong et al. (2017) designed customized bus services considering passenger-to-vehicle assignment and vehicle routing. Lyu et al. (2019) proposed a solution framework for DRT service network design associated with bus stop locations, routing, and timetabling. Recently, Wu et al. (2024) investigated a dynamic dial-a-ride problem considering spatial demand correlation and request canceellation via approximate dynamic programming and scenario approach.

In terms of demand patterns, one category of research designed flexible bus services using parsimonious continuous models, assuming uniformly distributed demand across space (Daganzo, 1978; Diana et al, 2006; Quadrifoglio and Li, 2009; Kim and Schonfeld, 2014, 2015; Chen and Nie 2017a, b; Zhang et al, 2017). There are also a handful of works focusing on designing flexible bus services under stochastic demand. For instance, Lee et al. (2021) optimized zonal-based flexible bus services considering stochastic demand. The problem is formulated as a two-stage stochastic programming model with recourse. Later, Lee et al. (2022) extended their work to consider time-dependent travel time and dynamic demand.

2.2. Solution approach

Common approaches for DRT problems can be classified into exact algorithms (Braekers and Kovacs, 2016; Schenekemberg et al., 2022) and heuristic algorithms (Masmoudi et al., 2016, 2017; Montenegro et al., 2021; Ren et al., 2022). Although the problem can be solved to its global optimal solution by exact algorithms (e.g., branch-and-cut, branch-and-

price), the computation burden is very large such that they do not scale well for large systems and real-time decision-making. As such, exact algorithms can be only applied to the static DRT problem. Heuristic algorithms can generally provide feasible solutions within an acceptable time (Braekers et al., 2014; Wu et al., 2021; Wu and Li, 2024). However, such model-based algorithms rely heavily on problem characteristics and prior knowledge, which are easily affected by parameter estimation, modelling error, and problem size, and have low adaptability to random dynamic complex environments.

Reinforcement learning (RL) has been applied in logistics and transportation services for various contexts, such as taxi dispatching (Lin et al., 2018; Mao et al., 2020; Liu et al., 2022), bike-sharing repositioning (Li et al., 2018), and train control (Wang et al., 2023). Yan et al. (2022) reported that RL applications excel in solving combinatorial optimization problems with high performance compared to exact algorithms or heuristic algorithms, including the traveler salesman problem (Bello et al., 2016; Nazari et al., 2018; Kool et al., 2019; Drori et al, 2020) and vehicle routing problem (Chen and Tian, 2019; Kalakanti et al., 2019). Some studies (Joe and Lau, 2020; Zhang et al., 2023) addressed the methods for solving routing problems of dynamic scenarios. Recently, there has been an interest in using RL approaches to solve dial-a-ride problems associated with ridesharing issues. Oda and Joe-Wong (2018) proposed a DQN-based framework to learn the optimal zone for idle vehicles, and Singh et al. (2019) extended this approach to accommodate multiple riders in a ridesharing vehicle. However, these studies used a decentralized training process, where each vehicle independently learns its own DQN problem without communication with other vehicles, reducing the reliability and stability of the training process. Zhou et al. (2022) developed a multi-agent reinforcement learning (MARL) framework with centralized training and decentralized execution (CTDE) to solve the dynamic electric vehicle dispatching problem. However, these studies only use standard RL approaches to address the targeted problems, which still face challenges related to the drawbacks of RL, such as slow convergence. To address this issue, Zhao et al. (2020) proposed a hybrid framework for solving the vehicle routing problem, which improves the initial solution obtained by the RL approach through a local search method. Additionally, Ahamed et al. (2021) proposed a centralized approach that combines heuristic rules to solve the medium-scale and large-scale crowdsourced urban delivery problem. Despite the improvements in algorithm performance seen in these studies, they have not fundamentally enhanced the learning policy of RL structures with valuable heuristic experiences or addressed the drawbacks of standard RL approaches.

2.3. Objectives and contributions

A few gaps are identified from the literature. First, existing studies only address reservation requests specifying time windows, overlooking the potential hybrid request mode that can optimize their utilization. Second, traditional research predominantly focuses on assigning vehicles only to currently known requests without leveraging the historical demand data. The historical travel demand data from the day-to-day operation can be used to predict future demand and dispatch vehicles preemptively, which contributes to improving service quality and system efficiency. Third, although substantial research effort has been made to dynamic FBS optimization problems, existing research usually relies on the repetitive one-period static model and construction insertion heuristics, which are sub-optimal since they fail to consider the future dynamics and measures of the whole system.

In this study, we set out to investigate prediction failure risk-aware dynamic scheduling flexible bus services with hybrid requests and fairness considerations, using a novel multi-agent deep reinforcement learning framework integrating with imitation learning and heuristics. Overall, our study makes contributions to the problem as well as the general methodology of the RL:

First, we investigate dynamic scheduling flexible bus services with hybrid requests that can potentially increase ride pooling and resulting profitability, a topic with limited research attention. Our problem jointly optimizes vehicle routing, timetable, holding control, and passenger assignment, which is a new generalization to the DRT problem where hybrid requests, multiple visits, and future predicted demand are explicitly considered.

Second, we devise a novel representation of system states and actions to characterize this stochastic dynamic problem. Our work goes beyond the simple adoption of the MARL in the existing literature, by making three methodological contributions as follows:

- In terms of improving the foresight of planning, we enhance the RL framework by embedding the module of demand forecasting to leverage historic demand data and account for anticipated future demand. We achieve this by designing a rolling optimization framework incorporating demand prediction method and prediction failure risk-aware decision-making mechanism for prediction errors.
- In terms of improving exploration, we not only propose a pruning strategy to reduce action space to a feasible compact space, but also develop a novel representation of the state of each agent with time-related information indicating how emergent for a specific bus to serve a request. Such representation highlights the spatio-temporal relationship in the vehicle-passenger matching process and further allows for an improved strategy to help buses heuristically search out valuable experiences as soon as possible.
- In terms of improving training, we integrate our proposed MARL framework with Variable Neighborhood Descent algorithm (VND) and imitation learning (IL). Three customized operators are designed in the VND algorithm, where the novel concept of 'entropy' is applied to guide the applications of the three operators to improve solution quality. This method can leverage the information provided by VND to infer more valuable and representative decisions.

Third, we conduct extensive numerical experiments and real-world applications to verify our model and solution approach and derive significant managerial insights.

The rest of this paper is organized as follows. In Section 3, we describe the benefits and challenges of operation with demand prediction. In Section 4 the model is presented, followed by the integrated RL method in Section 5. In Section 6 computational experiments and a real-world application are conducted and managerial insights are provided. At last, the concluding remarks are provided.

3. Benefits and challenges of operation with demand prediction

Scientific literature predominantly focuses on scheduling FBS with currently known requests, overlooking the importance of future requests that can optimize their utilization. For example, we consider a vehicle and two requests, request 1 and request 2. Request 1's presence is known but far from the vehicle. Request 2 is located between the vehicle and request 1, but is unknown at this time. Without demand prediction, the platform would reject request 1 due to high costs. Conversely, if the presence of request 2 can be anticipated, it turns out that a more profitable plan can be expected that serves request 2 first and request 1 later. The example shows that integrating demand prediction into the schedule is essential to make informative decisions.

However, this is a particularly difficult task since any prediction model is difficult to provide a perfect prediction. In practice, prediction errors can exist in the future demand due to its stochastic attributes. The outputs of the prediction models

(e.g., machine learning models) can also be variable due to unavoidable learning errors. The presence of prediction errors could undermine the quality of the planned scheme. For instance, the waste of capacity might arise because any request does not occur as predicted. It may cause a failed planning scheme due to an unsuccessful match between the predicted requests and buses. Therefore, to warrant the benefits of demand prediction, it is imperative to develop a prediction error correction mechanism to eliminate prediction errors and rectify the policies promptly.

4. Problem formulation

4.1. Problem description, assumptions, and notations

The FBS service area is modelled as a directed network $\mathcal{G}(\mathcal{S}, \mathcal{A})$. $\mathcal{S} = I \cup J$ denotes the set of nodes, where $I = \{i | i = 1,2, \cdots, |I|\}$ and $J = \{j | j = |I| + 1, |I| + 2, \cdots, |I| + |J|\}$ denote the set of physical bus stops and depots, respectively. Each node denotes a physical stop where vehicles pick up/drop off passengers. The links of the network represent the road segments between stops. $K = \{k | k = 1,2, \ldots, |K|\}$ denotes the set of vehicles. $\mathcal{A}$ denotes the set of arcs between nodes in $\mathcal{S}$.

In terms of requests, both reservation and immediate requests include the information about the desired pick-up stop $i_n^p \in I$, drop-off stop $i_n^d \in I$, the number of passengers q_n and desired departure time windows $[t_n^e, t_n^l]$. By dividing operation duration into multiple periods, the service provider pre-collects reservation requests and allocates them into corresponding periods before operation. In this study, the reservation requests are collected one day in advance (before operation). During operation, with the joining of newly immediate requests, the service provider needs to dynamically accommodate them into the planning of the latest period. The immediate requests can be rejected to avoid over-allocation of fleet resources to ensure the economic viability of the FBS operation and excessive waiting time for passengers as a quitting mechanism.

The service provider wants to optimize both the level of service and operation cost. However, the objectives of the service level and operation cost may be competing, in that, a lower operation cost can result in more time window deviation and even request rejection. Moreover, due to the spatio-temporal randomness of immediate requests, the passenger waiting time for certain immediate requests can vary largely, raising the issue of unfairness. To this end, we investigate the prediction failure risk-aware dynamic scheduling flexible bus services with hybrid requests and fairness considerations, leveraging historical requests in future demand prediction. Since the holding actions can increase the flexibility of planning routes, vehicle holding is allowed at both depots and immediate bus stops. The problem jointly optimizes vehicle routing, timetable, holding control and the selection of immediate demand. The dual objectives are (a) minimizing total system cost, and (b) optimizing waiting time fairness for immediate requests.

To facilitate model development, the following assumptions are made.

(1) Multi-period optimization is applied to plan the flexible bus operations throughout the time horizon. However, we do not limit each bus to return to any depot at the end of each period except the last period.

(2) In each period, the buses depart from and return to the depots. This is done primarily for crew scheduling where the continuous work duration should be limited to avoid potential fatigue and traffic safety issues.

(3) Each stop can be visited by multiple vehicles multiple times in each period.

(4) All vehicles are homogeneous with the same capacity.

(5) The service provider is allowed to reject immediate requests.

Table 1 provides the notations and definitions primarily used in the model.

Table 1 Notations primarily used in the model and algorithm

Symbols	Definitions
Sets and indices	
I	Set of stops, $I = \{i \mid i = 1,2,\cdots,\lvert I\rvert\}$.
J	Set of depots, $J = \{j \mid j = \lvert I\rvert + 1, \lvert I\rvert + 2,\cdots,\lvert I\rvert + \lvert J\rvert\}$.
K	Set of buses, $K = \{k \mid k = 1,2,\cdots,\lvert K\rvert\}$, where k denotes the index of a bus.
G	Set of periods, $G = \{g \mid g = 1,2,\cdots,\lvert G\rvert\}$, where g denotes the index of a period.
N_g	Set of riding requests in period g, which represents the set of reservation requests $\dot{N}_g$ and the set of immediate requests $\ddot{N}_g$, i.e., $N_g = \dot{N}_g \cup \ddot{N}_g$. $N_g = \{n \mid n = 1,2,\cdots,\lvert N_g\rvert\}$, where n denotes the index of a request.
$\widetilde{N}_g$	Set of predicted requests in period g.
T	Set of time intervals, $T = \{t \mid t = 1,2,\cdots,\lvert T\rvert\}$.
τ	Index of training step.
FBS model with demand prediction	
$r_{g,k}$	Planned route of bus k of period g.
C_g^f	Operation cost of period g.
C_g^u	User cost of period g, including both the time windows penalty and rejection penalty cost.
ζ_g	Fairness indicator of immediate requests waiting time of period g.
i_n^p	Pick-up stop of request n.
i_n^d	Drop-off stop of request n.
$DIS(\cdot,\cdot)$	Distance from one node to another.
t_n^e	Earliest departure time for reservation request or the submission time for immediate request $n \in N_g$.
t_n^l	Latest departure time for request $n \in N_g, \forall g \in G$.
W_n	Waiting time for request $n \in N_g, \forall g \in G$.
t_n^p	Pick-up time of request n.
t_n^d	Drop-off time when request n was delivered.
t_n^b	Rejection time for immediate request $n \in \ddot{N}_g, \forall g \in G$.
q_n	The number of passengers for request n.
$\tilde{q}_n$	Predicted number of passengers for request n.
y_n^b	Binary variable to determine whether request n is rejected, $y_n^b = 1$ represents request n is rejected; otherwise $y_n^b = 0$ represents request n is accepted.
$i_{g,t,k}$	Location of bus k at time interval t of period g.
$z_{g,t,k}$	The number of onboard passengers of bus k at time interval t of period g.
$t_{g,t,k}^{arl}$	Arrival timestamp of bus k after it finishes the decision made at interval t of period g.
$N^{\alpha}(i_{g,t,k})$	Set of on-board requests when bus k arrives at location $i_{g,t,k}$ at time interval t of period g.
Z	Vehicle capacity.
H	Period length.
α	Tardiness coefficient, representing the ratio between the planned in-vehicle travel time and direct (shortest path) travel time is not greater than α.
β	Scaled coefficient for fairness rewards to cost rewards.
δ^b	Unit rejection penalty cost.
δ^l	Unit late arrival penalty cost.
$\delta^{\mathcal{H}}$	Unit holding time penalty cost.
δ^f	Unit mileage cost.
ρ	The weight of the objective.
vel	Average running speed.
Integrating RL and local search framework	
$S_{g,t}$	State at time interval t of period g, including the vehicle state of each bus $S_{g,t,k}^K, \forall k \in K$ and the request state $S_{g,t,n}^N, \forall n \in N$.
$o_{g,t,k}$	Local state of bus k at time interval t of period g.
$\xi_{g,t,n}^p / \xi_{g,t,n}^d$	Time-related information of request n at time interval t of period g, indicating the urgency that picking up/delivering n, which is independent for each bus $k \in K$.
$a_{g,t,k}$	Action of bus k at time interval t of period g, including movement action $a_{g,t,k}^{\mathcal{M}} \in \{i_n^p, i_n^d, j\}$ and holding time action $a_{g,t,k}^{\mathcal{H}} \in \mathbb{R}$.
$\hat{A}_{g,t,k}$	Valid action space of bus k at time interval t of period g.

$R_{g,t}$	Reward at time interval t of period g, including cost reward and fairness reward.
π_{θ_g}	Policy of period g, which is characterized by a neural network with parameter of θ_g.
Q_{tot}/Q_k	Joint/decomposed (of bus k) Q function network.
Γ	Exploration function to obtain action under the improved ϵ-greedy strategy.
ℓ	The service loop by separating a complete route in the local search framework.
μ_1, μ_2, μ_3	Entropy metrics indicating the potential to improve solution quality in local search framework.
$\Delta(r)/\Delta(\ell)/\Delta(n)$	Duration of the route r/ loop ℓ/request n.
$\mathcal{N}_n(\cdot)$	Neighborhood of a solution when n opt is applied in local search framework.
e^{RL}/e^{LS}	Experience tuple of RL/Local search algorithm.
L	Loss function including TD-error loss L^{TD} and imitation learning loss L^{IL}.
ς	Margin function to avoid zero-value in IL.
γ	Discount factor on future rewards.
ϵ_1, ϵ_2	Initial exploration rates used in improved ϵ-greedy strategy.
$\tau_{\max}^{\epsilon_1}, \tau_{\max}^{\epsilon_2}$	Maximum decay steps for ϵ_1 and ϵ_2 used in the exploration rate decay mechanism.
$\tau_{\max}^{RL}/\tau_{\max}^{LS}$	Maximum steps/iterations of RL/Local search algorithm.
ω^{RL}/ω^{IL}	Batch size of RL/IL algorithm.

4.2. Preliminaries

4.2.1. Rolling horizon framework

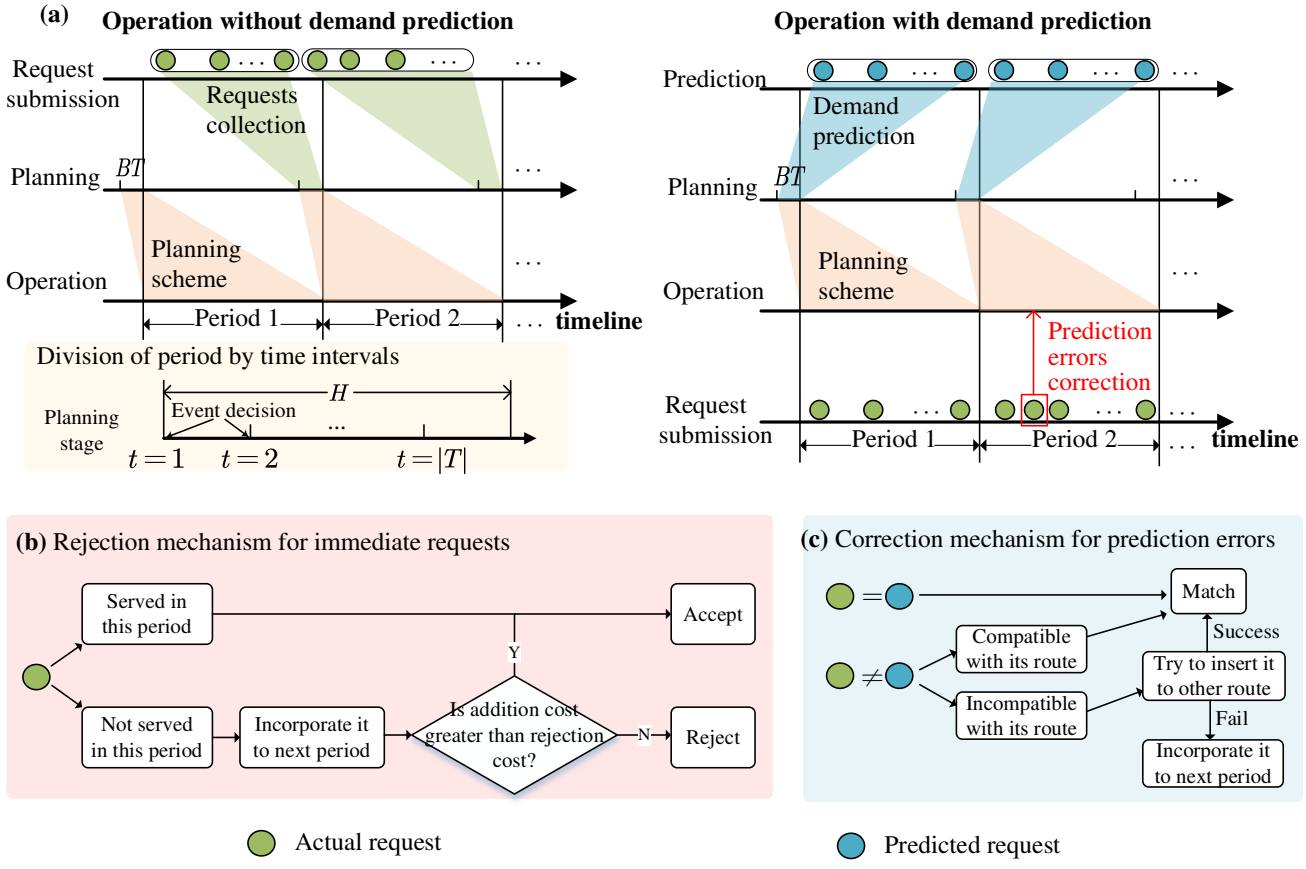

Fig. 1 Rolling horizon planning framework

The whole planning framework is illustrated in Fig. 1. Specifically, to handle the high dynamism of the problem, as shown in Fig. 1(a), a rolling horizon method is employed to transform the problem into a series of interdependent sub-problems. The whole planning horizon is divided into $|G|$ periods with a length of H. The set of requests corresponding to

each period g is N_g, including $\dot{N}_g$ (the set of reservation requests collected before the time horizon) and $\ddot{N}_g$ (the set of immediate requests submitted during the planning horizon and incorporated into the planning stage of period g). Note that, under the operation with demand prediction, the immediate requests of $\ddot{N}_g$ are predicted to be submitted within period g, while under the operation without demand prediction, they are submitted before period g but incorporated into $\ddot{N}_g$ at the planning stage of period g. Then, to construct the MDP model, each period $g \in G$ is divided into $|T|$ time intervals. Each agent in MDP makes the decision only at the beginning of each time interval. Therefore, the maximum number of decisions for each vehicle in one period is $|T|$. We set the duration of the optimization execution stage as buffer time (BT), during which the system plans for the next period. In addition, to tackle the possible issues that may arise in the dynamic scheduling process, we also propose the rejection mechanism for immediate requests (Fig. 1(b)) and correction mechanism for prediction errors (Fig.1(c)), which will be described as follows.

4.2.2. Rejection mechanism for immediate requests

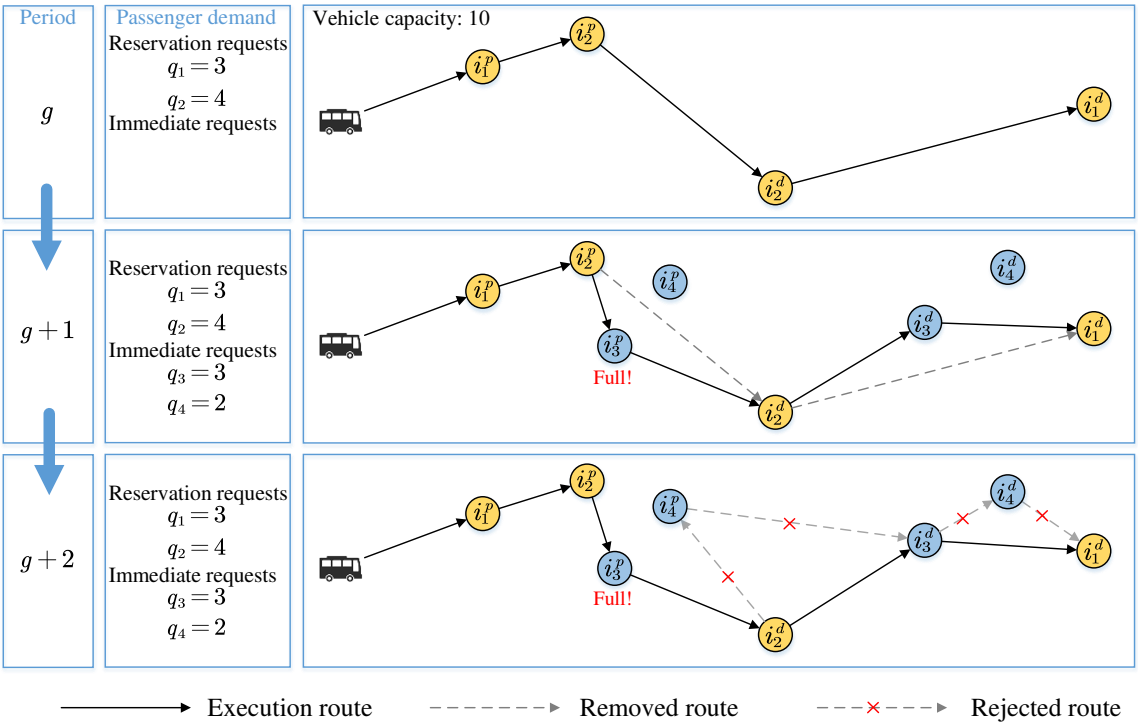

Fig. 2 Illustration of request insertion and rejection

Fig. 1(b) presents the rejection mechanism for immediate requests. Within the context of the rolling planning approach that incorporates the insertion of new immediate requests, the process of rejecting requests can become complex. Fig. 2 provides a toy example of the request insertion and rejection operations in our approach. In period g, a bus with a capacity of 10 seats is scheduled to serve two reservation requests (requests 1 and 2) in the order by the planned route. Moving into period $g+1$, with the joining of two immediate requests (requests 3 and 4), the system needs to adjust the original route to accommodate these new requests. However, due to capacity constraints, the bus can only accommodate request 3 in this period, leaving request 4 pending, despite the geographical proximity of their pick-up locations. In period $g+2$, the service provider has two options: 1) integrating request 4 in the current period, incurring additional waiting time for request 4 and extra costs for the bus to detour; or 2) rejecting request 4, thereby incurring a rejection penalty cost. While both options compromise the solution quality relative to the initial route for $g+1$, it is rational for the service provider to opt for the alternative with the lower additional cost (including possible late arrival penalty for in-planned reservation requests, since

they cannot be rejected). In this scenario, rejecting request 4 is deemed more advantageous. Thus, the system opts to reject request 4, accepting the penalty for rejection. Upon rejection, the rejected time t_4^b for request 4 is set to $(g+1)\cdot H$, i.e., the beginning of $g+2$, and the binary variable y_n^b is set to 1, which means the request 4 has been marked as rejected.

In summary, there are two main advantages to introducing a rejection mechanism for immediate requests: 1) for passengers, it includes a quitting mechanism to avoid excessively long waiting times for immediate requests, and 2) for the service provider, it enables the potential to improve solution quality, albeit at a cost, by avoiding detours that deviate significantly from the original route.

4.2.3. Correction mechanism for prediction errors

As shown in Fig. 1(c), the correction of prediction errors takes place once any request is submitted not as predicted. Whether the predicted immediate request n is submitted as predicted could not be verified until its predicted submission time. Therefore, if bus k is planned to pick up request n but arrives at its pick-up stop i_n^p before its submission time t_n^e, then bus k will hold at i_n^p till t_n^e, and verify whether request n is submitted at t_n^e. If request n is submitted as predicted, then the pick-up time of request n, t_n^p equals to t_n^e; otherwise, the prediction errors about request n occur and the correction is required.

In general, there are three types of prediction errors of request n, which are: 1) incorrect number of passengers, 2) incorrect submission time, and 3) incorrect stops. The condition triggering correction is whether the constraints are violated, which can be derived from planned routes of the existing scheme and updated information of request n. Once the constraints are not met due to the prediction errors, the corresponding correction is required to ensure the effectiveness of the planned scheme. Suppose the predicted number of passengers of request n is $\tilde{q}_n$, its actual number of passengers is q_n, and it is planned to be assigned to bus k. When $\tilde{q}_n \neq q_n$, it is required to identify whether the capacity of bus k still meets the capacity constraint (Section 4.5.1), not only when picking up request n but also within the following route of the planned scheme. If the capacity of bus k in the updated route still meets the capacity constraint, the service provider could execute the existing scheme; otherwise, bus k could not serve request n as expected, which means the existing scheme failed and the corresponding correction is necessary to ensure an effective schedule. The correction begins with searching out whether any buses in operation can serve request n, which involves a special operation (specified in Appendix B). If there is not any bus that could serve request n, it will be incorporated into the requests set for the next period.

4.2.4. Over-period compensation

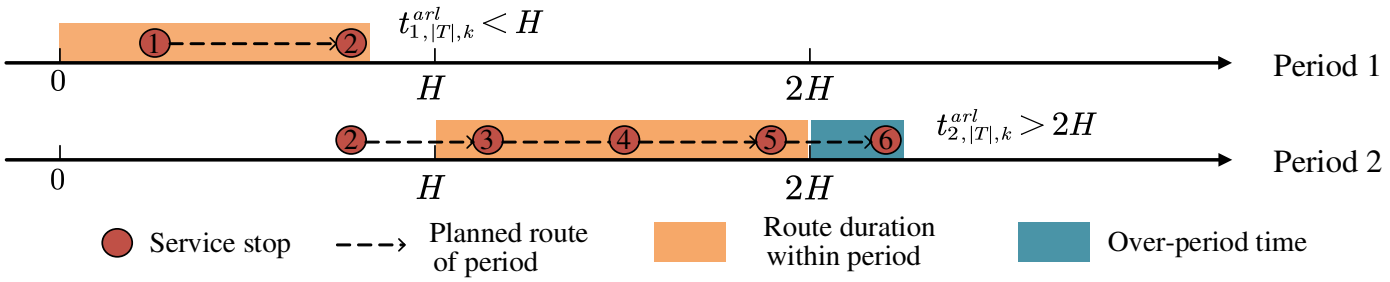

Fig. 3 Illustration of over-period compensation

The rolling horizon framework, however, introduces another issue in that the duration of some planned routes might be beyond the length of the period, which we call 'over-period'. To address this issue, we propose a mechanism of over-period compensation. For each bus $k \in K$, in each period we initialize a variable $t_{g,t,k}^{arl}$ to denote the timestamp when bus k finishes the current decision at time interval t, so when the period g ends (i.e., $t = |T|$), $t_{g,|T|,k}^{arl}$ is its route duration. If $t_{g,|T|,k}^{arl} > g \cdot H$,

then the duration of $r_{g,k}$ is over-period. In this case, we update the information about when bus k could be employed in the next period $g+1$, i.e., add $t^{arl}_{g,|T|,k}$ to the initialization of the next period. Fig. 3 illustrates the process of over-period compensation: the route of bus k in period 1 ends before H ($t^{arl}_{1,|T|,k} < H$), so period 1 is not over-period. However, since the route duration in period 2 exceeds $2H$ (i.e., $t^{arl}_{2,|T|,k} > 2H$), the over-period time $t^{arl}_{2,|T|,k} - 2H$ needs to be added to the beginning of period 3 for bus k. The specifics of over-period compensation will be described in Section 5.3.2.

4.3. Cost formulation

4.3.1. The operation cost

The operation cost is associated with the running mileage and holding time of buses, which can be calculated as follows:

$$C_g^f = \sum_{k=1}^{|K|}\sum_{t=1}^{|T|}\left[\delta^f \cdot DIS\left(i_{g,t,k}, i_{g,t+1,k}\right) + \delta^{\mathcal{H}} \cdot a^{\mathcal{H}}_{g,t,k}\right] \tag{1}$$

where $a^{\mathcal{H}}_{g,t,k}$ means the holding time of bus k at time interval t of period g.

4.3.2. The user cost

The user cost consists of the late arrival as time windows penalty cost and rejection penalty cost. If the vehicle arrives later than the latest departure times, the penalty cost is proportional to the length of the deviation time. In addition, since the service provider is allowed to actively reject immediate requests, extra penalty costs will occur for each rejected request. Therefore, the user cost can be calculated as follows:

$$C_g^u = (1 - y_n^b) \cdot \delta^l \sum_{n \in N_g} q_n \cdot \max\left\{t_n^p - t_n^l, 0\right\} + y_n^b \cdot \delta^b \sum_{n \in \ddot{N}_g} q_n \tag{2}$$

where $y_n^b = 0$ represents request n is accepted; otherwise, request n is rejected to be served.

4.4. Objectives

4.4.1. First objective: minimizing the system cost

The first optimization objective of our model is to minimize the total system cost of each period.

$$\min(C_g^f + C_g^u), \forall g \in G \tag{3}$$

4.4.2. Second objective: optimizing the fairness of request waiting time

Mitigating the unfairness of waiting time for immediate requests is essential for FBS with hybrid requests. Concerning the request rejection mechanism, for the rejected immediate request $n \in \ddot{N}_g$ with $y_n^b = 1$, if any, its rejected time is the beginning of the period g, i.e., $t_n^b = (g-1) \cdot H$. Under operation without demand prediction, request n was submitted before period g and incorporated into the planning of period g, i.e., $t_n^e < t_n^b$. In this case, the waiting time for request n is $t_n^b - t_n^e$. However, under the operation with demand prediction, request n is predicted to be submitted within period g, i.e., $t_n^e > t_n^b = (g-1) \cdot H$, which means in actual operation it will be rejected right after its submission, so its corresponding waiting time is 0. Thus, the waiting time of W_n is calculated as follows:

$$W_n = \begin{cases} t_n^p - t_n^e, & y_n^b = 0 \\ \max\{(t_n^b - t_n^e), 0\}, & y_n^b = 1 \end{cases} \tag{4}$$

When the fairness of request waiting time is not considered, it is likely to induce long waiting times for some requests but short waiting times for others due to the service provider's pursuit of lower costs, which can in turn result in passenger dissatisfaction. From the perspective of the users, the waiting time for each request is expected to be as equal as possible to avoid unfairness. Essentially, a greater variance in waiting time indicates worse fairness for immediate requests. Therefore,

we introduce a fairness indicator $\zeta_g = \sum_{n \in N_g} |W_n - \overline{W_g}| / |N_g|$ to measure the degree of fairness in the waiting time for all requests. A lower value of ζ_g implies a better degree of fairness of period g. As a result, the second objective function considering the fairness indicator is formulated as follows:

$$\min \zeta_g = \sum_{n \in N_g} |W_n - \overline{W_g}| / |N_g|, \forall\, g \in G \tag{5}$$

4.5. Constraints

4.5.1. Vehicle capacity constraints

The vehicle capacity constraints must be satisfied when bus $k \in K$ visits each stop, as follows:

$$z_{g,t,k} \le Z, \forall t \in T, \forall\, g \in G, \forall k \in K \tag{6}$$

4.5.2. The movement constraints

For each bus k, it must complete serving all requests within the time horizon ($|G| \cdot H$). Moreover, those buses in operation need to return to one of the depots before the end of the time horizon, as follows:

$$\sum_{g=1}^{|G|} \sum_{t=1}^{|T|} \frac{DIS(i_{g,t,k}, i_{g,t+1,k})}{vel} + \sum_{g=1}^{|G|} \sum_{t=1}^{|T|} a_{g,t,k}^{\mathcal{H}} \le |G| \cdot H, \forall k \in K \tag{7}$$

$$i_{|G|,|T|,k} = j, \forall k \in K, \exists j \in J \tag{8}$$

where $\sum_{g=1}^{|G|} \sum_{t=1}^{|T|} a_{g,t,k}^{\mathcal{H}}$ is the total holding time of the bus within the planning horizon; $i_{|G|,|T|,k}$ denotes the location of bus k when the time horizon is over.

4.5.3. Tardiness coefficient constraints

If each bus $k \in K$ is allowed to be shared by as many passengers as possible, the vehicle utilization can be improved at the expense of longer detour time, which degrades passenger experience. For this reason, we introduce the tardiness coefficient constraint such that the travel time of each request must be no greater than α times the shortest time of that request, that is,

$$t_n^d - t_n^p \le \alpha \cdot \frac{DIS(i_n^p, i_n^d)}{vel}, \forall n \in N_g, \forall g \in G \tag{9}$$

5. MARL-based multi-objective optimization

This section solves the above multi-objective optimization problem using the multi-agent reinforcement learning (MARL) framework, where each vehicle is considered an agent. The MARL model can learn the correlation between the behavior of agents and the reward according to the environment, to maximize rewards. The application of MARL needs to define the Markov Decisions Process (MDP) with the core elements, such as the state, action, and reward, as well as the Bellman Equation for optimization.

5.1. State

In MARL, the state is a representation of the environment at a specific time interval, providing the relevant information required for every agent to make decisions and take action. In this paper, we design two state representations, which are global state and local state, respectively. The global state records all information during each interaction between agents and the environment, which is necessary for state transition and calculating rewards. On the other hand, the local state leverages

valuable time-related information, which helps each agent to make valuable decisions in the exploration based on their perspectives.

5.1.1. Global state

The global state is introduced to characterize all the agents and the environment, which is the key variable for making a decision and obtaining the reward generated during state transition. The global state $\boldsymbol{S}$ can be represented by (1) the vehicle state $\boldsymbol{S}^V$ and (2) the request state $\boldsymbol{S}^N$, i.e., $\boldsymbol{S} = \{\boldsymbol{S}^K, \boldsymbol{S}^N\}$. Let $\boldsymbol{S}_{g,t}$ denote the global state at time interval t of period g. Similarly, we introduce $\boldsymbol{S}^K_{g,t}$ and $\boldsymbol{S}^N_{g,t}$ to represent vehicle state and requests state at time interval t of period p respectively. Since our model is based on a MARL framework, we consider each vehicle as an agent, and the vehicle state $S^K_{g,t}$ includes the state information of each agent, i.e., $\boldsymbol{S}^K_{g,t} = \{s^K_{g,t,1}, s^K_{g,t,2}, \dots, s^K_{g,t,|K|}\}$. For a given agent $k \in K$, its corresponding vehicle state $S^K_{g,t,k}$ is defined as follows:

$$s^K_{g,t,k}: \{i_{g,t,k}, z_{g,t,k}, t^{arl}_{g,t,k}\}, \forall g \in G, \forall t \in T, \forall k \in K \tag{10}$$

where $t^{arl}_{g,t,k}$ denotes the arrival timestamp of bus k after it finishes the decision made at interval t of period g. Whether bus k can be employed in the scheduling decision process is identified by the relationship between the value of $t^{arl}_{g,t,k}$ and the current time, i.e., $(g-1)\cdot H + (t-1)\cdot H/|T|$, where $H/|T|$ is the length of each time step. Specifically, if $t^{arl}_{g,t,k} \le (g-1)\cdot H + (t-1)\cdot H/|T|$, bus k can be employed; otherwise, it needs to skip the current decision process.

Similarly, the request state $\boldsymbol{S}^N_{g,t}$ of all requests at time interval t of period g can be expressed as $[s^N_{g,t,1}, s^N_{g,t,2}, \dots, s^N_{g,t,|N|}]$, and state $s^N_{g,t,n}$ of a particular request n is defined as follows:

$$s^N_{g,t,n}: \{v_n, t^p_n, t^d_n\}, \forall g \in G, \forall t \in T, \forall n \in N_g \tag{11}$$

where v_n is the state variable to represent the index of the bus assigned for request n at the current time; t^p_n and t^d_n denote the state variable to represent the pick-up time and drop-off time of request n at the current time.

5.1.2. Local state with time-related information

Although the global state contains comprehensive information, it fails to capture the time-related information about the spatio-temporal relationships between passengers and vehicles, limiting the agents' ability to directly utilize this crucial information. For instance, in the initial stage of training when the value of each decision is unknown, requests with different departure time windows are treated equally, and real-time distances between vehicles and requests are ignored. Therefore, in order to better utilize this time-related information, we design a novel representation of the local state for each separated agent. This representation not only provides individual insights for each agent to make decisions but also allows for an improved exploration strategy (which will be described in Section 6.2). Specifically, let $o_{g,t,k}$ denote the local state of agent k at time interval t of period g, as defined below:

$$o_{g,t,k} = \left\{ i_{g,t,k}, z_{g,t,k}, t^{arl}_{g,t,k}, \overbrace{\xi^p_{g,t,1}, \xi^d_{g,t,1}, \cdots, \xi^p_{g,t,n}, \xi^d_{g,t,n}, \cdots}^{2|N_g|} \right\}, \forall g \in G, \forall t \in T, \forall k \in K \tag{12}$$

$$\xi^p_{g,t,n} = \begin{cases} t^l_n - \left[t^{arl}_{g,t,k} + DIS\left(i_{g,t,k}, i^p_n\right)/vel\right], & v_n = \emptyset \\ \mathcal{M}, & v_n \neq k \cap v_n \neq \emptyset \end{cases} \tag{13}$$

$$\xi^d_{g,t,n} = \begin{cases} t^p_n + \alpha \cdot \left[DIS\left(i^p_n, i^d_n\right)/vel\right] - \left[t^{arl}_{g,t,k} + DIS\left(i_{g,t,k}, i^d_n\right)/vel\right], & v_n = k \cap t^d_n = \emptyset \\ \mathcal{M}, & v_n \neq k \cup t^d_n \neq \emptyset \end{cases} \tag{14}$$

where the definitions of $i_{g,t,k}$, $z_{g,t,k}$, $t^{arl}_{g,t,k}$ are the same as Eq.(10); ξ^p_n and ξ^d_n denote the time-related information of request

$n, \forall n \in N_g$ observed by agent k, indicating how urgent it needs to be picked up or delivered, from the perspective of bus k. In Eq.(13), $v_n = \emptyset$ means request n is still not assigned to any vehicle, in which case $\xi^p_{g,t,n} = t^l_n - \left[t^{arl}_{g,t,k} + DIS\left(i_{g,t,k}, i^p_n\right)/vel\right]$ is the remaining time from the current location of k to the pick-up stop of request n before any time windows penalty cost occurs, indicating the urgency for bus k to pick up request n; In Eq.(14), $v_n = k \cap t^d_n = \emptyset$ means bus k has picked up request n but not delivered it yet, in this case, $\xi^d_{g,t,n} = t^p_n + \alpha \cdot \left[DIS\left(i^p_n, i^d_n\right)/vel\right] - \left[t^{arl}_{g,t,k} + DIS\left(i_{g,t,k}, i^d_n\right)/vel\right]$ is the remaining time from the current location of bus k to the drop-off stop of request n before breaking the tardiness constraint, indicating the urgency of request n to be delivered by bus k. If request n is assigned to other agents (i.e., $v_n \neq k \cap v_n \neq \emptyset$ in Eq. (13)), or served by k but has already dropped off (i.e., $v_n \neq k \cup t^d_n \neq \emptyset$ in Eq. (14)), the values of $\xi^p_{g,t,n}$ and $\xi^d_{g,t,n}$ are set to be a very large number $\mathcal{M}$, which implies that the urgency of picking up or delivering request n is almost zero for bus k because it no longer requires service. In summary, the dimension of $o_{g,t,k}$ is $2|N_g| + 3$.

5.2. Action

5.2.1. Action space

In this section, we further describe how to define the action space and how agents utilize the local state with time-related information to decide and select an action, i.e., given $o_{g,t,k}$, in general, the corresponding action space $A_{g,t,k}$ of agent k, containing every possible action, i.e., $a_{g,t,k} \in A_{g,t,k}$, can be defined as follows:

$$a_{g,t,k} = \left\{a^{\mathcal{M}}_{g,t,k} \in \left\{i^p_n, i^d_n, j\right\}, a^{\mathcal{H}}_{g,t,k} \in \mathbb{R}: n \in N_g, i^p_n \in I, i^d_n \in I, j \in J\right\}, \forall g \in G, \forall t \in T \quad (15)$$

where $a^{\mathcal{M}}_{g,t,k}$ is the movement action indicating the next move of bus k at time interval t of period g, including visiting pick-up/drop-off stop and returning to depot; $a^{\mathcal{H}}_{g,t,k}$ denotes the holding time of bus k before its movement of $a^{\mathcal{M}}_{g,t,k}$. The holding time is only considered when bus k decides to pick up a request, i.e., $a^{\mathcal{M}}_{g,t,k} = i^p_n$. In such case, the corresponding optimal holding time can be calculated as follows:

$$a^{\mathcal{H}^*}_{g,t,k} = \underset{a^{\mathcal{H}}_{g,t,k}}{\operatorname{argmin}}\left\{\delta^{\mathcal{H}} \cdot \max\left\{\left(t^e_n - t^{arl}_{g,t,k} - DIS\left(i_{g,t,k}, i^p_n\right)/vel\,, 0\right)\right\}, \delta^l q_n \cdot \max\left\{\left(t^l_n - t^{arl}_{g,t,k} - DIS\left(i_{g,t,k}, i^p_n\right)/vel\,, 0\right)\right\}\right\} \quad (16)$$

where the optimal holding time ($a^{\mathcal{H}^*}_{g,t,k}$) before picking up any request aims to minimize the user cost of time windows violation. When bus k is going to deliver a request or return to the depot, the holding time is set to 0 since there is no penalty cost associated with it. In summary, the dimension of the whole action space is determined by the request set of each period N_g, which is, $2|N_g|$ for picking up and delivering action, plus $|J|$ for returning to depot action and 1 for $a^{\mathcal{H}}$, i.e., $2|N_g| + |J| + 1$.

Based on the aforementioned definition, the schedule of bus k in period g can be represented by the sequential decisions output by MDP, i.e., the sequential recording of a series of actions of the whole period. For example, if the sequential actions recording of bus k is $\left(\left\{a^{\mathcal{M}} = i^p_1, a^{\mathcal{H}} = 3\right\}, \left\{a^{\mathcal{M}} = i^p_2, a^{\mathcal{H}} = 0\right\}, \left\{a^{\mathcal{M}} = i^d_2, a^{\mathcal{H}} = 0\right\}, \left\{a^{\mathcal{M}} = i^d_1, a^{\mathcal{H}} = 0\right\}, \left\{a^{\mathcal{M}} = j, a^{\mathcal{H}} = 0\right\}\right)$, the dispatching process is that bus k first holds at its origin for 3 minutes (because $a^{\mathcal{H}} = 3$) then heads to the pick-up stop of request 1, and immediately visits the pick-up stop of request 2, drop-off stop of request 2, followed by the drop-off stop of request 1, and finally returns to the depot j.

5.2.2. Pruning of action space

Due to the model assumptions and constraints, not every action in the whole action space is valid. If no modification for action space is adopted, these invalid actions will be included in the output of the neural network since the structure of the devised neural network remains unchanged throughout the training process. The presence of invalid actions not only enlarges the scale of state-action pairs which would demand a longer training period, but also leads to poor solution quality, even failing to find a feasible solution. For this reason, instead of repairing solutions for a feasible one, we develop a weight pruning

method to modify the action space to reduce the search space while ensuring the feasibility of model constraints during the training of the deep neural network. This method could effectively improve training performance. There are four pruning operations applied in the action space of bus k when $o_{g,t,k}$ is given, including pruning subject to visiting orders, tardiness constraints, movement constraints, and capacity constraints, which will be described in Appendix A. After the pruning of action space, we could obtain a modified action space $\hat{A}_{g,t,k}$ for bus k that only contains the valid actions, so-called 'valid action space'.

5.3. State transition

In this section, we will discuss two types of state transition under different conditions, namely, state transition between time intervals and state transition between periods. The former is the state changes caused by the decisions of agents between two adjacent time intervals, while the latter is the changes in the information caused by the transition of two adjacent periods.

5.3.1. State transition between time intervals

When state transition occurs from t to $t+1$ in period g, the state information needs to be updated according to previous actions selected by all the agents. Given $a_{g,t,k}$, the vehicle state of bus k, $s^K_{g,t,k}$ can be updated to $s^K_{g,t+1,k}$: $\{i_{g,t+1,k}, z_{g,t+1,k}, t^{arl}_{g,t+1,k}\}$ via Eqs. (17)-(19). Then, the corresponding updating to get $s^N_{g,t+1,n}$ is determined by $s^K_{g,t+1,k}$ and $a_{g,t,k}$, i.e., if $a^{\mathcal{M}}_{g,t,k} = i^p_n$, update $v_n \leftarrow k$, $t^p_n \leftarrow t^{arl}_{g,t+1,k}$; if $a^{\mathcal{M}}_{g,t,k} = i^d_n$, update $t^d_n \leftarrow t^{arl}_{g,t+1,k}$. In addition, we can update the local state with time-related information $o_{g,t,k} \rightarrow o_{g,t+1,k}$ derived from $s^K_{g,t,k}, \forall k \in K$ and $s^N_{g,t+1,n}, \forall n \in N$.

$$i_{g,t+1,k} = \begin{cases} i^p_n, & a^{\mathcal{M}}_{g,t,k} = i^p_n \\ i^d_n, & a^{\mathcal{M}}_{g,t,k} = i^d_n \\ j, & a^{\mathcal{M}}_{g,t,k} = j \\ i_{g,t,k}, & otherwise \end{cases} \tag{17}$$

$$z_{g,t+1,k} = \begin{cases} z_{g,t,k} - q_n, & a_{g,t,k} = i^p_n \\ z_{g,t,k} + q_n, & a_{g,t,k} = i^d_n \\ z_{g,t,k}, & otherwise \end{cases} \tag{18}$$

$$t^{arl}_{g,t+1,k} = t^{arl}_{g,t,k} + DIS\left(i_{g,t,k}, i_{g,t+1,k}\right)/vel + a^{\mathcal{H}^*}_{g,t,k} \tag{19}$$

5.3.2. State transition between periods

In the rolling horizon framework, the state transition between periods can be described as the state transition from the ending time $t = |T|$ of period g to the start time $t = 1$ of period $g+1$, where $g+1 \leq |G|$. At this phase, the rejection mechanism for the immediate requests and over-period compensation are considered.

When the transition from g to $g+1$ occurs, if there is no over-period request or route, we can simply initialize the state variable via $S^K_{g,|T|}$ and N_{g+1}; otherwise, we need to identify (1) whether there are still unassigned requests of N_g, and (2) whether each bus $k \in K$ could complete its route $r_{g,k}$ before the ending of period g to update $t^{arl}_{g+1,1,k}$. In case (1), for each unassigned request n, if $n \in \ddot{N}_g$, it will be considered whether to be rejected or incorporated into the next period: the system will try to insert request n to the original plan of period $g+1$, and it could be incorporated into the $g+1$ only if the decrease in quality of modified planning scheme is lower than the penalty for rejecting request n, otherwise, it will be rejected, i.e., record $y^b_n = 1$ and $t^b_n = (g-1) \cdot H$; if $n \in \dot{N}_g$, incorporate it into the requests set of the next period, i.e., $N_{g+1} \leftarrow N_{g+1} \cap \{n\}$, and update $S^N_{g+1,1}$ via the updated N_{g+1}. By this means, the maximum waiting time for an immediate request n is determined by the rejection penalty cost, i.e., if within a potential route the equation there exists $\delta^l\left(t^p_n - t^l_n\right)q_n > \delta^b q_n$, it will be rejected by the service provider and unable to be incorporated into the next period, to avoid excessive waiting time,

as a quitting mechanism for immediate requests. When it comes to the last period, i.e., $g = |G|$, if there is any reservation request $n \in \dot{N}_{|G|}$ that is not served, we need to remove the immediate request with the largest user cost from $\dot{N}_{|G|}$ as rejection, and then redo the planning again. This rejection operation for immediate requests would continue until all reservation requests could be served, to ensure their priority.

And in case (2), we could update $s^K_{g+1,1,k}$ via Eqs.(20)-(22), where $i_{g,|T|,k}$ and $z_{g,|T|,k}$ denote the ending location and on-board passenger of period g, respectively; and if $t^{arl}_{g,|T|,k} > (g+1) \cdot H$, which means route $r_{g,k}$ is over-period, then inherit $t^{arl}_{g,|T|,k}$ to the next period; otherwise, initialize $t^{arl}_{g+1,1,k}$ as $(g+1) \cdot H$, the beginning of period $g+1$.

$$i_{g+1,1,k} = i_{g,|T|,k} \tag{20}$$

$$z_{g+1,1,k} = z_{g,|T|,k} \tag{21}$$

$$t^{arl}_{g+1,1,k} = \max\{t^{arl}_{g,|T|,k}, (g+1) \cdot H\} \tag{22}$$

5.4. Reward function design

In the MARL model, the reward can be obtained once a state transition takes place. The reward function is essential to the objective optimization and the convergence of the training results. For the dual objectives of our model, we set two corresponding reward functions, including the cost reward R^C and fairness reward R^ζ, and introduce a weighting factor ρ to calculate the total reward function. Therefore, the reward $R(S_{g,t+1}|S_{g,t})$ after the state transition from $S_{g,t}$ to $S_{g,t+1}$ is formulated as follows:

$$R(S_{g,t+1}|S_{g,t}) = \rho \cdot R^C(S_{g,t+1}|S_{g,t}) + (1-\rho) \cdot R^\zeta(S_{g,t+1}|S_{g,t}) \tag{23}$$

5.4.1. Cost reward R^C

We now apply the system cost analysis to the reinforcement learning framework. The transportation cost during state transition is generated once the position of each vehicle changes, that is,

$$C^f_{g,t,k} = [\delta^f \cdot DIS(i_{g,t,k}, i_{g,t+1,k}) + \delta^{\mathcal{H}} \cdot a^{\mathcal{H}}_{g,t,k}] \tag{24}$$

To calculate the reward of user cost at time interval t of period g, we need to identify the request served at t, which is determined by the movement action selected by each bus. The user cost at time interval t of period g can be calculated as follows:

$$C^u_{g,t} = \delta^b \sum_{n \in N_g} q_n y^b_n + \begin{cases} \delta^l \sum_{n \in N_g} q_n (1 - y^b_n)(t^p_{g,t,n} - t^l_n), & \exists a^{\mathcal{M}}_{g,t,k} = p_n \cap t^p_{g,t,n} > t^l_n \\ 0, & otherwise \end{cases} \tag{25}$$

where the condition $\exists a^{\mathcal{M}}_{g,t,k} = p_n$ guarantees that the late arrival penalty for request n (if any) can only be calculated once in a period since the movement of p_n can only be served by one vehicle in our model.

As a result, the cost reward $R^C_{p,t}$ at time interval t of period g can be calculated as follows:

$$R^C_{g,t} = -\left(\sum_{t \in T} \sum_{k \in K} C^f_{g,t,k} + \sum_{t \in T} C^u_{g,t}\right) \tag{26}$$

5.4.2. Fairness reward R^ζ

In this study, the fairness reward R^ζ is obtained by recursion of the fairness indicator ζ_g, which is determined by the requests waiting time throughout the time intervals of a period. According to Eq. (4), the waiting time of request n by time interval t of period g can be calculated as follows:

$$W_{g,t,n} = \begin{cases} t_n^p - t_n^e, & y_n^b = 0 \cap t_n^p \neq \emptyset \\ \max\{(g \cdot t \cdot H/|T| - t_n^e), 0\}, & y_n^b = 0 \cap t_n^p = \emptyset \\ \max\{(t_n^b - t_n^e), 0\}, & y_n^b = 1 \end{cases} \tag{27}$$

Due to the division of time intervals and the response times of different intermediate requests distributed over different time intervals, we need to calculate the average waiting time of immediate requests $\bar{W}_{g,t}$, as well as the fairness indicator $\zeta_{g,t}$ at each time interval t. As a result, the fairness indicator $\zeta_{g,t}$ at time intervals t of period g can be obtained as follows:

$$\bar{W}_{g,t} = \frac{\sum_{n \in N_g} W_{g,t,n}}{|N_g|} \tag{28}$$

$$\zeta_{g,t} = \frac{\sum_{n \in N_g} |W_{g,t,n} - \bar{W}_{g,t}|}{|N_g|} \tag{29}$$

As shown in Eq.(29), a smaller value of $\zeta_{g,t}$ implies the less variance in the distribution of request waiting time in period g, i.e., the better degree of fairness for passengers. Evidently, we expect to obtain the smallest value of $\zeta_{g,|T|}$ at the end of period p, i.e., the best degree of fairness for all the immediate requests of period g. However, a sparse reward issue can take place in the training process if the fairness reward is simply calculated at time interval $|T|$. For this reason, we devise a recursively computed fairness reward function for each time interval, which is expressed as follows:

$$R_{g,t}^{\zeta} = -(\zeta_{g,t} - \zeta_{g,t-1}) \cdot \beta \tag{30}$$

We adopt the method of objective weighting to address the multi-objective optimization problem, which scalarizes the vector of objectives into a single objective. Note that there might be a large gap between the value of $R_{g,t}^{C}$ and $R_{g,t}^{\zeta}$ due to different attributes. To ensure their calculation results have the same order of magnitude and facilitate a comprehensive weighted cost reward, R^{ζ} requires to be multiplied by a factor β. For the optimization of the second objective, if the value of $\zeta_{g,t}$ is smaller than the value of $\zeta_{g,t-1}$, a positive reward is given, which indicates the trend of improving fairness; otherwise, a negative reward is given to punish the trend of impairing fairness. In this way, we can optimize the fairness by the long-term cumulative reward in period g.

5.4.3. Optimization objectives in RL

Bellman Equation is the key to optimizing the MDP, which includes the definitions of policy function π and state-action value function Q, called Q function. The policy function π indicates which action should be selected when a certain state is given, i.e., $A = \pi_{\theta_g}(S_{g,1})$ (In deep reinforcement learning, the policy function is usually characterized by a deep neural network with the parameter θ_g). Since we divide the time horizon into $|G|$ periods and separately optimize the MDP constructed for each period, we need to set $|G|$ deep neural networks for optimization, and the parameter of each period $g \in G$ is θ_g. The Q function is the value estimation of the state-action pair, which measures the expected cumulative return according to the policy function. Specifically, when $S_{g,t}$ and π_{θ_g} are given, the value of the Q function is the cumulative reward obtained after performing $A_{g,t}$ according to policy π_{θ_g}, that is:

$$Q(S, A) = \mathbb{E}_{\pi_{\theta_g}}\{R_{g,t}|S = S_{g,t}, A = A_{g,t}\} = \mathbb{E}_{\pi_{\theta_g}}\left\{\sum_{t_1=1}^{t+t_1=|T|} \gamma^{t_1} R_{g,t+t_1}|S = S_{g,t}, A = A_{g,t}\right\} \tag{31}$$

where γ is the discount factor. In general, the optimization of MDP can be described as: for $S_{g,t}$ at any time interval $t \in T$, find a policy that could maximize the Q function values for the rest of the state-action pairs. However, in our model, we need

to optimize the planning scheme of the whole period $g \in G$, so we need to find an optimal policy $\pi^*_{\theta_p}$ to maximize the value Q function from $t = 1$ to $t = |T|$, that is:

$$\pi^*_{\theta_p} = \underset{\theta_g}{\operatorname{argmax}}\, Q\left(S = S_{g,1}, A = \pi_{\theta_g}(S_{g,1})\right) \tag{32}$$

6. New MARL architecture

Due to the spatial and temporal randomness of immediate requests, the planning time is crucial for solving the dynamic scheduling problem. To speed up the planning process, we introduce a MARL framework based on centralized training and a decentralized execution approach (CTDE), regarding each vehicle as an agent of the RL environment, which enables simultaneous training and planning of multiple vehicles to accelerate the training process. However, the solution quality has been a concern in terms of optimality in many RL approaches. To further improve the solution quality, as shown in Fig. 4, we propose an integrated architecture integrating the MARL framework with the local search, which entails three salient features: 1) incorporating an effective demand prediction approach into MARL training, enhancing foresight in the planning of each period; 2) improving the classic exploration strategy in traditional RL approaches by leveraging time-related information in local state representation; 3) integrating the MARL framework with the local search strategy using imitation learning, where the concept of 'entropy' is proposed to guide the applications of three designed operators.

For each period, specifically, we initialize the RL environment based on the scenario whether demand prediction is applied or not. Once the environment is initialized, the MARL framework uses the improved ϵ-greedy strategy to guide every agent to explore and interact with the environment, when the pruning method of action space (Section 5.2.2) is applied to reduce the space size and ensure each action is valid. The sequential actions, as well as other information, are recorded and stored in a RL buffer. As mentioned in Section 5.2.1, the FBS schedule solution can be formed by sequential actions. Therefore, during RL training we could sample the sequential actions of the best solution to form a RL-based solution as the input to the VND algorithm (will be described in Section 6.3) and yield an improved solution. This solution needs to be converted to the sequential transition as a form of RL and stored in a local search buffer. Then from RL and local search buffers, we can sample the experience for training, using the integrated loss function designed in Section 6.4. Once the training is over, we can output the best solution and perform prediction error, if a prediction model is applied. Finally, the output solution as the initial solution is enhanced once again by the local search method, before moving into the next period.

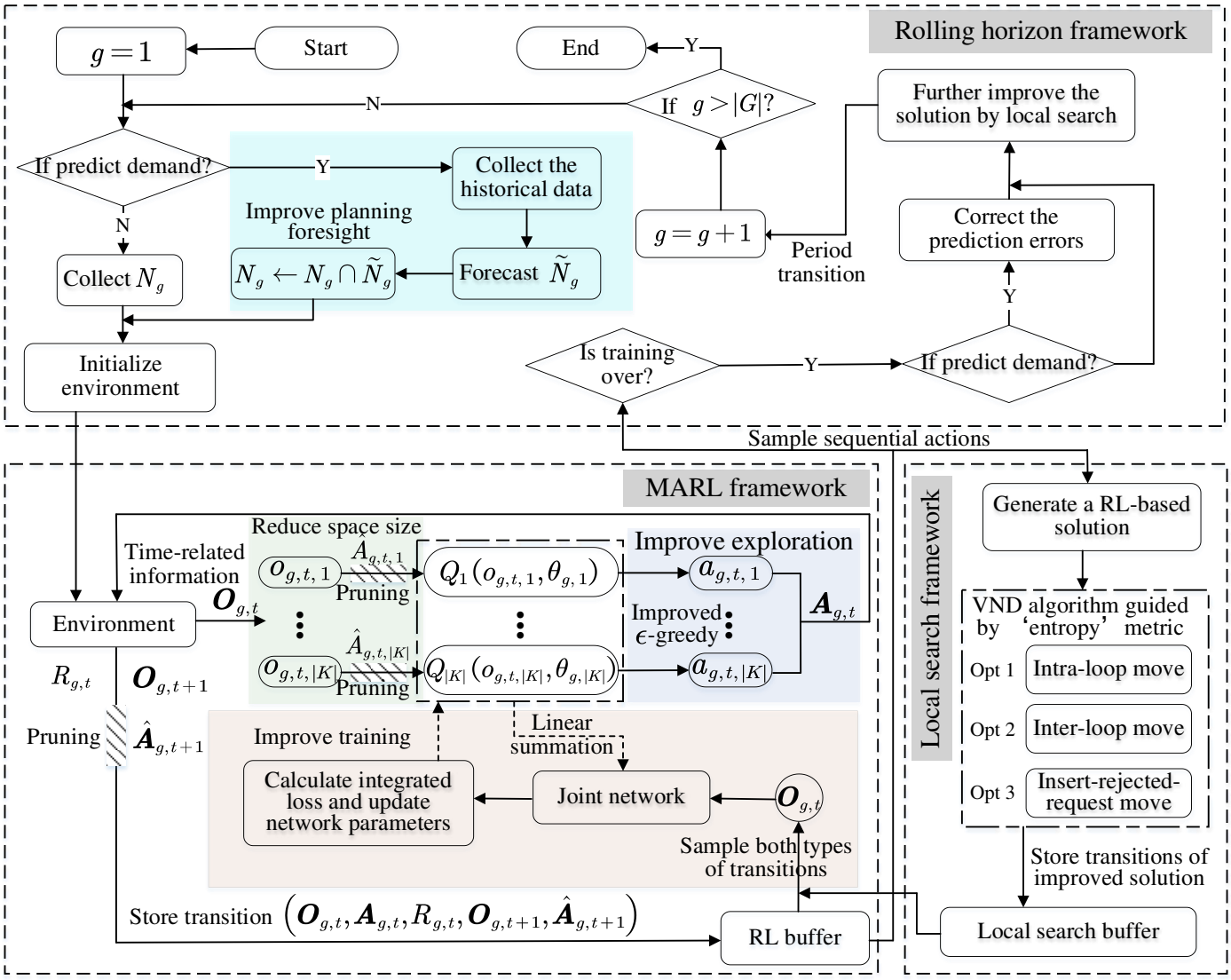

Fig. 4 The architecture of the integrated algorithm

6.1. Rolling VDN algorithm incorporating demand prediction

The value decomposition networks (VDN) algorithm (Peter et al., 2018) is one of the most prevalent algorithms to solve MARL problems, which is based on the CTDE approach. This algorithm regards 'team reward' in a fully centralized framework, characterizing team reward with a joint Q function Q_{tot}, however, allows each agent $k \in K$ to make decisions with its local Q function Q_k which could be decomposed from Q_{tot} by linear summation. In this paper, each Q function is characterized by a neural network. Before the training of period g, for each agent $k \in K$ we initialize a local Q function Q_k characterized by $\theta_{g,k}$. To save the computational resource, the parameter $\theta_{g,k}$ is shared by every agent $k \in K$. Given that we use the local state with time-related information to represent the state information, let the joint state be $\boldsymbol{O} = \left(o_1, o_2, \cdots, o_{|K|}\right)$ and joint action be $\boldsymbol{A} = \left(a_1, a_2, \cdots, a_{|K|}\right)$, the joint network is available by the linear summation of every Q_k, i.e., $Q_{tot}(\boldsymbol{O},\boldsymbol{A}) = \sum_{k=1}^{|K|} Q_k(o_k, a_k)$. Therefore, the input vector of Q_k of period g can be characterized by $o_{g,t,k}$ with a dimension of $2|N_g| + 3$ (described in Section 5.1.2). The output of the Q_k is characterized by action space. Since the holding time of each decision is determined by Eq. (16), the output of the network is only determined by the movement actions, which is $a_{g,t,k}^{\mathcal{M}} \in \left\{i_n^p, i_n^d, j\right\}$ with dimension of $2|I| + |J|$ (described in Section 5.2.1).

Therefore, we can get the Q_{tot} for training and Q_k for each agent to make decisions, which is key to the CTDE approach,

and our objective function for each period can be transferred as follows:

$$\pi_g^* = \underset{\theta_g}{\operatorname{argmax}}\, Q_{tot}\left(\boldsymbol{O}_{g,1}, \pi_g\left(\boldsymbol{O}_{g,1}\right)|\theta_g\right) = \underset{\theta_{g,k}}{\operatorname{argmax}} \sum_{k=1}^{|K|} Q_k\left(o_{g,0,k}, \pi_{g,k}\left(o_{g,0}\right)|\theta_{g,k}\right) \tag{33}$$

Since the rolling horizon scheme and demand prediction are introduced, period transitions and predicted demand should be incorporated into the optimization for each period. As a result, Eq.(33) can be further translated into:

$$\pi_g^* = \begin{cases} \underset{\theta_{g,k}}{\operatorname{argmax}} \sum_{k=1}^{|K|} Q_k\left(o_{g,1,k}, \pi_{g,k}\left(o_{g,1}\right)|N_g \cap \widetilde{N}_g, \theta_{g,k}\right), & g = 1 \\ \underset{\theta_{g,k}}{\operatorname{argmax}} \sum_{k=1}^{|K|} Q_k\left(o_{g,1,k}, \pi_{g,k}\left(o_{g,1}\right)|o_{g-1,|T|,k}, N_g \cap \widetilde{N}_g, \theta_{g,k}\right), & g > 1 \end{cases} \tag{34}$$

Concerning the experience replay mechanism, in each interaction of the agent with the environment, we record a transition tuple $e = \left(\boldsymbol{O}_{g,t}, \boldsymbol{A}_{g,t}, R_{g,t}, \boldsymbol{O}_{g,t+1}, \widehat{\boldsymbol{A}}_{g,t+1}\right)$ as the experience transitions, and store it into a buffer of fixed size, named 'RL buffer'.

6.2. Improved ϵ-greedy strategy in action selection via time-related information

In this section, we proposed a novel strategy that could utilize the time-related information to improve the exploration efficiency in the RL training course. In general, RL training contains two main processes: exploitation and exploration. During exploitation, how agent k selects an action is based on the policy π_{θ_g} characterized by the parameter of a deep neural network θ_g, which is proper for maximizing the expected return at the current moment because π_{θ_g} could return the $a_{g,t,k}$ with the largest reward while $o_{g,t,k}$ is given. However, it could only exploit the current experience samples, which might be insufficient to search out the global optimum. Exploration involves randomly selecting unknown actions from the state-action space, apart from the known state-action distribution. By exploring the entire state-action space we can gather as many experience samples as possible, increasing the likelihood of finding the global optimum. The goal of exploration is to maximize the expected return over the long term, ensuring that the agent explores different possibilities and avoids getting stuck in local optima.

6.2.1. Limitations of the classic ϵ-greedy strategy

Both exploitation (for maximizing immediate rewards) and exploration (for discovering better long-term outcomes) are crucial for RL training, but agents can only choose one action at a time, creating a trade-off. The ϵ-greedy algorithm addresses this by allowing agents to explore randomly or exploit with the highest reward, based on a probability ϵ ($\epsilon \leq 1$). Exploration rate decay can also be applied, starting with a high ϵ and decreasing it as training progresses. This strategy allows agents to focus more on exploration in the early stages of training and shift their focus towards exploitation later. However, as an important decision-making factor, time-related information has not been considered in the exploration of the classic ϵ-greedy method, whereas ignoring such important information might lead to additional exploration time.

6.2.2. Improved ϵ-greedy strategy

To enhance the efficiency of exploration, we improve the traditional ϵ-greedy strategy by incorporating time-related information. Specifically, we can get time-related information on each request for bus k when $o_{g,t,k}$ is given, i.e., $\overbrace{\xi^d_{g,t,1}, \xi^d_{g,t,1}, \ldots, \xi^p_{g,t,n}, \xi^d_{g,t,n}, \cdots}^{2|N_g|}$, by Eqs.(13)-(14), which indicates the urgency of picking up or delivering each request by bus k, respectively. As mentioned in Section 5.1.2, a smaller value of $\xi^p_{g,t,n}$ or $\xi^d_{g,t,n}$ indicates the greater urgency to pick up or deliver request n. Concerning only the time-related information, for bus k, it should serve the request with the greatest urgency (denoted as n^*) as soon as possible. However, we also need to identify whether serving the request n^* is a valid action, i.e.,

to identify whether $p_{n^*} \in \hat{A}_{g,t,k}$ or $d_{n^*} \in \hat{A}_{g,t,k}$. If serving request n (picking up or delivering) is not a valid action, then bus k needs to randomly select an action from the valid action space $\hat{A}_{g,t,k}$. Thus, let $\Gamma(o_{g,t,k})$ denote the function to obtain an action according to the time-related information of $o_{g,t,k}$, which is defined by Eqs.(35)-(36).

$$n^* = \underset{n}{\operatorname{argmin}}\{\min\{\xi^p_{g,t,n}, \xi^d_{g,t,n}\}\} \tag{35}$$

$$\Gamma(o_{g,t,k}) = \begin{cases} i^p_{n^*}, & \xi^p_{g,t,n^*} \neq \mathcal{M} \cap \xi^p_{g,t,n^*} < \xi^d_{g,t,n^*} \cap p_{n^*} \in \hat{A}_{g,t,k} \\ i^d_{n^*}, & \xi^d_{g,t,n^*} \neq \mathcal{M} \cap \xi^d_{g,t,n^*} < \xi^p_{g,t,n^*} \cap d_{n^*} \in \hat{A}_{g,t,k} \\ Random\ action\ from\ \hat{A}_{g,t,k}, & otherwise \end{cases} \tag{36}$$

Heuristically, incorporating time-related information could facilitate agents to make relatively more reasonable decisions compared to random exploration, especially at the beginning of the training. However, in the long run, relying only on time-related information to make a decision is myopic since it does not help agents foresee how current decisions will affect the future return, which is why random exploration is still necessary. To incorporate both exploration mechanisms, we follow the ϵ-greedy algorithm and set two initialized exploration rates, ϵ_1 and ϵ_2, at the beginning of the training, respectively, where $\epsilon_1 + \epsilon_2 \leq 1$. Regarding the exploration rate decay mechanism, let τ denote the current training step, and then $\epsilon_1(\tau)$ and $\epsilon_2(\tau)$ are the exploration rates of ϵ_1 and ϵ_2 at τ, respectively, which could instruct how bus k selects an action. As a result, the strategy of selecting a movement action is represented as follows:

$$a^{\mathcal{M}}_{g,t,k} = \begin{cases} Random\ action\ from\ \hat{A}_{g,t,k}, & with\ probability\ \epsilon_1(\tau) \\ \Gamma(o_{g,t,k}), & with\ probability\ \epsilon_2(\tau) \\ \underset{a \in \hat{A}_{g,t,k}}{\operatorname{argmax}} Q(o_{g,t,k}, a_{g,t,k}), & with\ probability\ 1 - \epsilon_1(\tau) - \epsilon_2(\tau) \end{cases} \tag{37}$$

where $\underset{a \in \hat{A}_{g,t,k}}{\operatorname{argmax}} Q(o_{g,t,k}, a_{g,t,k})$ denotes the greedy action which could bring the largest expected return at the current moment, as exploitation.

Although the exploration based on the time-related information can greatly contribute to the search for relatively good experience samples at the beginning, this effect will diminish as the training step increases since it could only explore a limited state-action space. For this reason, we make ϵ_2 begin to decay at first while remaining ϵ_1 unchanged. ϵ_1 begins to decay after ϵ_2 decays to zero, assuming that the largest decay steps of ϵ_1 and ϵ_2 are $\tau^{\epsilon_1}_{\max}$ and $\tau^{\epsilon_2}_{\max}$ respectively, where $\tau^{\epsilon_1}_{\max} > \tau^{\epsilon_2}_{\max}$. Let τ denote the current train step, then the corresponding exploration rates of $\epsilon_1(\tau)$ and $\epsilon_2(\tau)$ under decay mechanism can be calculated as follows:

$$\epsilon_2(\tau) = \epsilon_2 \cdot \left(1 - \min\left\{\frac{\tau}{\tau^{\epsilon_2}_{\max}}, 1\right\}\right) \tag{38}$$

$$\epsilon_1(\tau) = \epsilon_1 \cdot \left(1 - \min\left\{\frac{\max\{\tau - \tau^{\epsilon_2}_{\max}, 0\}}{\tau^{\epsilon_1}_{\max}}, 1\right\}\right) \tag{39}$$

6.3. Mechanism of local search heuristics

Local search algorithms have a good exploitation capability to find a better solution. Given this fact, we can enhance the solution quality by applying a local search method to improve the quality of the initial solution generated by RL. The key to the local search method is to design the local search operator to perform neighborhood move operations. The conventional random search operator tends to take up too many computational resources resulting in inefficient search. To improve computational efficiency, we design three neighborhood operators along with the concept of 'entropy' to guide the applications of these operators and measure their potential to improve solution quality.

6.3.1. Service loop

For the dynamic scheduling problem, the route length might be very long, possibly due to the large number of requests that each bus needs to serve in a period. Moreover, a neighborhood move can easily violate the constraints, such that the feasibility check of a neighborhood move is very time-consuming. To reduce the computation complexity and simplify the process of neighborhood move, we introduce a strategy that divides a completed route into several relatively short routes as basic units to perform neighborhood move, called 'service loops' represented by ℓ, which is similar to the sequence introduced in Section 5.2.2.2. Each service loop forms when a bus temporarily finishes its current task without carrying any request before going to the next pick-up stop. To illustrate, a completed route $r = \left(i_1^p, i_2^p, I_2^d, i_1^d, i_3^p, i_4^p, i_3^d, i_5^p, i_4^d, i_5^d\right)$, can be divided into two service loops, that is $\ell_1 = \left(i_1^p, i_2^p, I_2^d, i_1^d\right)$ and $\ell_2 = \left(i_1^d, i_3^p, i_4^p, i_3^d, i_5^p, i_4^d, i_5^d\right)$. Note that we add i_1^d to the beginning of ℓ_2 since each loop should contain the original and destination stops.

6.3.2. Priority lists designed based on entropy metrics

Instead of performing neighborhood move opt on random routes or requests, we propose an efficient policy based on priority rules. To find out performing neighborhood move on which request, loop or route is most likely to increase the cumulated reward obtained by RL, we separately measure and rank their potential in cost saving or fairness improvement, then construct three priority lists corresponding to requests, loops and routes respectively before performing neighborhood move. Specifically, to quantify the potential of individual routes, loops, and requests in improving the solution quality, we introduce the concept of 'entropy', represented by μ_1, μ_2, and μ_3, respectively. A higher entropy value indicates a higher potential to improve the solution quality. The three entropy metrics are formulated as follows:

$$\mu_1(k) = -\frac{\rho \cdot c(r_k) + (1-\rho) \cdot \sum_{n \in N(r_k)} (W_n - \bar{W}) \cdot \beta / |N(r_k)|}{\Delta(r_k)} \tag{40}$$

$$\mu_2(\ell) = -\frac{\rho \cdot c(\ell) + (1-\rho) \cdot \sum_{n \in N(\ell)} (W_n - \bar{W}) \cdot \beta / |N(\ell)|}{\Delta(\eta_v)} \tag{41}$$

$$\mu_3(n) = -\frac{\rho \cdot c(n) + (1-\rho) \cdot (W_n - \bar{W}) \cdot \beta}{\Delta(n)} \tag{42}$$

where $\Delta(r_k)$, $\Delta(\ell)$ and $\Delta(n)$ denote the duration of route r_k, loop ℓ and request n, which can be derived from the route sequence. $N(r_k)$ and $N(\ell)$ are the request sets of r_k and ℓ. $c(r_k)$, $c(\ell)$ and $c(n)$ denote the cost associated with route r_k, loop ℓ and request n, respectively, where $c(r_k)$ and $c(\ell)$ can be easily calculated because no shared trip exists between different routes or loops. However, the calculation of transportation cost caused by a single request might be complicated due to the presence of the shared trip, which is not exclusive to one request, but shared with two or even more requests. In this case, the converted trip distance of request n should be counted as the summation of the average distance of the shared trips and the distance of the exclusive trip. Let $N_\ell^\alpha(x)$ denote the on-board request set by bus visiting stop x in the loop ℓ, then the number of onboard requests by stop x can be calculated recursively by Eq.(43), where $\sum_{n \in N(\ell)} \sum_{i \in \ell, i = i_n^p}^{i=x} 1$ is the recursive number of the picked-up requests by visiting stop x in loop ℓ, and $\sum_{n \in N(\ell)} \sum_{i \in \ell, i^- = i_n^d}^{i=x} 1$ is the recursive number of the drop-off requests by visiting stop x (Note that i^- represents the stop preceding i in ℓ). Since the average distance of a shared trip between two adjacent stops i^- and i is the distance between them divided by $|N_\ell^\alpha(x)|$, the cost of request n in loop ℓ can be calculated by Eq.(44), where $a^{\mathcal{H}}(i)$ denotes the holding time before visiting stop i. Note that, assuming a loop $\ell = \left(i_1^d, i_2^p, i_3^p, i_2^d, i_3^d\right)$, since i_1^d is the ending stop of preceding loop as the beginning stop of loop ℓ, request 1 will not be incorporated in $N(\ell)$.

$$|N_\ell^\alpha(x)| = \sum_{n\in N(\ell)} \sum_{i\in\ell, i=i_n^p}^{i=x} 1 - \sum_{n\in N(\ell)} \sum_{i\in\ell, i^-=i_n^d}^{i=x} 1 \tag{43}$$

$$c(n) = \sum_{i^-\in\ell, i\in\ell, i=i_n^p}^{i=i_n^d} \frac{DIS(i^-,i)\cdot\delta^f + \delta^{\mathcal{H}} a^{\mathcal{H}}(i)}{|N_\ell^\alpha(i)|} + q_n\delta^l \cdot \max\{(t_n^p - t_n^l), 0\} \tag{44}$$

It is worth noting that each entropy metric is designed in line with the reward function defined in this paper, but is divided by a duration time, indicating how much reward increases in unit operation time. In general, due to the side effects, a larger value of entropy indicates a greater likelihood of increasing reward when performing neighborhood moves. Therefore, we first construct **priority list 1** in descending order of $\mu_1(k)$ ($k \in K$). Then we construct a **priority list 2** in descending order of $\mu_2(\ell)$, and finally construct a **priority list 3** in descending order of $\mu_3(n)$.

6.3.3. Local search strategy

As shown in Fig. 5, we define three types of opts of neighborhood move, i.e., intra-loop move, inter-loop move, and insert-rejected-request move. The intra-loop move is for adjusting the visiting sequence inside a loop, while the inter-loop move is for adjusting sequences between different loops. And insert-rejected-request move is for accommodating a rejected request into a possible loop. We have summarized the local search procedures in Appendix B, which could return an improved solution.

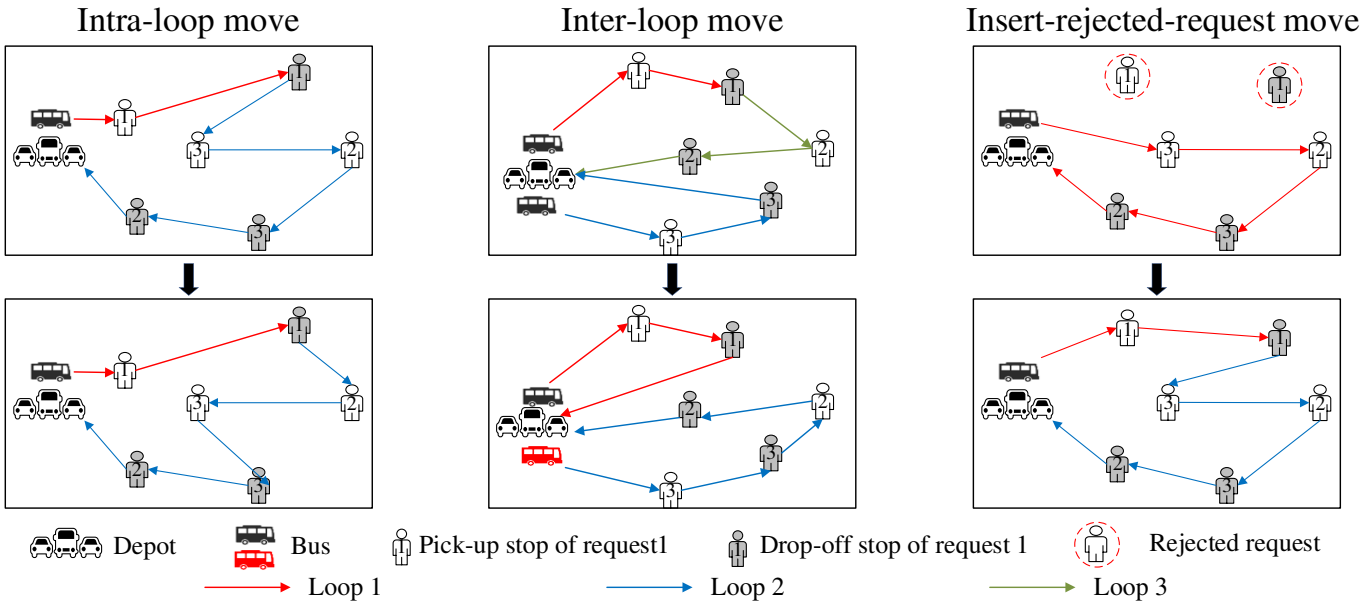

Fig. 5 Illustration of three neighborhood move opts

According to the devised opts, we propose a local search strategy based on Variable Neighborhood Descent (VND) algorithm. Given an initially planned scheme of a period, i.e., the initial solution x_0, as well as the maximum iterations as stop criterion, the VND algorithm can be specified as follows:

VND
Input: x_0: initial solution; τ^{LS}: stop criterion of maximum iterations;
Output: x: output solution.
1: // $\mathcal{N}_n(x)$: the neighborhood of solution x when n opt is applied; $R(x)$: the objective value of solution x
2: $x \leftarrow x_0$;
3: $\tau \leftarrow 1$;

4: Set neighborhood structure $\mathcal{N}_1$ of Inter-loop move, $\mathcal{N}_2$ of Intra-loop move and $\mathcal{N}_3$ of Insert-rejected-request move;
5: **repeat**
6: $n \leftarrow 1$;
7: **repeat**
8: Find the best neighbor x_1 in $\mathcal{N}_n(x)$ with the largest $R(x_1)$;
9: If there exists x_1 and $R(x_1) > R(x)$, set $x \leftarrow x_1$, $n \leftarrow 1$; otherwise, set $n \leftarrow n + 1$;
10: **until** $n > 3$
11: $\tau \leftarrow \tau + 1$;
12: **until** $\tau > \tau^{LS}$
13: **Return** x

where the objective value $R(x)$ can be calculated as follows:

$$R(x) = \rho \cdot \left(C^f + C^u\right) + (1 - \rho) \cdot \zeta \cdot \beta \tag{45}$$

Once an improved solution x is obtained, if any, we need to convert x to an experience similar to that in RL. x contains the planned routes of all buses, which can derive the actions of agents at each time interval, called 'local search guided actions'. Thus, we need to initialize an RL episode for period g and each agent can only select actions based on local-search-guided actions until the end of the episode. Similar to the experience replay mechanism in RL, for each interaction between agents and the environment, a transition $e = \left(\boldsymbol{O}_{g,t}^{LS}, \boldsymbol{A}_{g,t}^{LS}, R_{g,t}^{LS}, \boldsymbol{O}_{g,t+1}^{LS}, \widehat{\boldsymbol{A}}_{g,t}^{LS}\right)_{LS}$ (Note that in local search experience, we need to record $\widehat{\boldsymbol{A}}_{g,t}^{LS}$, rather than $\widehat{\boldsymbol{A}}_{g,t+1}$ in RL experience, for calculating the loss in Section 6.4), as a local search transition, is recorded and stored in a local search buffer of fixed size.

6.4. Loss function design

TD-error is the error in estimating the state value function with experiences in training, which is one of the main loss functions. We could sample a batch of experiences with size ω^{RL} from the RL buffer each time and calculate the MSE loss function of TD-error by Eqs.(46)-(48).

$$e^{RL} = \left(O_{g,t}, A_{g,t}, R_{g,t}, O_{g,t+1}, \hat{A}_{g,t+1}\right) \tag{46}$$

$$TD_{e^{RL}} = R_{g,t} + \gamma \cdot \sum_{k=1}^{|K|} \max_{a \in \hat{A}_{g,t+1}} Q_k\left(o_{g,t+1,k}, a_{g,t+1,k}, \theta_g'\right) - \sum_{k=1}^{|K|} Q_k\left(o_{g,t,k}, a_{g,t,k}, \theta_g\right) \tag{47}$$

$$L^{TD} = \frac{1}{\omega^{RL}} \sum_{e^{RL}=1}^{\omega^{RL}} TD_{e^{RL}}^2 \tag{48}$$

where $O_{g,t}$ contains the local state of all agents, i.e., $O_{g,t} = \left[o_{g,t,1}, o_{g,t,2}, \cdots o_{g,t,|K|}\right]$, as well as $A_{g,t}$, $O_{g,t+1}$ and $\hat{A}_{g,t+1}$; θ_g' is the parameter of the target network when the Double DQN is used, which will be updated via θ_g at regular intervals.

To improve the training performance of the RL algorithm, we utilize local search experiences to supervise the training process to converge to the better solution obtained by the local search algorithm, i.e., the mechanism of imitation learning. A batch of experiences with the size of ω_{LS} are sampled from the local search buffer, which can be used to calculate the imitation learning loss L^{IL} by Eqs.(49)-(51).

$$e^{LS} = \left(\boldsymbol{O}_{\boldsymbol{g,t}}^{\boldsymbol{LS}}, \boldsymbol{A}_{g,t}^{LS}, R_{g,t}^{LS}, \boldsymbol{O}_{g,t+1}^{LS}, \widehat{\boldsymbol{A}}_{g,t}^{LS}\right) \tag{49}$$

$$L^{IL} = \frac{1}{|\omega^{IL}|} \sum_{e^{LS}=1}^{|\omega^{IL}|} \sum_{k=1}^{|K|} \left\{ \max_{a \in \hat{A}_{g,t}} \left[Q_k\left(o_{g,t,k}, a_{g,t,k}, \theta_g\right) + \varsigma\left(a_{g,t,k}^{LS}, a_{g,t,k}\right) \right] - Q_k\left(o_{g,t,k}, a_{g,t,k}^{LS}, \theta_g\right) \right\} \tag{50}$$

$$\varsigma\left(a_{g,t,k}^{LS}, a_{g,t,k}\right) = \begin{cases} 0.8, & \underset{a \in \hat{A}_{g,t}}{\operatorname{argmax}}\left[Q_k\left(o_{g,t,k}, a_{g,t,k}, \theta_g\right)\right] \neq a_{g,t,k}^{LS} \\ 0, & \underset{a \in \hat{A}_{g,t}}{\operatorname{argmax}}\left[Q_k\left(o_{g,t,k}, a_{g,t,k}, \theta_g\right)\right] = a_{g,t,k}^{LS} \end{cases} \tag{51}$$

where $\varsigma\left(a_{g,t,k}^{LS}, a_{g,t,k}\right)$ is the margin function designed to avoid the zero value of the loss function when $\underset{a \in \hat{A}_{g,t,k}}{\operatorname{argmax}}\left[Q_k\left(o_{g,t,k}, a_{g,t,k}, \theta_g\right)\right] = a_{g,t,k}^{LS}$.

In summary, the total loss function is calculated as follows:

$$L = L^{TD} + L^{IL} \tag{52}$$

7. Numerical experiments

This section illustrates the implementation of our proposed algorithm for the numerical test and case study. In the numerical test, the well-known Sioux Falls network is adopted. In the case study, we assess the applicability of our model and algorithm in a real-world 'Shared Bus' project implemented in the Guangzhou Higher Education Mega Center, Guangdong province, China. All experiments are conducted on a PC with Inter (R) Core (TM) i9-10900 CPU @ 2.80GHz core processor, Nvidia 2080 GPU, 32GB RAM, and the system is Windows 10 system, using Python 3.7.

7.1. Evaluation indicators

To better evaluate the optimization performance of our proposed algorithm on the multi-period problem, we have designed the following evaluation indicators:

(1) Effective average user cost (EAUC): This indicator represents the average cost of serving each passenger. In the presence of rejection mechanisms, it is reasonable to distribute the cost of rejections among the accepted passengers. Therefore, we use EAUC as a cost evaluation indicator instead of the total cost. Additionally, EAUC can provide a reference for the service provider's pricing decisions. EAUC can be calculated as follows:

$$EAUC = \frac{\sum_{g \in G}\left(C_g^f + C_g^u\right)}{\sum_{g \in G} \sum_{n \in N_g} q_n \cdot \left(1 - y_n^b\right)} \tag{53}$$

(2) Weighted average fairness indicator (WAFI): This indicator is one of the objectives of our optimization model. As mentioned in Section 4.4.2, ζ_g represents the degree of fairness in the waiting time for the requests of period g. However, since our model applies a rolling horizon framework and the second objective aims to minimize ζ_g, ζ_g can only reflect the fairness of a particular period rather than the entire planning horizon. To evaluate fairness more comprehensively, we calculate the weighting factor for each period based on the number of passengers with immediate requests within each period, as follows:

$$WAFI = \sum_{g \in G} \frac{\sum_{n \in N_g} q_n \cdot \zeta_g}{\sum_{g \in G} \sum_{n \in N_g} q_n} \tag{54}$$

(3) Average late arrival time (ALAT): This indicator represents the average delay in arrival time for each passenger and is an important indicator of service quality. Note that in some cases, a request may be rejected after its latest expected departure time, which is also considered a late arrival. Therefore, ALAT can be calculated as follows:

$$ALAT = \frac{\sum_{g\in G}\sum_{n\in N_g} q_n \cdot \left[\max\{t_n^p - t_n^l, 0\}\cdot(1-y_n^b) + \max\{t_n^b - t_n^l, 0\}\cdot y_n^b\right]}{\sum_{g\in G}\sum_{n\in N_g} q_n \cdot \left(1-y_n^b\right)} \tag{55}$$

(4) Response rate (RR): This indicator is the ratio between the number of successfully served requests and all requests during operation.

$$RR = \left(1 - \frac{\sum_{g\in G}\sum_{n\in N_g} y_n^b}{\sum_{g\in G}\sum_{n\in N_g} 1}\right)\cdot 100\% \tag{56}$$

7.2. Numerical test

7.2.1. Experimental setup

As depicted in Fig. 6, the Sioux Falls network consists of 24 stops, with stop 0 serving as a depot, and the remaining 23 stops acting as intermediate stops. To better simulate travel demand characterized by urban residents, we introduce a central area (highlighted in orange) within the road network with a higher density of travel demand.

For the single-period problem, we consider a static problem of medium size, and randomly generated 10 instances with 30 requests over one hour. In a static problem instance, the origin and destination of each request are randomly generated among the service stops. However, we do not consider the influence of the 'central area' in the static problem. The expected earliest departure time (t_n^e) for request n is randomly drawn from a uniform distribution within a 1-hour range, i.e., $t_n^e \sim \boldsymbol{U}(0, 60)$, while the latest departure time (t_n^l) is obtained by adding a random value drawn from a uniform distribution of 1 to 15 to the expected earliest departure time. The number of required seats for each request is a random integer within the range of 1 to 3.

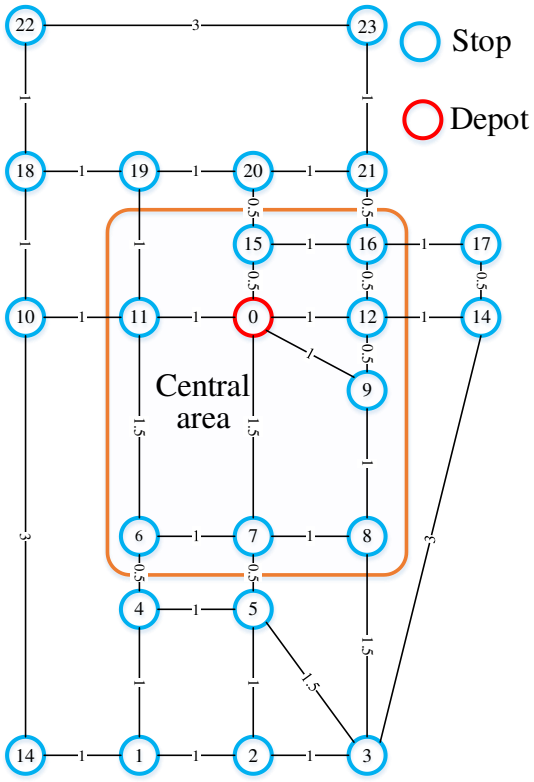

Fig. 6 Sioux Falls Network

In the multi-period problem, we considered a dynamic scenario. As for request generation, we assume that the number

of requests generated from the stop i_1 to i_2 within the planning horizon follows a Poisson distribution with an index λ_{i_1,i_2}, i.e., $|N_g| \sim \boldsymbol{P}(\lambda_{i_1,i_2})$. In this case, if the origin i_1 is in the central area, λ_{i_1,i_2} is set to 0.2; otherwise, λ_{i_1,i_2} is set to 0.16. The type of each generated request (reservation or immediate request) is determined by a random number with a probability of 0.5. The generation of the required number of seats for each request, the expected departure time window for each reservation request, as well as the submission time and the latest departure time for an immediate request, are similar to those in the single-period problem instance. Based on these assumptions, we generated an instance with 86 requests over three hours, and 50% of them were reservation requests.

We assume that the number of time intervals of each period ($|T|$) is 30, the average speed of each vehicle (vel) is 30 km/h, the associated unit transportation and penalty cost coefficients δ^f, δ^H, δ^l and δ^b are 1.2 ¥/km, 0.5 ¥/min, 0.5 ¥/$(min \cdot pax)$ and 10 ¥/pax, respectively, the tardiness coefficient (α) is 2.5, and the weight of the cost objective (ρ) is set to 0.8. The scaled coefficient of two types of rewards (β) is set to 20, which is empirically obtained through the trial-and-error method by running preliminary experiments with different values of β, such that the EAUC and WAFI indicators have the same order of magnitude in multi-objective optimization. In the principle of objective weighting, the value of β could be flexibly tuned through additional tests or sensitivity analysis in other applications. These values serve as the default settings unless specifically indicated otherwise.

Regarding the hyperparameter values, we employ 2-layer fully connected RNN networks as our deep neural network architecture, with each hidden layer consisting of 64 neurons. Additionally, we set the number of training steps ($\tau_{\max}^{RL}$) to 8000, the iterations of local search algorithm ($\tau_{\max}^{LS}$) to 2000, the learning rate to 0.01, the discount factor (γ) to 0.95, the initial exploration rates (ϵ_1 and ϵ_2) to 0.4 and 0.6, respectively, and the corresponding decay steps ($\tau_{\max}^{\epsilon_1}$ and $\tau_{\max}^{\epsilon_2}$) to 4000 and 8000 (as described in Section 5.2.3). For experience replay, we set the size of the RL buffer and local search buffer to 5000 and 2000, respectively, and the mini-batch size for ω^{RL} and ω^{LS} to 32. The interval for updating the target network is set to 100. The RMS optimizer is adopted in our experiments.

7.2.2. Discussion of the single-period problem

The capability to solve the single-period problem is basic to measure one algorithm's performance. In this section, we evaluate our proposed algorithm's performance to solve the single-period problem in terms of assessing training evolution. In addition, we explore the benefits of the novelties introduced in this paper, including the integration with IL and local search method (Section 6.3 and 6.4) and the improved ϵ-greedy strategy (Section 6.2).

7.2.2.1. Assessing DRL training

Training is the cornerstone of MARL because it determines the ultimate solution quality. In this subsection, we evaluate the training process of the proposed MARL algorithm in this paper, which applies the CTDE framework (described in Section 6.1) to train each agent. In comparison, we also introduce a decentralized MARL framework, which individually trains the policy for each agent. Both algorithms are applied to solve 3 randomly generated medium-size instances mentioned in Section 7.2.1. In addition, since bi-objective optimization is one of the main focuses of this paper, it is necessary to assess the effect of different reward weights (ρ) on the algorithm performance. Therefore, we evaluate the performance under different values of ρ, i.e., $\rho = 0$, $\rho = 0.5$ and $\rho = 1$.

To validate the stability of the above algorithms and reduce random errors, we employed 10 different random seeds to initialize the neural network for each instance during the training process and obtained 10 sets of results for each algorithm. We calculated the average value of the results from these 10 experiments as the output value for each instance. This approach

helps to account for the variability introduced by random initialization and provides a more robust assessment of the performance (which will be continuously conducted in the following experiments).

Fig. 7 presents the training evolution of CTDE and the decentralized framework-based MARL algorithm. The graph demonstrates the consistent superiority of our proposed algorithm (CTDE) over the decentralized algorithm in almost all instances except for instance 1 which exhibits a closer performance between both algorithms when ρ is set to 0.5 and 1, indicating a comparable convergence in cumulative reward. Nevertheless, our proposed algorithm exhibits remarkable performance across the rest of the instances and maintains its superiority when various objective weights are incorporated. Regarding training stability, the curves of CTDE of all instances consistently ascend and converge to a stable level after 5000 steps, regardless of the value of ρ. Conversely, the Decentralized algorithm often fails to achieve convergence in a majority of instances (5 out of 9). This observation reveals the robust scalability of our proposed algorithm, which ensures consistent and reliable training performance. In terms of convergence value, measured as the average cumulative reward during the final 500 training steps, a notable gap is observed between CTDE and Decentralized. On average, CTDE achieves convergence values that are 61.23%, 32.24% and 32.1% on average higher when ρ is set to 0, 0.5 and 1, respectively. This signifies the superior convergence capability of our proposed algorithm for searching for an improved solution. Furthermore, an interesting observation arises as the difference between the two curves tends to widen with decreasing ρ values. This finding suggests the effectiveness of our algorithm in addressing fairness objectives and providing optimal solutions in scenarios where fairness considerations are critical.

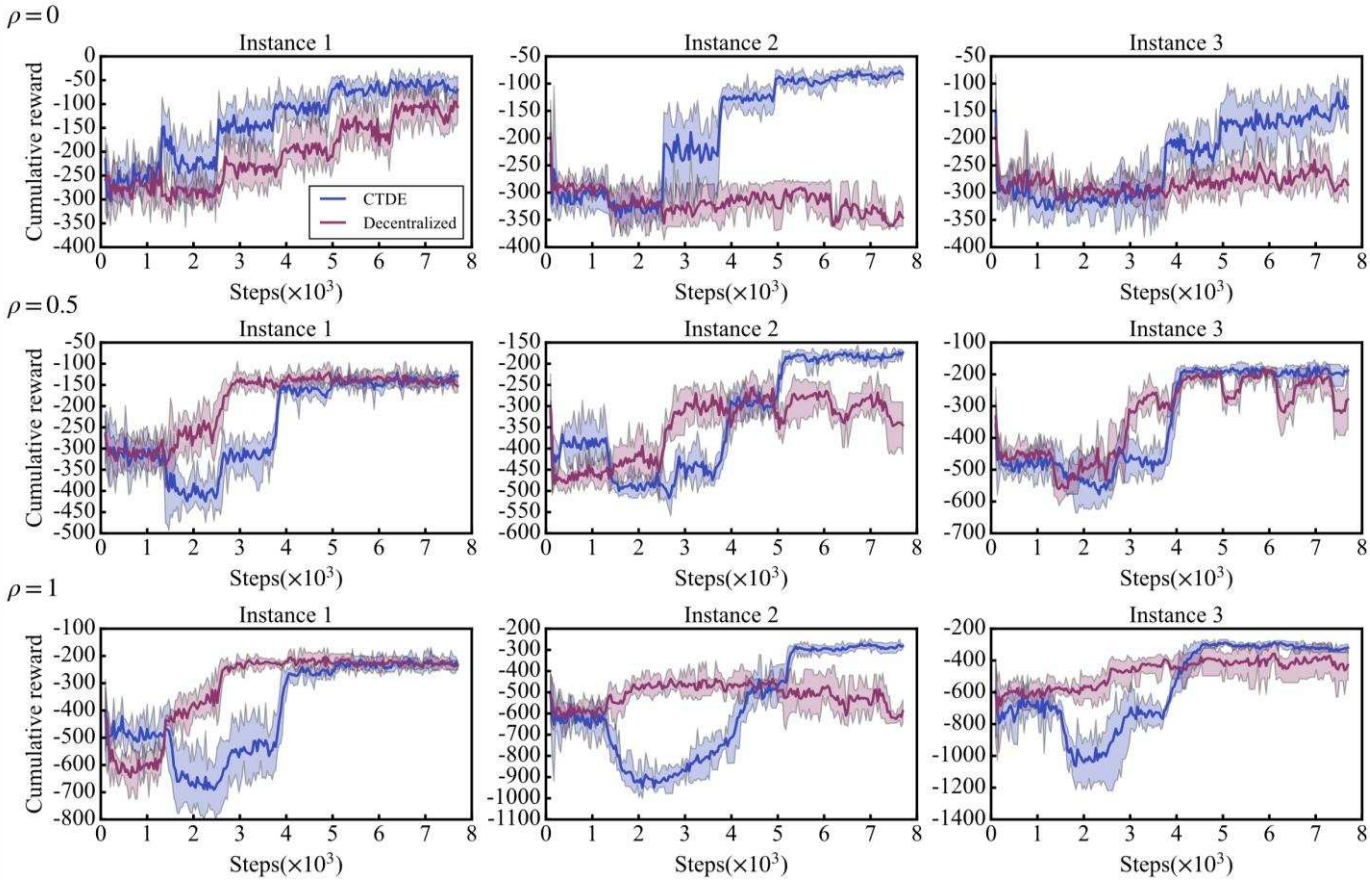

Fig. 7 Evolution of training process (10 random seeds)

Fig. 8 illustrates the training evolution of our proposed algorithm solving instance 1, as measured by four evaluation

indicators described in Section 7.1. All four indicators exhibit a favourable optimization trend as the training progresses. Regarding the EAUC and WAFI indicators, they initially show fluctuations within the first 4000 steps and then rapidly converge to values of 6.268 and 3.485, representing a reduction of 36.66% and 68.32% compared to their initial values respectively. The result shows that our algorithm is effective in solving the dual-objective optimization problem for cost savings and fairness improvement simultaneously. Meanwhile, the evaluation indicators for service quality (the ALAT and RR) improve significantly. Specifically, ALAT decreases from its highest value of approximately 6.059 minutes to 0.142 minutes, while RR increases from its lowest value of approximately 58% to 96%.

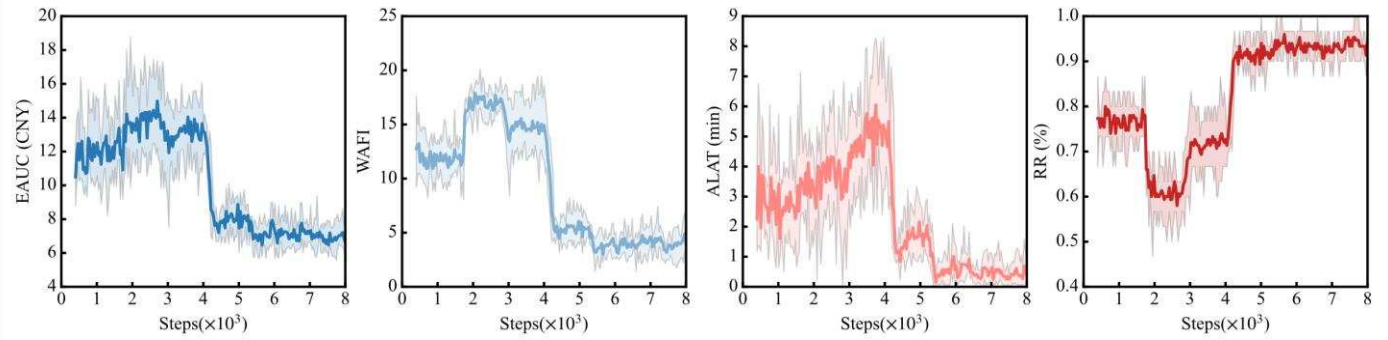

Fig. 8 Evolution of evaluation indicators solving instance 1 (10 random seeds)

7.2.2.2. Algorithmic comparison

To validate the effectiveness of heuristics-guided DRL, we conduct the ablation study by comparison among the following models. All three algorithms are applied to solve 10 randomly medium-size instances following the method mentioned in Section 7.2.1. The instances were generated to have no differences other than random effects.

1. **Heuristics-guided DRL** refers to the DRL combining IL with local search methods, augmented with the improved ϵ-greedy strategy leveraging the time-related information to guide the exploration during DRL training.
2. **DRL** represents the traditional DRL without IL and local search methods.
3. **Local search** refers to the heuristics described in Section 6.3.3.
4. **Heuristics-guided DRL with classic ϵ-greedy strategy** refers to the DRL combining IL with local search methods, but without the improved ϵ-greedy strategy. The classic ϵ-greedy strategy is widely used in most RL studies (Lowe et al., 2017; Mao et al., 2020; Ahamed et al., 2021), such as Q-learning and traditional DRL.

The study compares the heuristics-guided DRL with two benchmarks, namely DRL and local search, on problems of the 10 instances and shows that the method provides objective values that are on average 25.7% better than DRL and 43.4% better than local search, respectively. This result highly demonstrates the added benefit of integrating IL and local search methods with DRL.

We also compare the heuristics-guided DRL to the version with the classic ϵ-greedy strategy. The result shows that heuristics-guided DRL with the improved ϵ-greedy strategy provides the objective value that is on average 52.1% better than that with the classic ϵ-greedy strategy. This demonstrates the effectiveness of introducing the designed improved ϵ-greedy strategy.

7.2.3. Discussion of the multi-period problem

To verify the effectiveness of our proposed algorithm in addressing the multi-period problem as a dynamic scenario, we

first generated 10 instances based on parameters of Poisson distribution in Section 7.2.1 and applied our proposed algorithm to solve them. We focus on examining the benefits of using the operation that incorporates demand prediction methods (referred to as 'with prediction'). As a comparison, we also solve the same instance using the approach that does not incorporate demand prediction (referred to as 'without prediction'). Note that in the numerical tests, due to the lack of historical data, we are unable to perform proper demand prediction. Hence, we assume that the immediate requests in the generated instances as perfect demand predictions with no prediction error. In the multi-period problem experiment, the default value of period length (H) is set to 30 minutes, and the values of the other parameters are set to their default values in Section 7.2.1 unless specifically indicated otherwise.

We analyze the results of three indicators and the average planning time for each period when solving the multi-period problem with and without demand prediction. In this experiment, we used 10 random seeds to initialize the DNN network, and the results are listed in Appendix C. The results demonstrate that the operation with demand prediction consistently outperforms the operation without demand prediction across all 10 seeds. Significant reductions of 58.4%, 73.4%, and 94.6% are observed in EAUC, WAFI, and ALAT, respectively. These highlight the advantages of incorporating demand prediction in terms of cost reduction, passenger fairness improvement, and service quality enhancement when the prediction method achieves sufficient accuracy. The substantial improvement can be attributed to the larger scale effect and the ability to anticipate future demands provided by the demand prediction mechanism. Note that the planning time for each period in the operation with demand prediction is longer compared to the operation without demand prediction (12.88s on average). This is due to the additional computational time required for correcting prediction errors, involving neighborhood moving and feasibility checking, which is performed when each immediate request is submitted (even though in perfect prediction instances neighborhood moving is ignored, feasibility checking is still carried out throughout the entire planning stage). In addition, no significant difference can be observed in RR between the two operations, which implies that the operation with prediction can significantly improve the other three indicators to a greater extent while ensuring a high response rate.

7.2.4. Sensitivity analysis

A series of sensitivity analyses are conducted to examine the model performance under different situations. Three critical factors impacting the model performance are studied: the period length (H), the weight of the cost objective (ρ), and the prediction error ratio (ψ).

7.2.4.1. Sensitivity analysis to the period length H

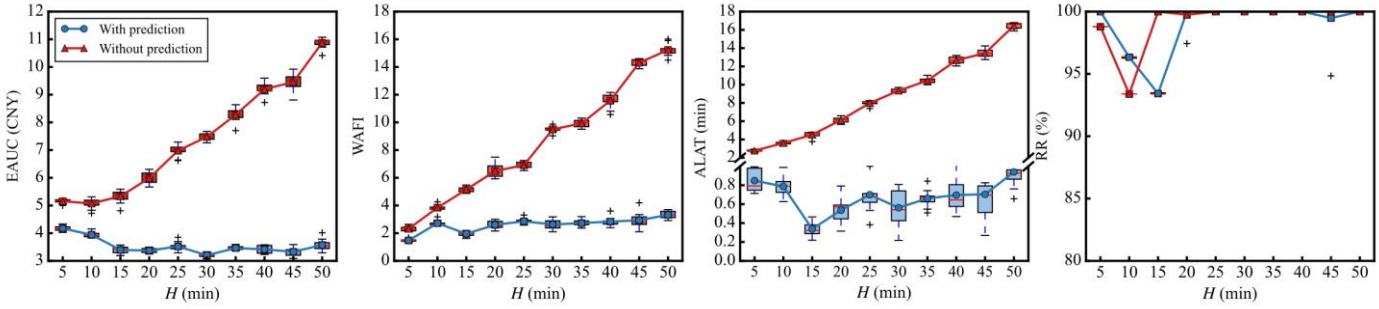

Fig. 9 Sensitivity to the period length

Fig. 9 illustrates the impact of different period lengths for time horizon division on system performance under operations with and without demand prediction. While the differences in RRs are trivial, the gaps of EAUC, WAFI, and ALAT between

operations with and without demand prediction are generally larger as H increases. There is an inflection point for EAUC. As H increases from 5 to 10, EAUC decreases slightly by 1.56%, while increasing by 114.3% as H further increases to 50. It is because a larger H allows more requests to be planned simultaneously at the expense of less flexibility in schedule adjustment, such that the negative impact of reduced frequency of schedule adjustment outweighs the benefit of centralized planning when H exceeds a threshold. On the contrary, EAUC decreases as H grows when demand prediction is considered. There are two reasons for this. First, demand prediction can mitigate the negative impact of the reduced frequency of schedule adjustments. Second, a longer look-ahead length (H) allows for considering a larger amount of system information for the near future. Similarly, WAFI and ALAT increase considerably with a higher value of H, whereas the changes are trivial when demand prediction is considered. This is because the immediate requests cannot be served in time under a low frequency of schedule adjustment.

7.2.4.2. Sensitivity analysis to the objective weight ρ

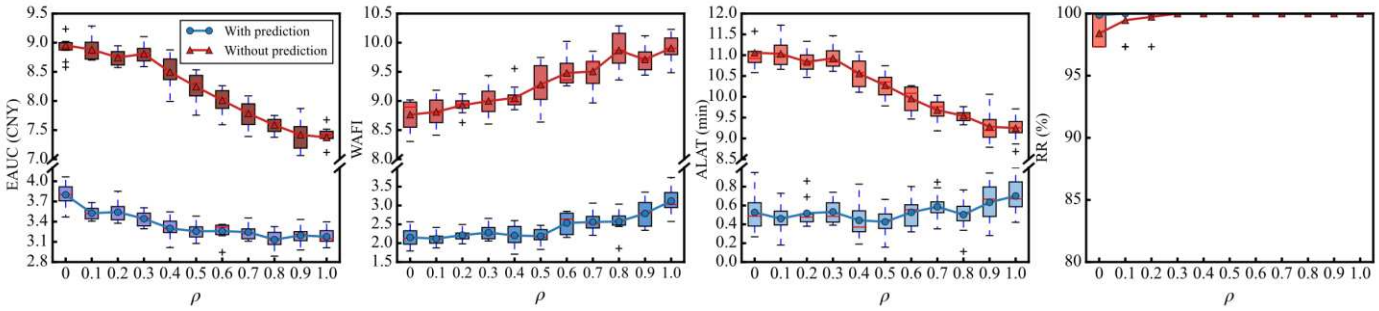

Fig. 10 Sensitivity to the objective weight

Fig. 10 presents the impact of the objective weight ρ on the four indicators. Eq. (29) shows that as ρ increases, the cost objective will be more emphasized. Typically, when $\rho = 0$, only fairness optimization is considered; when $\rho = 1$, cost optimization is emphasized exclusively. The differences in RRs between both operations are trivial. As ρ increases, EAUC decreases while WAFI increases, which indicates the trade-off between costs and fairness. The magnitude of changes in EAUC and WAFI are different. Specifically, when ρ increases from 0 to 1, EAUC reduces exponentially by 17%. WAFI increases slightly by 1.523% as ρ increases from 0 to 0.5, whereas it increases significantly by 42.66% when ρ increases to 1. Based on these results, when the operation with prediction is adopted, it appears that setting ρ to 0.5 could yield a good trade-off between the two objectives, since it can quickly reduce EAUC without significant increase in WAFI. ALAT without prediction reduces considerably as ρ increases, whereas the change is trivial under the operation with prediction. This is mainly because cost optimization contributes to reducing time windows penalty cost, and emphasizing fairness more would deteriorate this indicator. However, introducing the prediction approach could mitigate this negative effect.

7.2.4.3. Sensitivity to the prediction error parameter Ψ

While it is shown that demand prediction improves the system performance to some extent, it is not uncommon for the service provider to encounter the issue of information reliability. The question remains open whether the predictive optimization model can outperform the model without prediction in the presence of prediction errors. To measure the impact of prediction errors on the operation, we modify the generated request set to simulate the scenario with prediction errors. We assume that the predicted number of passengers of immediate requests generated in the numerical test is $\tilde{q}_n$, and the actual number of passengers for each request (q_n) equals $\tilde{q}_n$ plus an random error term $\Delta \sim N(0, \Psi\sigma_0)$, represented by Eq.(57), where

Ψ denotes the demand prediction error parameter and $\tilde{q}_n(\Psi)$ denotes the actual number of passengers of request n with Ψ.

$$q_n(\Psi) = \begin{cases} \tilde{q}_n + \sigma, & \tilde{q}_n + \sigma \geq 0 \\ 0, & \tilde{q}_n + \sigma < 0 \end{cases}, \sigma \sim N(0, \Psi\sigma_0), \forall n \in \ddot{N}_g, g \in G \tag{57}$$

Fig. 11 presents the impact of demand prediction errors on system performance, where the horizontal lines correspond to the indicator values without prediction. To examine the benefit of prediction error correction, the results of prediction failure-aware policy, referred to as 'Prediction + Correction', are benchmarked with its simplified version without prediction error correction. As we can see, the operation incorporating a prediction method (no matter with or without correction mechanism) outperforms by far the operation without prediction at the expense of a slight reduction of RR, which validates the benefits of introducing demand prediction method. Under the operation with prediction, the indicators appear to deteriorate as Ψ increases, especially when Ψ increases from 0 to 0.1. Nevertheless, EAUCs, WAFIs and RRs with prediction error correction are consistently better than those without prediction error correction, and the gaps between them appear to be larger with a higher value of Ψ, particularly for EAUCs and ALATs. This indicates that the prediction error correction mechanism contributes to alleviating the negative impact of prediction errors. The differences in WAFIs are not outstanding, which implies that the impact of prediction errors on fairness is limited.

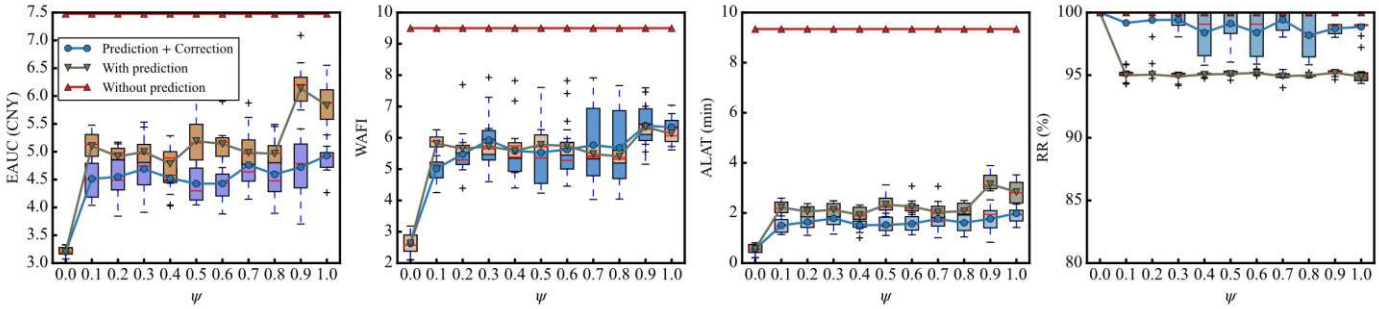

Fig. 11 Sensitivity to the prediction error parameter

7.2.5. Benchmarking with state-of-the-practice methods under different problem sizes

The problem instances, ranging from small to large, include Case 1 with 12 requests and 1 bus, Case 2 with 91 requests and 12 buses, Case 3 with 180 requests and 25 buses, and Case 4 with 302 requests and 30 buses. These instances, generated randomly for generality, span a 90-minute duration divided into three periods under a rolling planning framework, effectively gauging the algorithm's performance across diverse problem scales.

For comparison, we employ the commercial solver Gurobi and the genetic algorithm (GA, Tan et al., 2001). The settings of crossover rate and mutation rate of GA are set to 0.8 and 0.2, respectively. The variable neighborhood search (VNS) method is adopted as another benchmark, which includes three specified neighborhood opts described in Section 6.3.3, and two additional opts of random move for 'intra-loop' and 'inter-loop' are designed to increase search potential. The maximum iterations of GA and VNS are set to 1000. Another benchmark, the traditional DRL without enhancement by IL and Local search is also applied here as a benchmark, which works well in recent pickup and delivery problem studies (Singh et al., 2019; Ahamed et al., 2021; Liu et al., 2022). All algorithms are measured by the objective value calculated by Eq. (45) and four indicators described in Section 7.1, in addition to average computational time and training time per period (for DRL-based algorithms only). The average computational time is the actual execution time for real-time implementation in the rolling planning framework.

Table 2 reports the performance comparison of five algorithms for solving four cases. It can be seen that our proposed algorithm (DRL + IL + Local search) outperforms other benchmarks in terms of objective and computational time in all cases except for case 1, where all algorithms obtain the optimal solution. As the problem size increases, the performance gap between the heuristic and DRL-based methods widens significantly. For instance, in Case 2, our algorithm's objective value is notably lower than that of GA and VNS by 29% and 27.8%, respectively, and this gap expands to 43% in Case 4, underscoring the DRL-based algorithm's advantage for large-scale challenges. In addition, DRL-based algorithms demonstrate superior RR compared to heuristic methods. The RRs of GA in Case 2 and Case 3 are quite low, at 54.7% and 51.5% respectively, which might have a serious impact on actual operation. The RRs of VNS in Case 2 and Case 3 reach nearly 100%, very close to that of our proposed algorithm. However, the gap widens in Case 4 by 13.2%, where GA is unable to obtain a feasible solution. This indicates that the DRL-based algorithms have better performance in solving the dynamic schedule problem in the hybrid-request context. The difference between GA and VNS in RR might be attributed to the introduction of a customized operation VNS, 'Inject-Rejected-Request move' is designed for VNS to reduce the rejected requests.

Furthermore, another advantage of DRL-based algorithms is less computational time. Even in Case 4 with the largest request size, the average computational time of our proposed algorithm is only 5.49s, while that of VNS reaches 540s, which is 100 times greater, and GA is unable to solve Case 4 within 1000s. Furthermore, by comparing the result of 'DRL + IL + Local search' with traditional DRL, it is evident that our proposed integration framework keeps consistent superiority for Cases 2, 3 and 4, with a reduction in the objective value of 8.7%, 8.8% and 14.7% respectively, despite the training time is increased when integrating IL and local search method, which are 3.6%, 3.9%, 7.3% and 6.5% among four cases. In fact, since the service provider can apply a well-trained network in real-world applications, the cost of the increase in training time can be neglected.

In conclusion, our proposed algorithm not only effectively reduces objective value across varying problem sizes but also maintains high computational efficiency and adaptability. Although the training time of the algorithm increases with the size of the problem, its performance in terms of the key indicators highlights its considerable potential and effectiveness in tackling real-world challenges across various settings.

Table 2 Performance comparison of different problem sizes

Case	Algorithm	Case 1	Case 2	Case 3	Case 4
Objective value	Gurobi	124.6	/	/	/
	GA	124.6	857.2	1696	/
	VNS	124.6	843.1	1342	3148
	DRL	124.6	666.7	1112	2105
	DRL + IL+ Local search	124.6	608.6	1014	1795
EAUC (CNY)	Gurobi	4.089	–	–	–
	GA	4.089	10.27	10.41	–
	VNS	4.089	5.703	4.504	7.044
	DRL	4.089	4.326	3.731	4.34
	DRL + IL+ Local search	4.089	3.984	3.422	3.649
WAFI	Gurobi	2.75	–	–	–
	GA	2.75	2.329	2.597	–
	VNS	2.75	5.617	4.27	6.576
	DRL	2.75	4.514	3.889	5.327
	DRL + IL+ Local search	2.75	3.846	3.461	4.363
ALAT (min)	Gurobi	4.321	–	–	–
	GA	4.321	0.7445	0.2662	–
	VNS	4.321	2.35	1.275	4.467
	DRL	4.321	2.109	1.368	2.622
	DRL + IL+ Local search	4.321	1.586	1.004	1.708

RR (%)	Gurobi	100%	–	–	–
	GA	100%	54.72%	51.45%	–
	VNS	100%	98.2%	99.2%	86.7%
	DRL	100%	99.9%	100%	98.5%
	DRL + IL+ Local search	100%	100%	100%	99.9%
Average computational time per period (sec)	Gurobi	0.16	–	–	–
	GA	0.74	30.1	74.3	–
	VNS	0.24	32.4	110	540
	DRL	0.07	0.56	0.73	4.28
	DRL + IL+ Local search	0.16	0.7	1.29	5.49
Average training time per period (sec)	Gurobi	–	–	–	–
	GA	–	–	–	–
	VNS	–	–	–	–
	DRL	5.61	87.8	234	489
	DRL + IL+ Local search	5.8	90.7	251	521

Note: A solution could not be obtained within 1000s or the data is not available is marked as '–'

7.3. Case studies

7.3.1. Real-world application of demand prediction model

In this section, we introduce a real-world application for forecasting demand. Long Short-Term Memory (LSTM) recurrent neural networks have established a strong foothold in the realm of time series forecasting due to their exceptional ability to model sequential data and capture long-term dependencies (Wang et al., 2019). Their architecture is specifically advantageous in learning from and remembering information over extended periods, which is crucial for accurate predictions in dynamic systems like transportation. The specific description of the prediction model is detailed in Appendix D.

7.3.2. Case description

In this section, we report on the applicability of our algorithm via a real-world bus network, Guangzhou Higher Education Mega Center, and compare the performance of our algorithm with the state-of-the-practice and other approaches from the literature. At present, 68 pick-up stops and 100 drop-off stops have been set up in this area for the 'Share Bus' project. Fig. 12 shows the interface of the App and the operation of the flexible bus service, and Fig. 13 shows the distribution of stops in the area. The operation of 'Shared Bus' project is of FBS operation, which means the scheduled routes of buses are not fixed and dynamically changed in response to the upcoming requests. The information on riding requests and vehicle operation is provided by the local bus company. The operation duration is from 8:00 to 21:00, which is quite long for static optimization. To align with our FBS framework, we formulate this case study as a multi-period problem, set the period length H to 30 minutes, and assume the requests of the first period are reservation requests, while the rest are regarded as immediate requests. The other model parameters are set following the default values in Section 7.2.1, except for the fleet size $|K|$ and the objective weight ρ, which is set to 5 and 0.5 in this case.

(a) Interface of App

(b) Vehicle employed

Fig. 12 ‘Share Bus’ application in Guangzhou Higher Education Mega Center

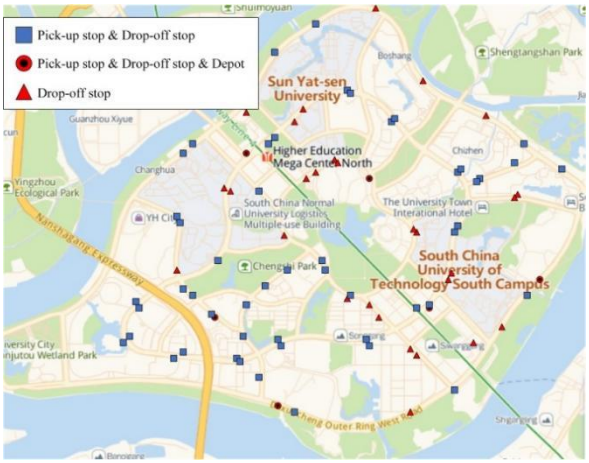

Fig. 13 Spatial distribution of bus stops

As mentioned in Section 7.3.1, the historical data for a total of 21 days from Aug. 3 to Aug. 24, 2021, which is provided by the ‘Shared Bus’ project, is used as the training set to predict the immediate requests from Aug. 25 to Aug. 31, and the historical data for those 7 days is used as the testing set. The average number of passengers in those 7 days is 117. As for the prediction model, we set the prediction length to 15 minutes for predictions. Note that a predicted request is not identical to an actual request, because the predictions could only estimate the number of passengers departing from one stop to another within the prediction interval, without precise departure time windows, while that of an actual request is deterministic. To this end, we assume the predicted demand from one stop to another within a certain prediction interval as one request, whose time window spans the whole prediction interval. In other words, if a request is generated within the prediction interval as predicted, and the number of passengers is correctly forecasted, we call it a successful prediction, otherwise, it is a failed predicted request. Given that prediction errors are inevitable in real-world operations, implementing a correction mechanism for these errors is crucial to maintaining the effectiveness of our proposed algorithm. By this means, we need to recalculate the relevant indicators after each correction. This correction would not lead to a serious impact, as demonstrated in Section 7.2.4.3, our proposed algorithm has shown better performance than that without a demand prediction model, even with a significant prediction error parameter. However, evaluating the performance of the prediction model is essential for further analysis and operational improvements. To this end, we employ two classic evaluation metrics, mean squared error (MSE) and mean absolute error (MAE), commonly used in prediction studies, and introduce failed prediction ratio (FPR) as an additional measure of the accuracy of our model. These three metrics are calculated as follows:

$$
\begin{aligned}
MSE &= \frac{1}{\sum_{g\in G}\sum_{n\in\tilde{N}_g} 1}\sum_{g\in G}\sum_{n\in\tilde{N}_g}(q_n-\tilde{q}_n)^2 \\
MAE &= \frac{1}{\sum_{g\in G}\sum_{n\in\tilde{N}_g} 1}\sum_{g\in G}\sum_{n\in\tilde{N}_g}|q_n-\tilde{q}_n| \\
FPR &= \frac{The\ number\ of\ failed\ predicted\ requests}{The\ number\ of\ requests}\times 100\%
\end{aligned}
\tag{58}
$$

7.3.3. Results and discussion

Table 3 demonstrates the request size and result of the measurement of prediction metrics from Aug. 25 to Aug. 31. The model demonstrates a capacity for reasonable accuracy on certain days, as evidenced by lower MSE, MAE and FPR, particularly on Aug. 27. However, the highest MSE reaches to 0.3134, recorded on Aug. 28, just one day after its best recording,

indicating variability in the model's prediction accuracy across different days. Even with prediction error, the average FPR of seven days is 14.6%, which is still not greater than the high-setting prediction error parameters in Section 7.2.4.3.

Based on the predicted demand, we applied our proposed algorithm and two other well-performed benchmarks (individual DRL and Local search methods) discussed in Section 7.2. Three algorithms are performed with demand prediction, which are referred to as 'DRL + IL + Local Search + LSTM', 'DRL + LSTM', and 'Local search + LSTM', respectively. In addition, we compare them with our proposed algorithm without a demand prediction model (referred to as 'without prediction'), to solve the actual optimization problem for those 7 days. For each period, we set the training steps to 5000.

As measured by the four evaluation indicators described in Section 7.1, the results also are compared with the results of the historical operation provided by 'Shared Bus' project (referred to as 'existing approach'), which is claimed using hybrid heuristic algorithms including GA. Without loss of generality, we set 10 random seeds for each designed algorithm.

Table 3 Demand prediction metrics from Aug. 25 to Aug. 31

Date	Aug. 25	Aug. 26	Aug. 27	Aug. 28	Aug. 29	Aug. 30	Aug.31
Request size	105	113	113	67	74	113	101
MSE	0.2381	0.2832	0.115	0.3134	0.1622	0.2655	0.2376
MAE	0.1429	0.1947	0.0796	0.2239	0.1081	0.1593	0.1584
FPR (%)	11.43	17.7	8.85	25.37	13.51	11.5	13.86

Table 4 reports the results of the overall objective value on average of each test day. Despite the slight increase in planning time, our proposed algorithm reaches the best performance in terms of objective value, with gaps of 5.9%, 7.5%, 13.3% and 50.1%, compared to other results respectively. The average planning time is approximately 20 seconds for each period. Note that the experiments were conducted using a personal computer. Service providers usually have more powerful servers that can computer 10-20 times faster. Therefore, the results can scale to operational durations with immediacy requirements.

Fig. 14 presents the specific performance result of each algorithm and state-of-the-practice. Evidently, our proposed algorithm keeps its superiority over other approaches, with the best results of indicators throughout the seven days. Compared to the 'actual data', all our designed algorithms improve the EAUC, WAFI, ALAT and RR indicators considerably. Among them, the EAUC indicators witness the average reductions in 7 days of 24.3%, 19.7%, 18.8% and 19.3% respectively, while the WAFI indicators of those drop by 76.1%, 72.8%, 73.4% and 69.7%, respectively. The ALATs decrease by 87.6%, 86.1%, 79.2% and 73.5%, while the RRs increase to almost 100%. These improvements highlight the promising application results of our approaches. Interestingly, we observe that the ALAT indicators of the 'Local search + LSTM' and 'without prediction' approaches on Aug. 26 are approximately 0.266 and 0.402 minutes higher than that of the 'actual data'. However, this trade-off is reasonable as it significantly improves the response rate for passengers. In fact, on that day, the RR indicators of these two approaches reached almost 100%, whereas the RR of the actual data remains the lowest value at 74.8%.

We proceed to evaluate the advantages of introducing demand prediction operation. By comparing the 'DRL + IL + Local Search + LSTM' with 'without prediction', the improvements of the four indicators by the former in 7 days are reported as follows: the EAUC decreased by 6.1%; the WAFI decreased by 21.1%; the ALAT decrease by 53.3%, and the RR slightly drop by 0.04%. This result shows that incorporating demand prediction greatly improves cost and fairness indicators while reducing late arrival time, all with a negligible impact on the response rate.

Table 4 Metrics of the objective value and planning time of the case study

Algorithm	Objective value (on average)	Average planning time per period (sec)
DRL + IL + Local search + LSTM	456.5	21.2

DRL + LSTM	483.6	18.1
Local search + LSTM	490.9	11.2
Without prediction	517.3	19.5
Existing approach	685	–

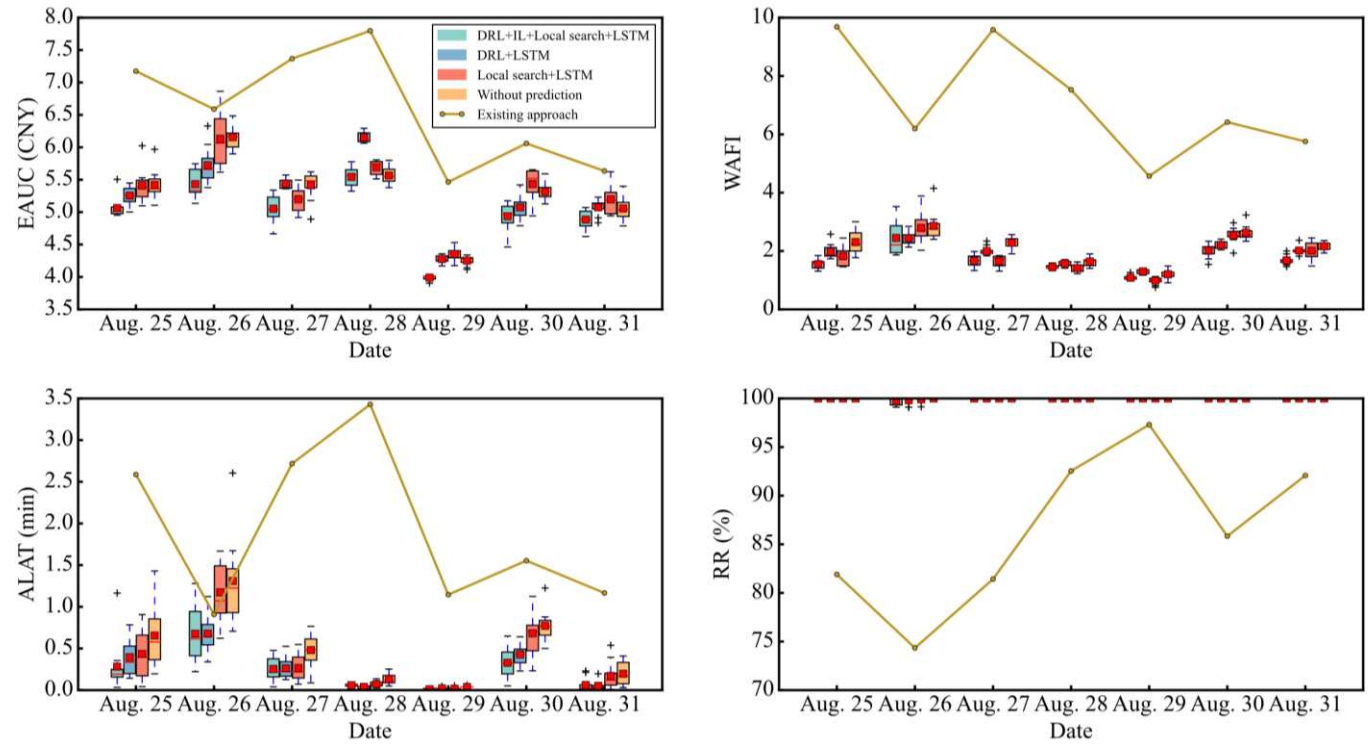

Fig. 14 Results of the case study

8. Concluding remarks

The flexible bus service is an essential component of the multimodal transit system. In this paper, we investigate prediction failure risk-aware dynamic scheduling FBS with hybrid requests and fairness considerations. The problem is formulated as a multi-objective sequential decision process and addressed by a new MARL framework integrating imitation learning and heuristics. The demand forecasting and prediction error correction modules are embedded into the MARL framework to enhance the foresight of the planned scheme. Furthermore, we leverage the valuable time-related information about the real-time spatio-temporal relationship between operating vehicles and passengers, and further propose an improved ϵ-greedy strategy to improve exploration efficiency. In addition, we integrate the MARL framework with a local search and the IL approach. In particular, three customized operators and the 'entropy' metric are proposed in the local search method. This combination not only utilizes the great exploration capability of the RL but also the exploitation capability of the local search method, to further improve the performance in improving solution quality.

Our model is validated by the Sioux Falls network and a real-world case study in Guangzhou. We provide four evaluation indicators to investigate the performance of our proposed algorithm as well as several benchmarks solving both single-period and multi-period problems for various operational settings. The results highlight the benefits of our proposed method: 1) it is comparable to the prevailing RL approaches as well as other benchmarks in terms of solution quality in both static and dynamic scenarios; 2) incorporating demand prediction method into the flexible bus operation can increase the system's profit and improve its service level for passengers. The improvement due to prediction is significant even when the prediction is

imperfect because the correction mechanism for prediction errors ensures the effectiveness of our algorithm.

The results also can provide several useful managerial implications to the service provider. First, despite the presence of potential prediction errors, the service provider should operate with demand prediction by leveraging the historical travel demand data to yield a foresighted scheme, while striving to enhance prediction accuracy. Second, the time-related information about the real-time spatio-temporal relationship between operating vehicles and requests is valuable, since it provides excellent guidance in the dynamic scenario. Third, the important operational settings affect the outcome of our algorithm essentially: 1) there is a trade-off between the request scale and flexibility of route adjustment frequency due to different period lengths in dynamic planning, whereas the operation with demand prediction could resolve this conflict more compatibly; 2) there is a trade-off between cost objective and fairness objective in the hybrid request mode, and in practice, the service provider can weigh the objectives based on the relative importance between different groups of passengers.

Our approach has the potential to act as a generic methodology for solving a wide range of similarly structured dynamic optimization problems in other application fields. In subsequent work, more exogenous factors and solution approaches can be embedded in our modelling framework. For instance, while this paper primarily focuses on the dynamic scheduling problem with a single-vehicle type, it might be interesting to explore the heterogeneous dynamic scheduling problem with different vehicle types and capacities. Another extension to consider is to combine the present modelling framework with pricing optimization in reaction to passengers' willingness to pay, to further increase ride pooling and improve profitability.

Acknowledgements

This work is jointly supported by the National Science Foundation of China (Project No. 72071079, 52272310), Guangdong Basic and Applied Research Foundation (Project No. 2023A1515011696), and the Fundamental Research Funds for the Central Universities (Project No. 2023ZYGXZR026).

Appendix A. Pruning of action space

A1. Pruning subject to visiting orders

It is necessary to conduct the pruning method on action space subject to visiting orders to avoid the waste of redundant and invalid decisions. First, to ensure that a certain request cannot be assigned to more than one bus, at the decision phase for any bus $k \in K$, $a_{g,t,k}^{\mathcal{M}} = i_n^p$ can be considered as a valid action only when $t_n^p = \emptyset$, i.e., request n has not been picked up by any bus yet. Second, we need to ensure that the delivery of a request must come after its pick-up event, i.e., for a request n, its delivery must come after its pick-up. In this case, we can define $N^{\alpha}(i_{g,t,k})$ as the set of on-board requests for bus k arriving at $i_{g,t,k}$. Therefore, for bus k, if request $n \in N^{\alpha}(i_{g,t,k})$, then delivering request n (i.e., the action $a_{g,t,k}^{\mathcal{M}} = i_n^d$) is considered a valid action for bus k at that moment. On the other hand, if request n does not belong to $N_{g,t,k}^{\alpha}$, the action of $a_{g,t,k}^{\mathcal{M}} = i_n^d$ is removed from $\hat{A}_{g,t,k}$.

A2. Pruning subject to tardiness constraints

The sequence of pick-up and drop-off stops of any vehicle is subject to the tardiness constraint (Section 4.5.3). Specifically, if $\left|N^{\alpha}(i_{g,t,k})\right| = 0$, i.e., there is an on-board request with bus k, then no pruning subject to the tardiness constraint is required. If $\left|N^{\alpha}(i_{g,t,k})\right| \neq 0$, for any unassigned or unfinished request, whether it can be served (including pick-up and delivery) by bus k is determined by its present location $i_{g,t,k}$, $N^{\alpha}(i_{g,t,k})$ and the pick-up time of the related requests of

$N^{\alpha}(i_{g,t,k})$. There are two scenes to identify whether bus k can visit a stop to serve request n:

Scene 1: To identify whether $a_{g,t,k}^{\mathcal{M}} = i_n^p$ is a valid action for bus k when request n has not been assigned to any vehicle yet, i.e., $n \notin N^{\alpha}(i_{g,t,k})$. In this situation, if there exists a trip sequence ℓ that accommodates every request of $N^{\alpha}(i_{g,t,k})$ and request n could satisfy the tardiness constraint, then i_n^p could be a valid action for bus k. The beginning stop of the sequence is certain, i.e., i_n^p, the pick-up stop of request n. Specifically, considering a possible sequence $\ell = \begin{pmatrix} & & \overbrace{\quad\quad}^{|N^{\alpha}(i_{g,t,k})|+1} \\ i_n^p, & \cdots, i_n^d, & \cdots \end{pmatrix}$, the pick-up time and drop-off time of every request including n can be derived from the route sequence and previous information (where the pick-up time of all requests is known except request n). If there exists a sequence ℓ which satisfies Eqs. (59)-(61), then $a_{g,t,k}^{\mathcal{M}} = i_n^p$ is valid for bus k at time interval t of period g; otherwise, it should be removed from $\hat{A}_{g,t,k}$.

$$t_n^p = t_{g,t,k}^{arl} + DIS\left(i_{g,t,k}, i_n^p\right)/vel + a_{g,t,k}^{\mathcal{H}^*} \tag{59}$$

$$t_{n_1}^d = t_n^p + \sum_{i^- \in \ell, i \in \ell}^{i=i_{n_1}^d} \frac{DIS(i^-, i)}{vel}, \forall n_1 \in N^{\alpha}(i_{g,t,k}) \cap \{n\} \tag{60}$$

$$t_{n_1}^d - t_{n_1}^p \leq \frac{DIS\left(i_{n1}^p, i_{n1}^d\right)}{vel} \cdot \alpha, \forall n_1 \in N^{\alpha}(i_{g,t,k}) \cap \{n\} \tag{61}$$

where i^- denotes the stop preceding i in sequence ℓ; Eq. (59) calculates the pick-up time for request n if $a_{g,t,k}^{\mathcal{M}} = p_n$; Eq. (60) calculates the drop-off time of requests of $N^{\alpha}(i_{g,t,k}) \cap \{n\}$ delivered by bus k following sequence ℓ; Eq. (61) is set to ensure that the travel time of each request of $N^{\alpha}(i_{g,t,k}) \cap \{n\}$ meets the tardiness constraint.

Scene 2: To identify whether $a_{g,t,k}^{\mathcal{M}} = i_n^d$ is a valid action for bus k when request n has been assigned to bus k but not delivered yet, i.e., $n \in N^{\alpha}(i_{g,t,k})$. In this situation, we only need to consider the sequence ℓ for delivering $N^{\alpha}(i_{g,t,k})$ that could satisfy the tardiness constraint. Similarly, assuming the beginning stop of ℓ is $i_{g,t,k}$, if there is no feasible sequence $\ell = \begin{pmatrix} & \overbrace{\quad\quad}^{|N^{\alpha}(i_{g,t,k})|} \\ i_{g,t,k}, & \cdots i_n^d, \cdots \end{pmatrix}$ that could satisfy Eqs. (62)-(63) then $a_{g,t,k}^{\mathcal{M}} = i_n^d$ is invalid for bus k at present, which should be removed from $\hat{A}_{g,t,k}$.

$$t_{n_1}^d = t_{g,t,k}^{arl} + \sum_{i^- \in \ell, i \in \ell}^{i=i_{n_1}^d} \frac{DIS(i^-, i)}{vel}, \forall n_1 \in N^{\alpha}(i_{g,t,k}) \tag{62}$$

$$t_{n_1}^d - t_{n_1}^p \leq \frac{DIS\left(i_{n1}^p, i_{n1}^d\right)}{vel} \cdot \alpha, \forall n_1 \in N^{\alpha}(i_{g,t,k}) \tag{63}$$

To find out such a possible sequence, we could enumerate the possible permutations of the drop-off stops of $N^{\alpha}(i_{g,t,k}) \cap \{n\}$ if **scene 1** is considered, where the complexity is $O\left(\left(\left|N^{\alpha}(i_{g,t,k})\right| + 1\right)!\right)$, or the drop-off stops of $N^{\alpha}(i_{g,t,k})$ if **scene 2** is considered, where the complexity is $O\left(\left|N^{\alpha}(i_{g,t,k})\right|!\right)$.

A3. Pruning subject to movement constraints

Due to the presence of the movement constraint (Section 4.5.2), each bus needs to complete its route in period g before the period is over. This involves the available decision of whether bus k could visit a new stop or just hold temporarily, rather than return to the depot at a particular time interval, which is determined by $i_{g,t,k}$, $t_{g,t,k}^{arl}$ and $N^{\alpha}(i_{g,t,k})$. As mentioned in A2, the possible incoming sequence of bus k comes from the permutations of $N_{g,t,k}^{\alpha} \cap \{n\}$ if it decides to pick up an unassigned request n, or the permutations of $N^{\alpha}(i_{g,t,k})$ if it decides to deliver one of its incomplete requests. In this situation, we need to

consider three scenes to identify the feasibility of the movement constraint when $o_{g,t,k}$ is given.

Scene 1: To identify whether bus k could pick up request n. In this case, if there exists a sequence ℓ accommodating $N^{\alpha}(i_{g,t,k}) \cap \{n\}$ with i_n^p as the beginning stop, i.e., $\ell = \begin{pmatrix} & & \overbrace{\quad}^{|N^{\alpha}(i_{g,t,k})|+1} \\ i_n^p, & \cdots, i_n^d, & \cdots \end{pmatrix}$ that could satisfy Eq. (64), which means bus k could finish the delivery of ℓ and return to any depot before the end of the period $|G|$, then $a_{g,t,k}^{\mathcal{M}} = i_n^p$ is a valid action for bus k; otherwise, it is removed from $\hat{A}_{g,t,k}$.

$$t_{g,t,k}^{arl} + \frac{DIS(i_{g,t,k}, i_n^p)}{vel} + a_{g,t,k}^{\mathcal{H}^*} + \sum_{i^- \in \ell, i \in \ell} \frac{DIS(i^-, i)}{vel} + \frac{DIS(\ell_{-1}, j)}{vel} \leq |G| \cdot H, \exists j \in J \tag{64}$$

where ℓ_{-1} denotes the ending stop of ℓ.

Scene 2: To identify whether bus k could deliver request n. In this case, if there exists a sequence ℓ accommodating $N^{\alpha}(i_{g,t,k})$ with $i_{g,t,k}$ as the beginning stop, i.e., $\ell = \begin{pmatrix} & \overbrace{\quad}^{|N^{\alpha}(i_{g,t,k})|} \\ i_{g,t,k}, & \cdots i_n^d, & \cdots \end{pmatrix}$ that could satisfy Eq. (65), then $a_{g,t,k}^{\mathcal{M}} = i_n^d$ is a valid action for the bus; otherwise, $a_{g,t,k}^{\mathcal{M}} = i_n^d$ needs to be removed from $\hat{A}_{g,t,k}$.

$$t_{g,t,k}^{arl} + \sum_{i^- \in \ell, i \in \ell} \frac{DIS(i^-, i)}{vel} + \frac{DIS(\ell_{-1}, j)}{vel} \leq |G| \cdot H, \exists j \in J \tag{65}$$

Scene 3: To identify whether the bus could return to the depot at present. For any employed bus k, it is only allowed to return to the depot when it completes its route of the period and the period is nearly over, because frequent travel of returning to the depot of a bus will increase the unnecessary system cost. To this end, we presumably set $\forall j \in J$ as invalid actions. However, if neither Eq. (64) nor Eq. (65) is satisfied, then we set $a_{g,t,k}^{\mathcal{M}} = \forall j$ as the only valid actions of bus k at present.

Since we have enumerated all the possible sequences in A2, the computation time of pruning subject to the movement constraint takes only a constant time.

A4. Pruning subject to capacity constraints

As far as the capacity constraint (Section 4.5.1) is concerned, whether bus k could pick up an unassigned request n is only determined by $z_{g,t,k}$, the present number of onboard passengers of bus k and q_n, and the reserved seats of the request n. If Eq. (66) is satisfied, then i_n^p is a valid action subject to the capacity constraint; otherwise, $a_{g,t,k}^{\mathcal{M}} = i_n^p$ needs to be removed from $\hat{A}_{g,t,k}$.

$$z_{g,t,k} + q_n \leq Z \tag{66}$$

Appendix B. Local search procedures

B1. Intra-loop move

Step 1: Select a loop

Select a route r_k via $r_k = \underset{k' \in K}{\operatorname{argmax}} \mu_1(k')$ from **priority list 1**. Then from **priority list 2** select a loop ℓ of route r_k via $\ell = \underset{\ell' \in r_k}{\operatorname{argmax}} \mu_2(\ell')$.

Step 2: Examine the move of requests in the selected loop

Select a request n from **priority list 3** via $n = \underset{n' \in \ell}{\operatorname{argmax}} \mu_3(n')$. Enumerate all feasible moves of n. To illustrate, considering loop sequence $\ell = \left(i_1^p, \cdots, i_{n-1}^p, i_{n-1}^d, i_n^p, i_n^d\right)$, first move $\left(i_n^p, i_n^d\right)$ to the ending position of ℓ (where it already is) , then move i_n^p to an earlier position once at a time, until i_n^p is right before i_1^p. For each move of i_n^p, check the feasibility of having i_n^d stay at its initial place, and move it to one position once at a time, until i_n^d reaching right after i_n^p. For each feasible move of $\left(i_n^p, i_n^d\right)$, calculate the route reward.

Remove n from the **priority list 3**. Repeat the above steps for the rest of the requests in ℓ.

Step 3: Identify the best move of $\left(i_n^p, i_n^d\right)$.

Among all solutions after the feasible moves in Step 2, if any, select one with the largest reward. If the new move yields a better solution than the original one, output the new solution; otherwise, stop the neighborhood move process and output the original solution. Remove ℓ from **priority list 2** and repeat the above steps until **priority list 2** is empty.

B2. Inter-loop move

The inter-loop move involves the change in a request and a loop. Thus, another potential metric $\mu_4(\ell, n)$ of loops is introduced to indicate which loop has the priority to accommodate the new request n, which can be calculated as follows:

$$\mu_4(\ell, n) = \frac{1}{|N(\ell)|} \cdot \sum_{n_1 \in N(\ell)} \left(t_{n_1}^l - t_{n_1}^p\right) - \frac{1}{|N(\ell)|} \cdot \sum_{i \in N(\ell)} \left[DIS\left(i, i_n^p\right)\right]/vel \tag{67}$$

where $\sum_{n_1 \in N(\ell)} \left(t_{n_1}^l - t_{n_1}^p\right)/|N(\ell)|$ is the average remaining slack time of each request of the loop ℓ in terms of reducing the late penalty, indicating how much time of ℓ could be adjusted to accommodate a new request. $\frac{1}{|N(\ell)|} \cdot \sum_{i \in N(\ell)} \left[DIS\left(i, i_n^p\right)\right]/vel$ indicates the average travel time from each stop in ℓ to the pick-up stop of the request n. Therefore, their difference, i.e., the value of $\mu_4(\ell, n)$, can measure the potential of inserting a particular request n into loop ℓ. Similarly, the new **priority list 4** can be constructed by calculating and ranking $\mu_4(\ell, n)$ of every loop when request n is given. Therefore, the inter-loop move options can be specified as follows:

Step 1: Select a request

Select a request n from **priority list 3** via $n = \underset{n'}{\operatorname{argmax}} \mu_3(n')$.

Step 2: Select a loop and insert the selected request into it

Select a loop ℓ (except the loop which previously included request n) from the **priority list 4** via $\ell = \underset{\ell', n \notin \ell'}{\operatorname{argmax}} \mu_4(\ell', n)$. Insert n into the ending position of ℓ.

Step 3: Examine all the feasible move

Considering moving $\left(i_n^p, i_n^d\right)$ to the ending position of ℓ, then perform Step 2 of the **Intra-loop move**, but only on request n. If there is any feasible move, calculate the reward; otherwise, remove ℓ from **priority list 4** and repeat the above steps until **priority list 4** is empty.

Step 4: Identify the best move of $\left(i_n^p, i_n^d\right)$.

Among all feasible moves in Step 3, if any, select the one with the largest reward increase; otherwise, stop and output the original solution. Remove n from **priority list 3** and repeat the above steps until **priority list 3** is empty.

B3. Insert-rejected-request move

The insert-rejected-request move involves inserting a rejected request into an existing loop. The procedure is specified as follows:

Step 1: Select a rejected request

Select a rejected request n with the largest rejection penalty cost, i.e., $n = \underset{n'}{\operatorname{argmax}}\, q_{n'} \cdot \delta^b$.

Step 2: Select a loop and insert the rejected request into it

Select a loop ℓ from the **priority list 4** via $\ell = \underset{\ell', n \notin \ell'}{\operatorname{argmax}}\, \mu_4(\ell', n)$. Insert n into the ending position of ℓ.

Step 3: Examine all the feasible moves and identify the best move as **Step 3** and **Step 4** of the **Inter-loop move**.

Appendix C. Comparison of indicators using operation with and without demand prediction (10 random seeds)

Seed	With prediction					Without prediction				
	EAUC (CNY)	WAFI	ALAT (min)	RR (%)	Planning time per period (s)	EAUC (CNY)	WAFI	ALAT (min)	RR (%)	Planning time per period (s)
1	2.998	2.298	0.3933	100	33	7.358	9.937	9.006	100	20
2	3.195	2.81	0.5978	100	32.83	7.288	10.289	9.194	100	19.5
3	3.06	2.331	0.4775	100	32.5	7.472	9.295	9.279	100	19.17
4	3.12	2.348	0.382	100	32	7.299	9.648	9.188	100	19.83
5	3.331	3.263	0.8736	100	32.67	7.567	9.935	9.642	100	19.83
6	2.892	2.285	0.3708	100	32.5	7.478	9.588	9.582	100	20
7	3.068	2.447	0.3966	100	32.83	7.562	9.478	9.473	100	20.17
8	3.14	2.714	0.648	100	32.83	7.638	9.575	9.552	100	19.67
9	3.082	2.547	0.419	100	32.83	7.285	9.548	9.145	100	19.83
10	3.134	2.662	0.5028	100	32.83	7.655	9.348	9.382	100	20
Average	**3.102**	**2.571**	**0.5061**	100	32.68	7.460	9.664	9.344	100	**19.8**

Appendix D. Description of the prediction model

For the prediction model, we harness the LSTM model to forecast OD demand. We commence by aggregating time series demand data at each stop, representing it as a vector of length $\mathcal{T}$ (in this case we set $\mathcal{T} = 7$), where $v_i^{t-\mathcal{T}\sim t} = \left(q_i^{t-\mathcal{T}}, q_i^{t-\mathcal{T}+1}, \cdots, q_i^t\right)$. Here, q_i^t denotes the number of passengers departing from stop i during time interval t. Then, we utilize the LSTM model with the input vector $v_i^{t-\mathcal{T}\sim t}$ to obtain the hidden state vector of stop i by $h_i^{p,t} = LSTM\left(v_i^{t-\mathcal{T}\sim t}, h_i^{p,t-1}\right)$, where $h_i^{p,t}$ captures the OD information from stop i up to t, as encoded in the memory of LSTM. Furthermore, to construct the predicted OD demand matrix, we define a learnable transition matrix $\mathcal{W}$ to capture the transition probabilities of OD traffic demand. Consequently, we can predict the demand from stop i to j at $t+1$, expressed as $\tilde{q}_{i,j}^{t+1} = \left(\mathcal{W}h_i^{p,t}\right)^T h_j^{p,t}$. The predicted OD demand matrix $\tilde{Q}^{t+1}$ is obtained by computing $\tilde{q}_{i,j}^{t+1}$ for every pair of stops. The optimization of the LSTM and $\mathcal{W}$ is conducted through gradient descent by minimizing the mean squared error loss function, formalized as $L^{predict} = \frac{1}{|Q^{t+1}|\times\mathcal{N}} \sum_1^{\mathcal{N}} ||Q^{t+1} - \tilde{Q}^{t+1}||$, where Q^{t+1} is the real value of the OD demand matrix at $t+1$, and $\mathcal{N}$ is the size of the training sample.

References

Ahamed T., Zou B., Farazi N. P., Tulabandhula T., 2021. Deep Reinforcement Learning for Crowdsourced Urban Delivery. Transportation Research Part B, 152, 227-257.

Bello I., Pham H., Le Q.V., Norouzi M., Bengio S., 2016. Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940.

Braekers K., Caris A., Janssens G.K., 2014. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. Transportation Research Part B, 67, 166-186.

Braekers K., Kovacs A.A., 2016. A multi-period dial-a-ride problem with driver consistency. Transportation Research Part B, 94, 355-377.

Cavallaro F., Nocera S., 2023. Flexible-route integrated passenger-freight transport in rural areas. Transportation Research Part A, 169, 103604.

Chen P., Nie Y., 2017a. Analysis of an idealized system of demand adaptive paired-line hybrid transit. Transportation Research Part B, 102, 38-54.

Chen P., Nie Y., 2017b. Connecting e-hailing to mass transit platform: Analysis of relative spatial position. Transportation. Research Part C, 77, 444-461.

Chen X.Y., Tian Y., 2019. Learning to perform local rewriting for combinatorial optimization. Adv. Neural Inf. Process. Syst. 32, 6281-6292.

Daganzo C.F., 1978. An approximate analytic model of many-to-many demand responsive transportation system. Transportation Research, 12, 325-333.

Delgado J.M.D., Oyedele L., 2022. Robotics in construction: A critical review of the reinforcement learning and imitation learning paradigms. Advanced Engineering Informatics, 54, 101787.

Desrosiers J., Dumas Y., Soumis F., 1986. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. American Journal of Mathematical and Management Sciences, 6(3-4), 301-325.

Detti P., Papalini F., de Lara G.Z.M., 2017. A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. Omega, 70, 1-14.

Diana M., Dessouky M.M., Xia N., 2006. A model for the fleet sizing of demand responsive transportation services with time windows. Transportation Res. Part B, 40, 651-666.

Donne D.D., Afandari L., Archetti C., Ljubić I., 2023. Freight-on-Transit for urban last-mile deliveries: A strategic planning approach. Transportation Research Part B, 169, 53-81.

Drori I., Kharkar A., Sickinger W.R., Kates B., Ma Q., Ge S., Dolev E., Dietrich B., Williamson D.P., Udell M., 2020. Learning to solve combinatorial optimization problems on real-world graphs in linear time. In: 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 19-24.

Fehn F., Engelhardt R., Dandl F., Bogenberger K., Busch F., 2023. Integrating parcel deliveries into a ride-pooling service-An agent-based simulation study. Transportation Research Part A, 169, 103580.

He D., Ceder A., Zhang W., Guan W., Qi G., 2023. Optimization of a rural bus service integrated with e-commerce deliveries guided by a new sustainable policy in China. Transportation Research Part E, 172, 103069.

Ho S.C., Szeto W.Y., Kuo Y., Leung J.M.Y., Petering M., Tou T.W.H., 2018. A survey of dial-a-ride problems: Literature review and recent developments. Transportation Research Part B, 111, 395-421.

Jiang G., Lam S.K., Ning F., He P., Xie J., 2020. Peak-Hour Vehicle Routing for First-Mile Transportation: Problem Formulation and Algorithms. IEEE Transactions on Intelligent Transportation Systems, 21(8), 3308-3321.

Joe W., Lau H.C., 2020. Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers. In: Proceedings of the International Conference on Automated Planning and Scheduling, 30, 394-402.

Kalakanti A.K., Verma S., Paul T., Yoshida T., 2019. RL SolVeR pro: Reinforcement learning for solving vehicle routing problem. In: 2019 1st International Conference on Artificial Intelligence and Data Sciences (AiDAS). IEEE, 94-99.

Kim M., Schonfeld P., 2014. Integration of conventional and flexible bus services with timed transfers. Transportation Research Part B, 68, 76-97.

Kim M., Schonfeld P., 2015. Maximizing net benefits for conventional and flexible bus services. Transportation Research Part A, 80, 116-133.

Kool W., Van Hoof H., Welling M., 2019. Attention, learn to solve routing problems! In Proceedings of the International Conference on Learning Representations.

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, 168, 102931.

Lee E., Cen X., Lo H.K., Ng K.F., 2021. Designing zonal-based flexible bus services under stochastic demand. Transportation Science, 55(6), 1227-1458.

Li Y., Zheng Y., Yang Q., 2018. Dynamic Bike Reposition: A Spatio-Temporal Reinforcement Learning Approach. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 1724-1733.

Lim A., Zhang Z., Qin, H., 2017. Pickup and delivery service with manpower planning in Hong Kong public hospitals. Transportation Science, 51 (2), 688-705.

Lin K., Zhao R., Xu Z., Zhou J, 2018. Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 1774-1783.

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. Transportation Research Part E, 161, 102694.

Lowe R., Wu Y., Tamar A., Harb J., Abbeel P., Mordatch I., 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 6382-6393.

Lyu Y., Chow C.Y., Lee V.C.S., Ng J.K.Y., Li Y., Zeng J., 2019. CB-Planner: A bus line planning framework for customized bus systems. Transportation Research Part C, 101, 233-253.

Mao C., Liu Y., Shen Z., 2020. Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach. Transportation Research Part C, 115, 102626.

Masmoudi M.A., Braekers K., Masmoudi M., Dammak A., 2017. A hybrid genetic algorithm for the heterogeneous dial-a-ride problem. Computers & Operations research, 81, 1-13.

Masmoudi M.A., Hosny M., Braekers K., Dammak A., 2016. Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. Transportation Research Part E, 96, 60-80.

Masson R., Lehuédé F., Péton O., 2014. The dial-a-ride problem with transfers. Computers & Operations Research, 41, 12-23.

Molenbruch Y., Braekers K., Caris A., 2017. Benefits of horizontal cooperation in dial-a-ride services. Transportation Research Part E, 107, 97-119.

Montenegro B.D.G., Sörensen K., Vansteenwegen P., 2021. A large neighborhood search algorithm to optimize ademand-responsive feeder service. Transporation Research Part C, 127, 103102.

Nazari M., Oroojlooy A., Snyder L.V., Takáč M., 2018. Reinforcement learning for solving the vehicle routing problem. In: Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018).

Oda, T., Joe-Wong, C., 2018. April. MOVI: A model-free approach to dynamic fleet management. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, 2708-2716.

Peter S., Guy L., Audrunas G., Wojciech M.C., Vinicius Z., Max J., Marc L., Nicolas S., Joel Z.L., Karl T., Thore G., 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In Proceedings of the

17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '18). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2085-2087.

Posada M., Andersson H., Häll C.H., 2017. The integrated dial-a-ride problem with timetabled fixed route service. Public Transport, 9(1-2), 217-241.

Psaraftis H.N., 1980. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. Transportation Science, 14(2), 130-154.

Quadrifoglio L., Li X., 2009. A methodology to derive the critical demand density for designing and operating feeder transit services. Transportation Research Part B, 43(10), 922–935.

Ren J., Jin W., Wu W., 2022. Multi-objective optimization for multi-depot heterogeneous first-mile transportation system considering requests' preference ranks for pick-up stops. Transportmetrica A, 19(3), 2103205.

Schasché S.E., Sposato R.G., Hampl N., 2022. The dilemma of demand-responsive transport services in rural areas: Conflicting expectations and weak user acceptance. Transport Policy, 126, 43-54.

Schenekemberg C.M., Chaves A.A., Coelho L.C., Guimarães T.A., Avelino G.G., 2022. The dial-a-ride problem with private fleet and common carrier. Computers&Operations Research, 147, 105933.

Shehadeh K.S., Wang H., Zhang P., 2021. Fleet sizing and allocation for on-demand last-mile transportation systems. Transportation Research Part C, 132, 103387.

Singh A., Al-Abbasi A., Aggarwal V., 2019. December. A reinforcement learning based algorithm for multi-hop ride-sharing: Model-free approach. In: Neural Information Processing Systems (Neurips) Workshop.

Tan, K.C., Lee, L.H., Zhu, Q.L., Ou, K., 2001. Heuristic methods for vehicle routing problem with time windows. Artificial intelligence in Engineering 15 (3), 281–295.

Tong L., Zhou L., Liu J., Zhou X., 2017. Customized bus service design for jointly optimizing passenger-to-vehicle assignment and vehicle routing. Transportation Research Part C, 85, 451-475.

Vansteenwegen P., Melis L., Aktaş D., Montenegro B., Vieira F., Sörensen K., 2022. A survey on demand-responsive public bus systems. Transportation Research Part C, 137, 103573.

Wang H., 2019. Routing and scheduling for a last-mile transportation system. Transportation Science, 53(1), 131-147.

Wang Y., Yin H., Chen H., Wo T., Xu J., Zheng K., 2019. Origin-Destination Matrix Prediction via Graph Convolution: a New Perspective of Passenger Demand Modeling. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19). Association for Computing Machinery, New York, NY, USA, 1227–1235.

Wu W., Zhou W., Lin Y., Xie Y., Jin W., 2021. A hybrid metaheuristic algorithm for location inventory routing problem with time windows and fuel consumption. Expert Systems with Applications, 166, 114034.

Wu W., Li Y., 2024. Pareto truck fleet sizing for bike relocation with stochastic demand: Risk-averse multi-stage approximate stochastic programming. Transportation Research Part E, 183, 103418.

Wu W., Zou H., Liu R., 2024. Prediction-failure-risk-aware online dial-a-ride scheduling considering spatial demand correlation via approximate dynamic programming and scenario approach. Transportation Research Part C, 104801.

Yan Y., Andy H.F.C., Chin P.H., Kuo Y.H., Wu Q., Ying C., 2022. Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities. Transportation Research Part E, 162, 102712.

Zhang J., Wang D.Z.W, Meng M., 2017. Analyzing customized bus service on a multimodal travel corridor: An analytical modeling approach. Journal of Transportation Engineering. Part A: Systems, 143(11), 1-12.

Zhang Z., Liu H., Zhou M., Wang J., 2023. Solving dynamic traveling salesman problems with deep reinforcement learning. IEEE Transactions on Neural Networks and Learning Systems, 34(4), 2119-2132.

Zhao J., Mao M., Zhao X., Zou J., 2020. A hybrid of deep reinforcement learning and local search for the vehicle routing problems. IEEE Transactions on Intelligent Transportation Systems, 22(11), 7208-7218.

Zhou T., M.Y. Law Kris M. Y. L. K., Creighton D., Wu C., 2022. GMIX: Graph-based spatial–temporal multi-agent reinforcement learning for dynamic electric vehicle dispatching system. Transportation Research Part C, 144, 103886.