Transportation Research Part E 186 (2024) 103551

Contents lists available at ScienceDirect

Transportation Research Part E

journal homepage: www.elsevier.com/locate/tre

Vehicle dispatching and routing of on-demand intercity ride-pooling services: A multi-agent hierarchical reinforcement learning approach

Jinhua Si [a], Fang He [a,*], Xi Lin [b], Xindi Tang [c]

[a] Department of Industrial Engineering, Tsinghua University, Beijing 100084, PR China
[b] Department of Civil and Environmental Engineering, University of Michigan, Ann Arbor, 48109, United States of America
[c] School of Management Science and Engineering, Central University of Finance and Economics, Beijing 100081, PR China

ARTICLE INFO

Keywords:
Intercity
Ride-pooling
Vehicle dispatching
Multi-agent hierarchical reinforcement learning

ABSTRACT

The integrated development of city clusters has given rise to an increasing demand for intercity travel. Intercity ride-pooling service exhibits considerable potential in upgrading traditional intercity bus services by implementing demand-responsive enhancements. Nevertheless, its online operations suffer the inherent complexities due to the coupling of vehicle resource allocation among cities and pooled-ride vehicle routing. To tackle these challenges, this study proposes a two-level framework designed to facilitate online fleet management. Specifically, a novel multi-agent feudal reinforcement learning model is proposed at the upper level of the framework to cooperatively assign idle vehicles to different intercity lines, while the lower level updates the routes of vehicles using an adaptive large neighborhood search heuristic. Numerical studies based on the realistic dataset of Xiamen and its surrounding cities in China show that the proposed framework effectively mitigates the supply and demand imbalances, and achieves significant improvement in both the average daily system profit and order fulfillment ratio.

1. Introduction

Conventional intercity bus services are facing increasing operational pressures. The number of intercity bus users in many countries has declined significantly in recent years (Ganji et al., 2021). The scale of the intercity bus industry in China, where more than 70% passenger trips are delivered by road transport, has been shrinking over the past decade. A large number of bus stations in China shut down due to continued losses, and the nationwide number of passenger vehicles has decreased by 30.06% compared to 2015 (MOT, 2022). The intercity bus industry in the United States also began shrinking after 2015. Greyhound, which operates the largest intercity bus service in North America, reduced its operations by 16% between 2016 and 2020 (Schwieterman et al., 2021). The possible reasons are twofold. From the *supply* perspective, the steady increase in car ownership compresses the demand for short-haul bus lines, while the infrastructure construction of high-speed railways and airlines tends to be improved in more regions, delivering fatal blows to many longer-haul bus lines. From the *demand* perspective, people's travel behaviors have been reshaped in the mobile Internet era, placing increased emphasis on convenience, comfort, and privacy when traveling. The attractiveness of traditional intercity buses is diminishing due to heightened competition from alternative modes of intercity transportation.

With the development of mobile Internet and the proliferation of smartphones, the intercity ride-pooling service within city clusters is becoming a promising direction for the digital transition and demand-responsive upgrading of the conventional intercity

* Corresponding author.
E-mail address: fanghe@tsinghua.edu.cn (F. He).

https://doi.org/10.1016/j.tre.2024.103551
Received 13 July 2023; Received in revised form 15 March 2024; Accepted 22 April 2024
Available online 30 April 2024

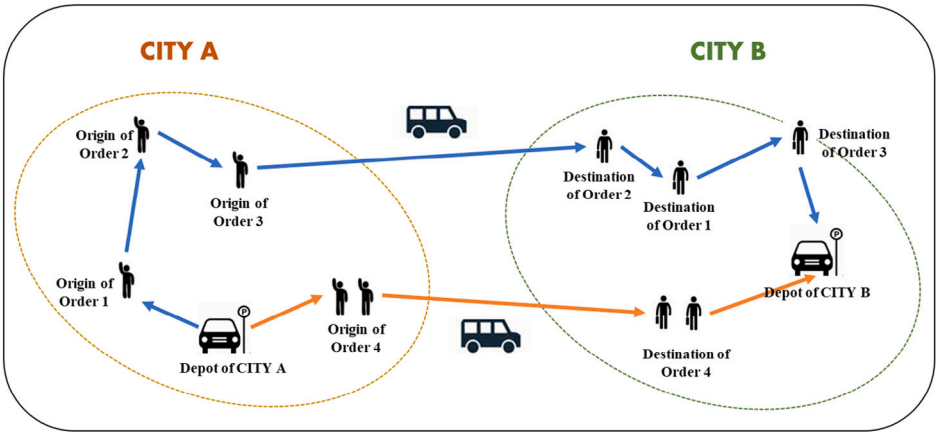

Fig. 1. Intercity passenger transport in a pooled-ride manner.

bus service. As an emerging travel mode, it utilizes vehicles with smaller capacities to provide door-to-door service for each passenger. Similar to urban ride-hailing platforms, a computing center makes centralized decisions in real time based on the information of orders and available vehicles. Each order is expected to specify the number of passengers, the locations for pick-up and drop-off in different cities, the time windows, and other necessary information. The computing center assigns orders to vehicles precisely and plans the route for each vehicle to pick up passengers in the departure city and deliver them to their destinations in a pooled-ride manner, as shown in Fig. 1.

The advantages of intercity ride-pooling services in city clusters are mainly manifested in three aspects. The *first* advantage is replacing the traditional fixed-schedule service with a dynamic ride-hailing service. The demand-responsive transit system, which has demonstrated its remarkable effectiveness in significantly enhancing the travel experience within urban public transport over the last decade (Wang and Yang, 2019; Liu and Ouyang, 2021; Vansteenwegen et al., 2022), holds great promise for delivering a more convenient and diversified travel experience in intercity transport. The demand-responsive services can also enhance system efficiency by matching supply and demand more intelligently and efficiently. The *second* advantage lies in the ability to plan fully flexible routes for each vehicle instead of fixed routes, offering door-to-door services to passengers. With the gradual expansion of urban areas and increased traffic congestion, the generalized cost for passengers concentrating at bus stations in the city has increased dramatically, which makes door-to-door services increasingly appealing and advantageous. The *third* advantage is that it fully considers the allocation of transportation resources at the city cluster level. The highly developed city cluster has become an increasingly significant geographic unit under the rapid urbanization process, where the strong socioeconomic connections between the cores and the peripheries brew up strong intercity travel demand (Fang and Yu, 2017). Road passenger transport is still competitive within city clusters, especially for those clusters with relatively low railway coverage. Rather than being subordinated to a fixed line between two cities, vehicles can be dispatched to any city within the cluster according to demand fluctuation, which can resolve the spatial imbalance between supply and demand.

While the aforementioned advantages highlight a promising market for intercity ride-pooling services within city clusters, there are still several issues that hinder the development of this innovative service. The core is how to build a highly effective and intelligent fleet operations strategy for the sophisticated dynamic system and unique scenarios of intercity travel. From the perspective of macroscopic spatiotemporal allocation of vehicle resources, platforms need to implement dynamic dispatching of fleets among multiple cities to reduce the gap between available seats and potential demand between each city pair in future periods. Extensive explorations have been conducted in the operational strategy optimization in related fields, such as urban ride-hailing and ride-sourcing platforms (Xu et al., 2018a; Tang et al., 2020; Guo et al., 2021; Kullman et al., 2022). From the perspective of microscopic order-matching and vehicle routing, the platform needs to solve the multi-vehicle pooled-ride routing problem to maximize the profit of each route in an online manner, which is a variant of the Pickup and Delivery Problem with Time Windows (PDPTW) or Dial-a-Ride Problem (DARP) (Cordeau, 2006; Ropke and Pisinger, 2006). The fleet operations strategy is based on the integrated optimization of macroscopic fleet dispatching and microscopic pooled-ride vehicle routing, which has been barely touched on in existing literature under intercity scenarios. The coupling structure makes it difficult to devise an efficient fleet management strategy using model-based approaches. Additionally, the differences between intracity and intercity travel lead to different optimization frameworks for ride-pooling services. The intercity demand-responsive transport system faces greater challenges in coping with supply and demand dynamics due to longer travel times. Additionally, relocation of idle vehicles is generally unreasonable in intercity ride-pooling services due to high travel costs. The intercity travel demand within a city cluster can be clustered into several mutually independent intercity lines, and the platform has to allocate vehicle resources to lines across broader temporal and spatial dimensions.

This study proposes an online two-level framework to dispatch vehicles among several cities effectively and update intercity pooled-ride vehicle routes dynamically. The *dispatching level* (upper level) assigns idle vehicles to intercity lines based on global

demand and supply information. The dispatching decisions for each city are derived synergistically using a novel multi-agent hierarchical reinforcement learning method, named multi-agent feudal networks (MFuN), to enhance agent cooperation and far-sighted assignment. The *routing level* (lower level) uses the adaptive large neighborhood search heuristic (ALNS) to solve the order-vehicle matching and pooled-ride routing problems at each time interval based on the decisions of the dispatching level. The two-level framework can provide online operational decisions while avoiding myopic actions under the guidance of deep reinforcement learning. The system aims to improve both the system profit and order fulfillment ratio, enabling service for more passengers within their reserved time windows. The main contributions of this study can be summarized as follows.

- This study focuses on the online fleet operations of on-demand intercity ride-pooling services, while the applications of demand-responsive services in the realm of intercity transport have received scarce research attention thus far. The coupling of vehicle resource allocation among cities and multi-vehicle pooled-ride routing problems brings challenges to proposing an integrated model for dispatching and routing. We also take into account the distinctive attributes of intercity travel when developing strategies, including long travel times and high intercity costs.
- We propose an integrated online operational framework for intercity ride-pooling services within a city cluster. The framework comprises two levels, with the upper level dedicated to addressing vehicle resource allocation among cities in the long term, and the lower level focusing on dynamic routing. At each dispatching horizon, the upper level of the framework optimizes the distribution of vehicle resources within a city cluster by dispatching vehicles to intercity lines to mitigate the spatiotemporal imbalances between supply and demand. At each matching interval with a shorter time scale, the lower level solves dynamic DARP for each intercity line respectively to enable efficient online management of the fleet at the scale of city clusters.
- We take advantage of a novel multi-agent hierarchical reinforcement learning method, named MFuN, in the upper level of the framework, to develop a non-myopic strategy for the stochastic dynamic vehicle resource allocation problem. MFuN shows favorable performance in promoting agent cooperation and capturing transition patterns in system states over extended horizons, and ensures that the actions of each agent contribute to maximizing the long-term profits of the entire system.
- We acquire realistic operational data of intercity ride-pooling services to evaluate the effectiveness of our framework. The numerical experiments conducted reveal that the proposed framework can develop proactive operational strategies to alleviate the spatiotemporal imbalance between supply and demand, leading to remarkable performance enhancements on both system profit and order fulfillment ratio.

The remainder of this paper proceeds as follows. Section 2 reviews recent literature addressing DARP and fleet dispatching problems. Section 3 specifically portrays the modeling of the problem. Section 4 introduces the proposed two-level operational framework. Section 5 conducts numerical experiments on two toy networks and one realistic network to demonstrate the optimization effectiveness under various supply and demand scenarios, and provide managerial insights for the platform operations. Finally, we summarize the study and provide an outlook on future research in Section 6.

2. Literature review

As an emerging service mode motivated by increasingly customized travel preferences and growing intercity population movements, intercity ride-pooling services remain largely unexplored in the literature. Nevertheless, researchers have made several progress in solving related problems, such as the pooled-ride routing problem and urban demand-responsive fleet dispatching problem.

Multi-vehicle pooled-ride routing problem is a variant of the Pickup and Delivery Problem with Time Windows (PDPTW) or Dial-a-Ride Problem (DARP) (Ho et al., 2018). This problem can be formulated as a mixed-integer linear program (MILP) problem, and existing solution methods can be categorized into heuristic methods (Ropke and Pisinger, 2006; Naccache et al., 2018; Goeke, 2019; Guo et al., 2022) and exact methods (Cordeau, 2006; Gschwind and Irnich, 2015; Braekers and Kovacs, 2016; Luo et al., 2019). However, for large-scale problems, exact algorithms such as branch-and-cut algorithms and branch-and-price algorithms are usually unable to provide optimal or even feasible solutions within an acceptable time, so heuristic algorithms such as neighborhood search algorithms are more widely applied in solving practical problems. Adaptive large neighborhood search (ALNS), first proposed by Ropke and Pisinger (2006), is one of the most effective heuristic algorithms to solve large-scale multi-vehicle dial-a-ride problems with pick-up and delivery time windows. ALNS utilizes multiple removal and insertion operators to enhance the movement in solution space in a simulated annealing metaheuristic manner. Subsequent studies have proposed many detailed techniques to improve the performance of the algorithm or to solve different variants of DARP (Demir et al., 2012; Ghilas et al., 2016; Gschwind and Drexl, 2019; Sun et al., 2020a).

The aforementioned routing algorithms for DARP are widely used in the fleet operations of urban demand-responsive services such as flexible bus systems and ride-pooling services. The rest of this section reviews studies in vehicle routing and fleet dispatching of these systems. We also consider several papers that focus on ride-hailing services with no shared rides, as their approaches can be effectively extended to ride-pooling services.

Flexible bus systems in practice can be classified into two categories. The first category is semi-flexible that each bus travels along a standard route and can slightly deviate from the predetermined route to serve emerging demand, while the second category is fully flexible that routes and timetables are determined based on demand (Vansteenwegen et al., 2022). Some studies model the semi-flexible bus scheduling and routing as static problems, among which the majority combine candidate stops into routes based on demand spatial distribution (Czioska et al., 2019; Sun et al., 2020b), while others collaboratively optimize the selection of bus

stops and suggestions for pick-up locations (Melis and Sörensen, 2022b). Matching of new orders and dynamic adjustment of routes are considered to develop dynamic models in some literature (Fielbaum et al., 2021; Tafreshian et al., 2021; Melis and Sörensen, 2022a). In the literature of fully flexible bus systems, the online vehicle routing problems are usually modeled as deterministic dynamic DARP and solved by the methods mentioned above (Agatz et al., 2012; Braekers and Kovacs, 2016; Huang et al., 2020; Guo et al., 2022; Wu et al., 2022). Certain literature uses stochastic optimization to capture the uncertainty of demand distribution and detour time (Lee et al., 2021).

The fleet operation problem for urban ride-hailing services constitutes one of the most extensively investigated topics of demand-responsive transport research. Although the majority of these studies predominantly concentrate on ride matching and vehicle dispatching with no shared ride, many studies have explored the operation problem of urban ride-pooling systems. Some studies proposed solution frameworks for the ride-pooling systems with requests in advance (Ma and Koutsopoulos, 2022), while others focused on the on-demand request matching and real-time vehicle routing. Alonso-Mora et al. (2017) and Simonetto et al. (2019) recast the real-time operation problem into a series of batch processes for request assignment and vehicle routing. The batch matching optimization for non-pooling services can be formulated as a bipartite matching problem (Tong et al., 2016; Qin et al., 2021b). However, because of the time window constraints for successive pick-up and delivery, the batch-matching process is more complicated when multiple requests can be served by the same vehicle (Ouyang et al., 2021). Some literature proposes dynamic optimization models to determine the optimal matching interval and radius (Guo et al., 2020; Yang et al., 2020; Qin et al., 2021a). Reinforcement learning methods for dynamic matching also have gained immense attention in recent years to prevent short-sighted decisions (Xu et al., 2018a; Tang et al., 2019; Qin et al., 2020).

Based on the aforementioned batch-matching framework, the spatiotemporal imbalance between supply and demand can be further mitigated by proactive vehicle dispatching. By leveraging extensive operational data, the platform can develop a sophisticated supply and demand prediction model, offering guidance for real-time vehicle dispatching (Ke et al., 2017; Guo and Zhang, 2020). Methods for fleet dispatching in urban ride-hailing services can be broadly classified into two categories. The *first* category models the problem as a Markov decision process and constructs various optimization models (Xu et al., 2018a). Various methods are developed to provide pooled-ride vehicle routing and rebalancing results under this stochastic environment, including shareability graph-based algorithm (Alonso-Mora et al., 2017; Tuncel et al., 2023), approximate dynamic programming approach (Yu and Shen, 2020), linear assignment algorithm (Simonetto et al., 2019), and data-driven metaheuristics (Bongiovanni et al., 2022). Lei et al. (2020) propose a two-layer framework, where the upper layer focuses on dynamic pricing and relocation, while the lower layer determines the passengers' choice behavior based on market equilibrium. Guo et al. (2021) employ a rolling-horizon robust optimization approach based on model predictive control (MPC) for vehicle location optimization.

The *second* category adopts model-free approaches to tackle large-scale fleet operations for ride-hailing services. Many studies use single-agent reinforcement learning to provide centralized decisions. Tang et al. (2020) designs a two-layer single-agent reinforcement learning framework to solve the problem of online dispatching, charging arrangement, and order matching for urban electric vehicles. Liu et al. (2022) propose a single-agent deep reinforcement learning approach for the vehicle dispatching problem, in which a single-agent queue is designed for dispatching multiple vehicles based on a global pruned action space. More studies utilize multi-agent reinforcement learning (MARL) framework to derive dispatching and scheduling strategies based on the well-trained value functions (Xu et al., 2018a; Jin et al., 2019; Guo and Xu, 2020; Jiao et al., 2021; Kullman et al., 2022). In these studies, an agent is usually defined as a vehicle or vehicles within a grid, with eligible actions including repositioning to a certain area and fulfilling certain orders. The innovations of these studies are mainly focused on applying frontier reinforcement learning techniques to enhance the model's effectiveness, such as the use of attention mechanisms and graph neural networks. Compared to single-agent reinforcement learning methods, multi-agent frameworks attain more efficient performance in large-scale problems, but their policies are decentralized and may sacrifice system-level optimality (Mao et al., 2020).

To summarize, although demand-responsive fleet management has garnered sustained attention in recent years, the distinctive characteristics of intercity ride-pooling services still pose challenges to the direct application of existing frameworks. Specifically, for the routing problem of each ride-pooling vehicle, the existing literature has proposed comprehensive methods for effective computation. However, applying routing algorithms at the city cluster level can result in a vast search space, while introducing intercity lines can simplify the multi-vehicle pooled-ride routing problem at the city cluster level into separate vehicle routing within each line. For fleet dispatching, the aforementioned dispatching strategies for urban flexible bus systems and ride-hailing services are not directly applicable to fleet operations within a network comprising multiple cities. Intercity transport is based on a topology of city networks with relatively long duration for intercity trips, and relocation of idle vehicles is generally not cost-effective. The most formidable challenge resides in the coupling between long-term vehicle resource allocation and dynamic vehicle routing, which has received scant attention in existing research. Both intercity dispatching and dynamic pooling are difficult to solve using model-based approaches, while realistic fleet operations necessitate an efficient online framework to integrate the optimization of these two levels.

3. Modeling

3.1. Problem settings

In our study, we consider fleet operations in the context of intercity ride-pooling services within a city cluster. The cluster $\mathcal{U} = \{1, \ldots, n\}$ contains several interconnected cities, and the platform manages a fleet to fulfill intercity travel demands within the cluster in a pooled-ride manner. Multiple orders served by the same vehicle in one ride must originate from the same city

and travel to the same city. It is assumed in this study that orders with different origin cities or different destination cities cannot be simultaneously served by the same vehicle because visiting other cities en route will significantly increase the detour time for passengers on board. Under this assumption, the ride-pooling services within the city cluster can be split into services for a set of intercity lines $\mathcal{L} \subset \mathcal{U} \times \mathcal{U}$, in which each line $\ell \in \mathcal{L}$ starts from city $o^{\ell} \in \mathcal{U}$ and ends at city $d^{\ell} \in \mathcal{U}$ without visiting any intermediate city. Notice that ride-pooling service is not available for all pairs of cities within the cluster. The platform exclusively offers intercity lines between cities with sufficient demand for intercity travel, whereas the demand level between other city pairs can be so minimal that ride-pooling services cease to benefit from economies of scale. Furthermore, the lines in this study are different from the fixed lines of conventional intercity bus services. Conventional intercity buses travel along fixed routes on a fixed schedule, while vehicles in this study travel along fully flexible routes on each line. Instead of adhering to specific lines, each vehicle can be dynamically assigned to any line originating from its current city for upcoming rides.

From the *supply* perspective, the fleet dispatching is based on the directed graph $G(\mathcal{U}, \mathcal{L})$. We assume all vehicles in the fleet $\mathcal{K}$ are identical in capacity and speed, but they originate from different cities and commence their operations at different times to work continuously for no more than a preset time limit $\bar{W}$. The platform assigns orders to vehicles so that vehicles can pick up and deliver passengers in the planned sequence to complete a trip. Since intercity trips are relatively costly, relocation between cities is discouraged, and hence the platform only assigns idle vehicles to lines originating from the cities they are located. Furthermore, for the sake of long-distance transportation safety, drivers have to take a certain period of rest at the depot after finishing an intercity ride.

From the *demand* perspective, this study primarily focuses on providing services to 'on-demand' intercity trip orders, wherein passengers require service within a relatively short period after reservation (e.g., within 2 h). In urban ride-hailing services, the on-demand order represents that the passenger expects prompt service immediately upon placing the order. However, when travelers need intercity trips, they tend to formulate travel arrangements in advance and anticipate receiving services within a predetermined time window. We can usually observe significant reservations at the early stage of the days-long booking period (Tsai, 2020), partly owing to the fact that traditional intercity transit system faces challenges in handling demand fluctuations caused by 'on-demand' orders with short booking lead-time. In our study, necessary information of each order includes the number of passengers, origin, destination, hard time windows, and the selected line. Passengers may also have requirements for the latest arrival time at their destinations. Passengers are assumed to wait at the designated origins within the reserved time windows for pick-up, so they will not leave the system until they get picked up or the reserved time window ends. In other words, we do not model the behavior of passengers canceling orders due to excessive waiting time in this study, because passengers can express their travel preferences by personalizing the time windows. Actually, our framework can be easily extended to address the problems with soft time window constraints or order cancellations. In our study, a feasible passenger–vehicle matching necessitates a compatible vehicle arriving at the origin and destination location before the respective latest time. The matching process is executed at short intervals for each line respectively. Once the order is assigned to certain vehicle during one matching process, the passenger will receive the information of the matched vehicle, so neither the driver nor the passenger can cancel the matched order. If no feasible match can be made before the latest pick-up time, the platform will forfeit this order.

3.2. Modeling framework

In this subsection, we briefly introduce the overall decision-making framework in fleet dispatching and vehicle routing of the intercity ride-pooling services. We discretize the daily time domain into T dispatching horizons of equal duration and the set of all horizons is denoted as $\mathcal{T}$. Orders of each line emerge following a specific stochastic process throughout the day, which are added to the matching pool of the respective line in real-time. The fleet operations within a dispatching horizon $t \in \mathcal{T} = \{0, 1, \ldots, T-1\}$ contain two levels: *first*, the platform assigns idle vehicles to different lines or reserves them at their current cities; *second*, the platform plans a route for each vehicle to fulfill orders in the matching pool for each line respectively. These two levels of fleet operations are executed at different temporal resolutions. The first-level idle fleet dispatching decisions are made once in a dispatching horizon, while the matching and routing are updated more frequently to improve order response rate and avoid order loss. Therefore, the platform splits each dispatching horizon into several matching intervals with equal lengths and updates the planned routes of all vehicles en route at each matching interval.

The relationship between dispatching horizons and matching intervals is further illustrated in Fig. 2(a). At the beginning of a dispatching horizon, the platform develops dispatching decisions for the current horizon based on the global supply and demand information. A relatively short horizon may lead to no idle vehicles for assignment, while an excessively long horizon could compromise the effectiveness of the dispatching. Based on realistic operational data and numerical tests, we set the horizon duration as 20 min in this study. A horizon is composed of a set of batch-matching processes. The first batch matching is executed instantly after the assignment of idle vehicles, which can provide initial routes for vacant vehicles to fulfill orders, and the other batch matching processes in a horizon are executed at the beginning of each matching interval. The matching interval can be within the range of 0.5 to 2 min in practice.

Based on this operational framework, the intercity fleet operations problem can be regarded as a two-level embedded fleet dispatching and vehicle routing problem. The vehicle routing problem at the lower level is based on the decisions of assigning vehicles to lines derived from the upper level, and the objective function value of the fleet dispatching problem at the upper level is obtained by the routing results of the lower-level problem, as illustrated in Fig. 2(b).

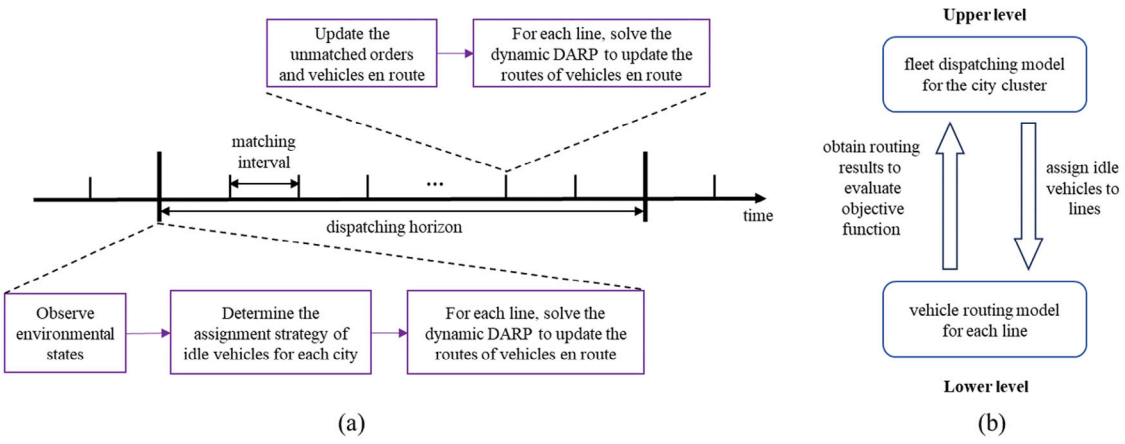

Fig. 2. (a) Illustration of the dispatching horizons and matching intervals. (b) Interaction between two levels.

3.3. The fleet dispatching model

The fleet dispatching problem at the upper level of the modeling framework can be described as a stochastic dynamic resource allocation problem (SDRAP) (Godfrey and Powell, 2002). Let $\hat{\mathcal{K}}_u^t$ denote the set of vehicles that enter the system to start working at the beginning of horizon $t \in \mathcal{T}$ at city $u \in \mathcal{U}$, $\tilde{\mathcal{K}}_u^t$ denote the set of vehicles that exit the system due to reaching limited work duration at the beginning of horizon t at city u, $\check{\mathcal{K}}_u^t$ denote the set of arrived vehicles at horizon t at city u, $\bar{\mathcal{K}}_u^t$ denote the set of reserved vehicles at horizon t at city u. We assume that all drivers have to take a rest of at least τ_r horizons after an intercity trip. Then all available vehicle resources for assignment at horizon t at city u is $\mathcal{K}_u^t = \hat{\mathcal{K}}_u^t \cup \check{\mathcal{K}}_u^{t-\tau_r-1} \cup \bar{\mathcal{K}}_u^{t-1} \setminus \tilde{\mathcal{K}}_u^t$. At horizon t, we determine the spatial allocation of idle vehicles $\mathcal{N}^t = (\mathcal{N}_{uv}^t)_{(u,v)\in\mathcal{L}}$, where $\mathcal{N}_{uv}^t$ is the set of vehicles dispatched from city u to city v at horizon t. Then the dynamics of all available vehicles in each city over horizons can be represented by the flow conservation constraints (1)–(4), where τ_{uv} in constraint (4) denotes the expected travel time between city u and city v. Additionally, it is anticipated that vehicles will return to their initial city once their daily working time reaches the designated limit $\bar{W}$, as shown in constraint (5).

$$\hat{\mathcal{K}}_u^t = \Big(\bigcup_{v:(u,v)\in\mathcal{L}} \mathcal{N}_{uv}^t\Big) \cup \bar{\mathcal{K}}_u^t \qquad t = 0, \forall u \in \mathcal{U} \tag{1}$$

$$\hat{\mathcal{K}}_u^t \cup \bar{\mathcal{K}}_u^{t-1} \setminus \tilde{\mathcal{K}}_u^t = \Big(\bigcup_{v:(u,v)\in\mathcal{L}} \mathcal{N}_{uv}^t\Big) \cup \bar{\mathcal{K}}_u^t \qquad \forall t \in \{1,\dots,\tau_r\}, u \in \mathcal{U} \tag{2}$$

$$\hat{\mathcal{K}}_u^t \cup \check{\mathcal{K}}_u^{t-\tau_r-1} \cup \bar{\mathcal{K}}_u^{t-1} \setminus \tilde{\mathcal{K}}_u^t = \Big(\bigcup_{v:(u,v)\in\mathcal{L}} \mathcal{N}_{uv}^t\Big) \cup \bar{\mathcal{K}}_u^t \qquad \forall t \in \mathcal{T} \setminus \{0,\dots,\tau_r\}, u \in \mathcal{U} \tag{3}$$

$$\check{\mathcal{K}}_v^t = \bigcup_{u\in\mathcal{U}:\tau_{uv}=\tilde{t}} \mathcal{N}_{uv}^{t-\tilde{t}} \qquad \forall t \in \mathcal{T}, \tilde{t} \in \{0,1,\dots,t\}, v \in \mathcal{U} \tag{4}$$

$$\tilde{\mathcal{K}}_u^t = \hat{\mathcal{K}}_u^{t-\bar{W}} \qquad \forall t \in \{\bar{W},\dots,T-1\}, u \in \mathcal{U} \tag{5}$$

Based on the system dynamics elaborated above, we model this stochastic dynamic vehicle resource allocation problem as a Markov decision process (Xu et al., 2018b; Tang et al., 2020).

Agents: The fleet of available vehicles within a city is perceived as an agent. There exist $|\mathcal{U}|$ agents because the idle fleet within each city makes autonomous vehicle-line assignment decisions. The components of this MDP formulation $M \triangleq \langle \mathbf{S}, \mathbf{A}, P, R \rangle$ are defined as follows.

States: The system state is captured by the tuple $\mathbf{S} = \{\mathbf{S}_h, \mathbf{S}_s, \mathbf{S}_d\}$. $\mathbf{S}_h$ indicates the current horizon of a day. $\mathbf{S}_s = \{\mathbf{S}_s^u\}_{u\in\mathcal{U}} = \{((\varphi_{(0)}^{uv})_v, \varphi_0^u, \varphi_1^u, \varphi_2^u, \dots \varphi_{\tau_s}^u)\}_{u\in\mathcal{U}}$ describes the supply states of available vehicle resource in each city. In the first term, $\varphi_{(0)}^{uv}$ is the total number of empty seats in vehicles currently in service of line (u,v), the vector $(\varphi_{(0)}^{uv})_v$ describes currently available seat resources in different lines originating from city u without newly dispatching any vacant vehicle. $\varphi_i^u = \left|\mathcal{K}_u^{t+i}\right|$ is the accumulated number of vacant vehicles for assignment in the future ith horizon. $\mathbf{S}_d = \{\mathbf{S}_d^{uv}\}_{(u,v)\in\mathcal{L}} = \{(\omega_{(0)}^{uv}, \omega_0^{uv}, \hat{\omega}_1^{uv}, \hat{\omega}_2^{uv}, \dots \hat{\omega}_{\tau_d}^{uv})\}_{(u,v)\in\mathcal{L}}$ describes the demand states in each line. $\omega_{(0)}^{uv}$ is the number of orders in line (u,v) that will expire if not served within the current horizon. ω_0^{uv} is the number of orders in the matching pool when making the dispatching decisions, while $\hat{\omega}_i^{uv}$ is the estimated number of newly emerging orders in line (u,v) in the future ith horizon, which can be obtained from the historical average value at the same horizon. We set $\tau_s = 3$ and $\tau_d = 5$ in subsequent numerical experiments in Section 5. The states not only specify the demand and supply information in the current horizon which will influence the instant reward but also describe the potential demand and supply dynamics in several future horizons, which is helpful to capture the spatial–temporal demand and supply fluctuation. Notice that more detailed attributes of supply and demand, such as the specific locations of vehicles and orders, are not included in $\mathbf{S}$, since we focus on approximating

the effectiveness of resource allocation in the upper level of the framework and leave the evaluation of detailed information to the lower-level routing problem.

Actions: $\mathbf{A} = \{\mathbf{A}^u\}_{u\in\mathcal{U}}$ represents the joint actions of all agents. In horizon t, the assignment decision of the idle fleet within city $u \in \mathcal{U}$ is a non-negative integer vector $\mathbf{A}_t^u = ((|\mathcal{N}_{uv}^t|)_{v:(u,v)\in\mathcal{L}}, |\bar{\mathcal{K}}_u^t|)$, indicating the number of vehicles assigned to all lines originating from city u, and the number of vehicles reserved at the current city. Here, the action vector $\{\mathbf{A}_t^u\}_{u\in\mathcal{U}}$ has to satisfy the flow conservation constraints (1) to (5). The summation of the elements within vector $\mathbf{A}_t^u$ cannot exceed the number of available vehicles within city u at the beginning of this horizon, which is $|\mathcal{K}_u^t| = |\hat{\mathcal{K}}_u^t| + |\check{\mathcal{K}}_u^{t-\tau_r-1}| + |\bar{\mathcal{K}}_u^{t-1}| - |\bar{\mathcal{K}}_u^t|$ and is denoted by φ_0^u in the supply state tuple $\{\mathbf{S}_s^u\}_{u\in\mathcal{U}}$. The number of available vehicles $|\mathcal{K}_u^t|$ varies over horizons, which indicates that the feasible action space of each agent is state-dependent.

State transitions: The state transition probability $\mathcal{P}(s'|s,a) = \mathcal{P}(\mathbf{S}_{t+1} = s'|\mathbf{S}_t = s, \mathbf{A}_t = a)$ explains the probability from state s to next state s' based on the selected action a. During one transition, the dispatching action $\{\mathbf{A}^u\}$ is executed to update the subsequent system states. The next horizon indicator $\mathbf{S}_h$ increases by one. The next supply state $\mathbf{S}_s$ is constructed following the constraints (1) to (5). The demand state $\mathbf{S}_d$ is updated by the matching results within the current horizon and the newly emerging orders. Since the supply state $\mathbf{S}_s$ consists of both the number of current idle vehicles ($\{(\varphi_{(0)}^{uv})_v\}$ and $\{\varphi_0^u\}$) and the number of vehicles that will be idle in future several horizons ($\{\varphi_i^u\}$), the distribution of the next supply states can be determined only based on current supply states and actions. Thus, the fleet dispatching problem satisfies the Markov property under our definition on states and actions, which is $\mathcal{P}(\mathbf{S}_{t+1} = s_{t+1}|\mathbf{S}_0 = s_0, \mathbf{A}_0 = a_0, \ldots, \mathbf{S}_t = s_t, \mathbf{A}_t = a_t) = \mathcal{P}(\mathbf{S}_{t+1} = s_{t+1}|\mathbf{S}_t = s_t, \mathbf{A}_t = a_t)$.

Rewards: At dispatching horizon t, the platform determines the global dispatching decision $\mathbf{A}_t$ and receives an immediate reward $R_t = r(\mathbf{S}_t, \mathbf{A}_t)$ by assigning idle vehicles to intercity lines following the instruction of $\mathbf{A}_t$. The immediate reward of the whole platform is the summation of the immediate reward of each agent respectively, i.e. $R_t = \sum_{u\in\mathcal{U}} R_t^u$. The immediate reward is defined as the total profits earned by vehicles dispatched at this horizon, minus the penalties due to lost orders during this horizon.

To calculate the immediate rewards in the simulator environment and in practice, we need to solve the dynamic vehicle routing problems based on the system states and dispatching decisions $\mathbf{A}_t$. Let R^{uv} denote the one-way trip fare of line $(u,v) \in \mathcal{L}$, C denote the vehicle cost per distance, and p_e denote the penalty rate for losing one passenger. Then, the rewards for the agent representing fleet within city u at horizon t are calculated as follows.

$$R_t^u = \sum_{v:(u,v)\in\mathcal{L}} \Big(\sum_{k\in\mathcal{N}_{uv}^t} R^{uv} s_k^t - \sum_{k\in\mathcal{N}_{uv}^t} C\delta_k^t - p_e R^{uv} \epsilon_{uv}^t \Big) \tag{6}$$

The first part of the formula corresponds to the whole-trip revenue of vehicles dispatched at horizon t, where s_k^t is the number of passengers that vehicle k picked in this trip. The second part corresponds to the traveling costs of the whole trip of all vehicles in $\mathcal{N}_{uv}^t$, where δ_k^t is the total trip distance of vehicle k. The last part corresponds to the order loss penalty, where ϵ_{uv}^t is the number of lost passengers in line (u,v) during horizon t.

In the fleet dispatching problem, we seek to obtain an optimal vehicle allocation policy π, which is a mapping from states $\mathbf{S}$ and actions $\mathbf{A}$ to action probability distribution $\pi(\mathbf{A}|\mathbf{S})$. The objective of policy optimization is to maximize the expected system return, which is the expected discounted total rewards of all possible trajectories during the T horizons conditional on the initial states $\mathbf{S}_0$. The overall objective function is shown as follows.

$$\max_{\pi} \mathbb{E}_{\mathbf{A}\sim\pi(\cdot|\mathbf{S})}[\sum_{t=0}^{T-1} \gamma^t r(\mathbf{S}_t, \mathbf{A}_t)|\mathbf{S}_0]$$

where $\mathbf{A} \sim \pi(\cdot|\mathbf{S})$ represents the action $\mathbf{A}$ is obtained by sampling from a stochastic policy π based on the state $\mathbf{S}$. For each horizon, the reward function $R_t = r(\mathbf{S}_t, \mathbf{A}_t)$ is determined by solving the lower-level problem based on the decision $\mathcal{N}_{uv}^t$, which makes it difficult to solve the dynamic resource allocation problem embedded with dynamic DARP using model-based approximate dynamic programming approaches. Thus, we take advantage of reinforcement learning techniques to tackle the large-scale fleet dispatching problem. A more detailed description of our reinforcement learning model is provided in Section 4.1.

Additionally, although we assume vehicles are identical in capacity and speed, they originate from various cities and commence work at different times, so vehicles are non-homogeneous in the rest of available work time at any given time and have different preferences on the final depots. To focus on mitigating supply–demand imbalances, our MDP formulation is based on the quantities of vehicles in each area. Consequently, the direct decisions are the numbers of vehicles dispatched to each line. However, when assigning certain vehicles in practice, it is necessary to also consider their final depots. Therefore, we develop a mathematical programming-based approach to map quantity-based allocations into actual vehicle assignment schemes considering the heterogeneity of vehicles, which will be elaborated upon in Section 4.1.3.

3.4. The vehicle routing model

The lower level of the framework optimizes the multi-vehicle pooled-ride routing problem for each line separately at each matching interval. For each line, the solution of the dynamic DARP consists of the complete routes of all vehicles, including vehicles en route and newly dispatched vehicles. Each route originates from the current location of the vehicle and ends at the depot of the destination city. Thus, routes planned at earlier matching intervals can be updated to serve newly emerging orders, but the vehicle-order matching obtained in earlier intervals cannot be violated. Vehicles travel along the currently planned routes at the current matching interval and update their routes based on new routing results at the beginning of the next interval. This process is repeated at each matching interval to enable online vehicle routing in operations.

We formulate the online routing problem of the line (u, v) in a matching interval at the dispatching horizon $t \in \mathcal{T}$ as a mixed-integer linear program (MILP). Let $\mathcal{K}_{uv}$ denote the set of vehicles that are currently in a ride of this line, which also includes newly assigned vehicles $\mathcal{N}_{uv}^t$ if this is the first matching after vehicle dispatching by the upper-level problem. The order set $\mathcal{O}_{uv}$ includes all orders in the matching pool and the orders already matched by vehicles en route.

Let v_k be the location of vehicle $k \in \mathcal{K}_{uv}$ at the beginning of this matching interval. For vehicle k, this trip will end at the depot E in the destination city, and we set the latest arrival time $\check{U}_k$ based on its departure time for the current trip. Setting the latest time to arrive at the depot ensures that the total duration of the vehicle's route does not exceed a reasonable range, preventing the situation where a vehicle waits excessively for an order, which is crucial for improving the utilization of vehicle resources. For order $p \in \mathcal{O}_{uv}$, the number of passengers is denoted as n_p, the origin is denoted as s_p with time window $[\check{L}_p, \check{U}_p]$, the destination is denoted as f_p with time window $[\hat{L}_p, \hat{U}_p]$. Notice that the vehicle-order matching obtained in earlier matching intervals cannot be violated. Thus, orders in $\mathcal{O}_{uv}$ can be classified into three categories. The first category $\mathcal{O}_{uv}^1$ are orders that have already been picked, and we use k_p to denote the vehicle that has served order p. The second category $\mathcal{O}_{uv}^2$ are orders that have already been matched but not picked, and we use k_p to denote the vehicle that has matched order p. The third category $\mathcal{O}_{uv}^3$ are orders still in the matching pool. Since orders in $\mathcal{O}_{uv}^1 \cup \mathcal{O}_{uv}^2$ have been matched in feasible routing solutions of the last few intervals, there must exist a feasible solution to complete the service of these orders at the current interval.

The routing problem is based on the complete graph $G(\mathcal{V}, \mathcal{A})$. The vertices $\mathcal{V} = \mathcal{V}_k \cup \mathcal{V}_f \cup \mathcal{V}_s \cup \{E\}$, where $\mathcal{V}_k = \{v_k, \forall k \in \mathcal{K}_{uv}\}$ indicates the locations of all vehicles at the beginning of the current interval, $\mathcal{V}_f = \{f_p, \forall p \in \mathcal{O}_{uv}\}$ indicates the destination nodes of all orders, $\mathcal{V}_s = \{s_p, \forall p \in \mathcal{O}_{uv}^2 \cup \mathcal{O}_{uv}^3\}$ indicates the origin nodes of the unserved orders, and E indicates the depot in the destination city. Let R^{uv} be the one-way trip fare on line (u, v), D_{ij} be the distance of arc $(i, j) \in \mathcal{A}$, C be the travel cost per distance. The objective of the routing problem is to maximize the revenue of all vehicles, subtracted by traveling costs from their current locations to the depot in city v. The objective function is shown as follows.

$$\max \sum_{p \in \mathcal{O}_{uv}} \sum_{k \in \mathcal{K}_{uv}} R^{uv} n_p d_{pk} - C \sum_{(i,j) \in \mathcal{A}} \sum_{k \in \mathcal{K}_{uv}} y_{ij}^k D_{ij}$$

Decision variables in the vehicle routing model are given by x_{ij}^{pk}, y_{ij}^k, d_{pk} and u_{kj}. The first three kinds of variables are binary variables, where x_{ij}^{pk} indicates whether vehicle k passes arc (i, j) with passengers of order p, y_{ij}^k indicates whether vehicle k passes arc (i, j), d_{pk} indicates whether vehicle k serves order p, u_{kj} is a positive variable indicating the time for vehicle k to arrive at node j. The constraints can be formulated as follows.

$$\sum_{j \in \mathcal{V} \setminus \{i\}} \sum_{k \in \mathcal{K}_{uv}} y_{ij}^k \le 1 \qquad \forall i \in \mathcal{V} \tag{7}$$

$$\sum_{j \in \mathcal{V} \setminus \{i\}} y_{ji}^k - \sum_{j' \in \mathcal{V} \setminus \{i\}} y_{ij'}^k = 0 \qquad \forall i \in \mathcal{V}_f \cup \mathcal{V}_s, k \in \mathcal{K}_{uv} \tag{8}$$

$$\sum_{j \in \mathcal{V} \setminus \{v_k\}} y_{v_k j}^k = \sum_{j \in \mathcal{V} \setminus \{E\}} y_{jE}^k = 1 \qquad \forall k \in \mathcal{K}_{uv} \tag{9}$$

$$d_{pk_p} = 1 \qquad \forall p \in \mathcal{O}_{uv}^1 \cup \mathcal{O}_{uv}^2 \tag{10}$$

$$\sum_{k \in \mathcal{K}_{uv}} d_{pk} \le 1 \qquad \forall p \in \mathcal{O}_{uv}^3 \tag{11}$$

$$\sum_{j \in \mathcal{V} \setminus \{f_p\}} x_{jf_p}^{pk} - \sum_{j \in \mathcal{V} \setminus \{f_p\}} x_{f_p j}^{pk} = d_{pk} \qquad \forall p \in \mathcal{O}_{uv}, k \in \mathcal{K}_{uv} \tag{12}$$

$$\sum_{j \in \mathcal{V} \setminus \{s_p\}} x_{s_p j}^{pk} - \sum_{j \in \mathcal{V} \setminus \{s_p\}} x_{j s_p}^{pk} = d_{pk} \qquad \forall p \in \mathcal{O}_{uv}^2 \cup \mathcal{O}_{uv}^3, k \in \mathcal{K}_{uv} \tag{13}$$

$$\sum_{j \in \mathcal{V} \setminus \{i\}} x_{ij}^{pk} - \sum_{j' \in \mathcal{V} \setminus \{i\}} x_{j'i}^{pk} = 0 \qquad \forall p \in \mathcal{O}_{uv}, i \in \mathcal{V}_f \cup \mathcal{V}_s \setminus \{s_p, f_p\}, k \in \mathcal{K}_{uv} \tag{14}$$

$$x_{ij}^{pk} \le y_{ij}^k \qquad \forall (i, j) \in \mathcal{A}, k \in \mathcal{K}_{uv} \tag{15}$$

$$\sum_{p \in \mathcal{O}_{uv}} x_{ij}^{pk} n_p \le W y_{ij}^k \qquad \forall (i, j) \in \mathcal{A}, k \in \mathcal{K}_{uv} \tag{16}$$

$$u_{kv_k} = t_0 \qquad \forall k \in \mathcal{K}_{uv} \tag{17}$$

$$u_{kj} - u_{ki} \ge D_{ij}/s - M(1 - y_{ij}^k) \qquad \forall (i, j) \in \mathcal{A}, k \in \mathcal{K}_{uv} \tag{18}$$

$$\check{L}_p d_{pk} \le u_{ks_p} \le \check{U}_p d_{pk} \qquad \forall p \in \mathcal{O}_{uv}^2 \cup \mathcal{O}_{uv}^3, k \in \mathcal{K}_{uv} \tag{19}$$

$$\hat{L}_p d_{pk} \le u_{kf_p} \le \hat{U}_p d_{pk} \qquad \forall p \in \mathcal{O}_{uv}, k \in \mathcal{K}_{uv} \tag{20}$$

$$u_{kE} \le \check{U}_k \qquad \forall k \in \mathcal{K}_{uv} \tag{21}$$

$$x_{ij}^{pk} \in \{0, 1\}, y_{ij}^k \in \{0, 1\}, d_{pk} \in \{0, 1\} \qquad \forall p \in \mathcal{O}_{uv}, (i, j) \in \mathcal{A}, k \in \mathcal{K}_{uv} \tag{22}$$

Constraint (7) ensures that each node can be passed at most once, and each arc is passed by at most one vehicle. If more than two vehicles pass the same arc, there must exist a shorter route for one of them. Constraints (8) and (9) are flow conservation

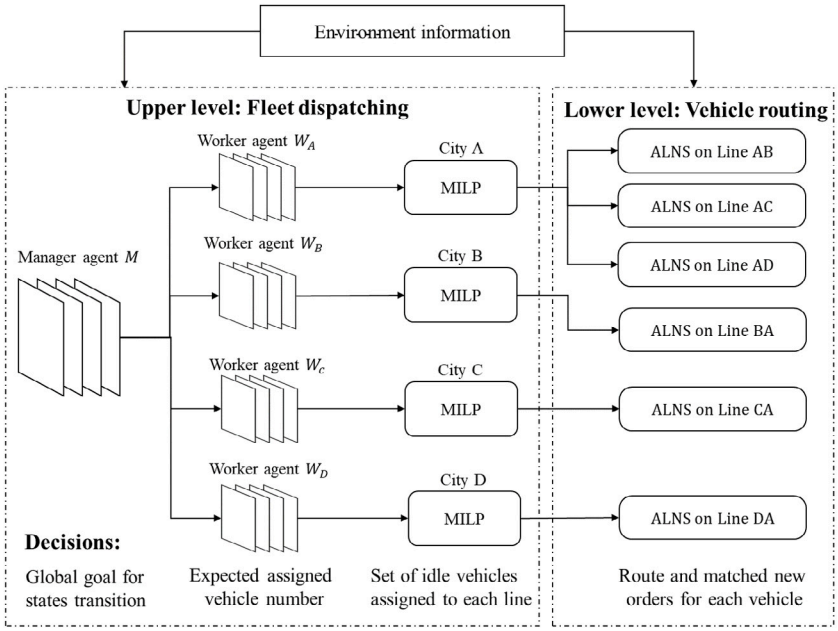

Fig. 3. The fleet operational decision process for a network with one central city A and three surrounding cities B, C, and D.

constraints that the route of any vehicle in $\mathcal{K}_{uv}$ originates from its current location and ends the depot in city v. Constraints (10) and (11) indicate the matching between vehicles and orders. The matching of orders in $\mathcal{O}^1_{uv}$ and $\mathcal{O}^2_{uv}$ are determined before this matching interval, while each order in $\mathcal{O}^3_{uv}$ can be matched to at most one vehicle en route. Constraints from (12) to (15) represent that passengers are picked by the matched vehicle at the origin node and delivered to the destination. Constraint (16) ensures that the number of passengers in the vehicle will not exceed the capacity W. Constraints from (17) to (21) determine the time for vehicles to serve each node, where t_0 represents the current time, s represents the vehicle speed, and M is a very large positive value in constraint (18). Based on the assumptions of hard time windows, vehicles have to serve the matched orders within the requested time windows. Vehicles may wait at the node if they arrive before the time window, while late arrivals lead to infeasible solutions.

The multi-vehicle pooled-ride routing problem is NP-hard because it contains the traveling salesman problem as a special case (Ropke and Pisinger, 2006). Since we have to solve this problem repeatedly at every matching interval, we leverage the ALNS heuristic to obtain a near-optimal solution efficiently. The detailed heuristic algorithm is introduced in Section 4.2.

4. Solution methodologies

To tackle the challenges introduced in Section 3, we propose a two-level reinforcement learning framework to implement dynamic fleet dispatching and online vehicle routing on intercity ride-pooling services. A novel multi-agent hierarchical reinforcement learning method is adopted in the upper level of the framework to optimize the spatial–temporal allocation of vehicle resources by assigning idle vehicles to lines. The lower level of the framework uses the ALNS heuristic to solve the proposed online multi-vehicle pooled-ride routing problem efficiently and effectively based on the assignment decision of the upper level.

Before delving into the specifics of the model, we provide an example that outlines the workflow across two levels of the framework. Consider the ride-pooling service provided in a city cluster with one central city A and three surrounding cities B, C, and D. Six intercity lines are available, including AB, BA, AC, CA, AD and DA. The corresponding operational decision process at the beginning of a dispatching horizon can be illustrated as shown in Fig. 3. Two levels of the framework observe distinct environmental information. The upper-level model employs a manager–worker hierarchical structure. The manager derives a global goal on state transitions. Each worker agent represents the available fleet within a city and determines the expected number of vehicles dispatched to each line based on the manager's instruction and local observations. A mixed integer linear program is executed independently for each city, mapping the assigned vehicle numbers to the specific vehicle assignments. Then the lower level of the framework utilizes the ALNS heuristic to address the routing problem for each line separately at each matching interval so that vehicles can serve the matched orders following the sequence planned.

4.1. Upper level: Multi-agent feudal networks

The upper level of the framework is expected to learn the optimal vehicle dispatching policy for each city within the city cluster. In our study, the fleet of idle vehicles within each city is regarded as an agent, multiple agents operate collaboratively to improve

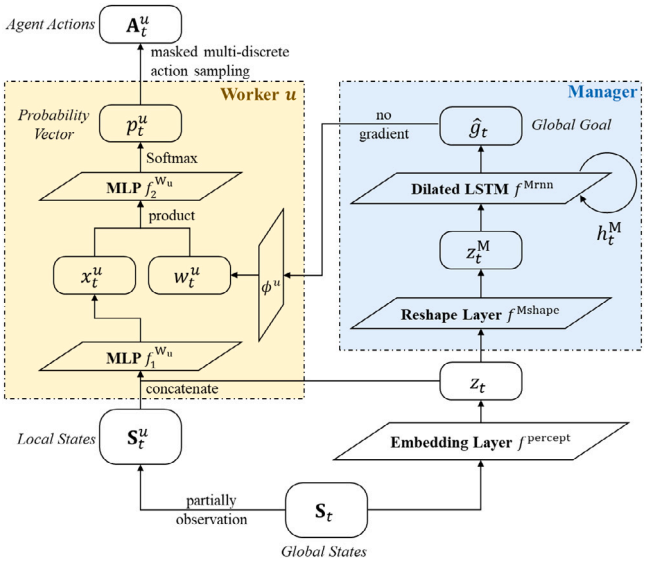

Fig. 4. Actor network structure of MFuN.

the system throughput. Thus, we use a multi-agent reinforcement learning framework at the upper level. MARL has been widely used in fleet operations in ride-sourcing platforms, which performs well in scalability in large-scale problems (Xu et al., 2018a; Guo and Xu, 2020; Jiao et al., 2021; Kullman et al., 2022). However, MARL has a prominent drawback that the training process may be non-stationary when each agent updates its own policy. The system may fail to converge to the global optimum because each agent makes decisions based on its own utility function and local observation. Conventional reinforcement learning methods, such as Q-learning or policy gradient methods, usually exhibit high variance in multi-agent environments where coordination of agents is required (Lowe et al., 2017).

To handle the problem aroused by decentralized training, we propose a Multi-agent Feudal Network (MFuN) framework to enhance dispatching decision coordination among agents. Feudal reinforcement learning (FRL) was first proposed by Dayan and Hinton (1992). Vezhnevets et al. (2017) introduced Feudal Networks (FuN) to extend conventional FRL to the form of deep neural networks, which perform well in long timescale credit assignment, especially in environments with sparse reward signals. A manager module and a worker module are constructed in FuN, where the manager sets goals as directions for the actions of the worker. The goals can be interpreted as an expected objective for global state transition in future horizons. The worker receives intrinsic rewards for actions consistent with goals. Multi-agent settings are introduced by Ahilan and Dayan (2019) to set multiple agents as workers. The shared goals can be used to encourage the cooperation of simultaneously-acting agents. This hierarchical framework is highly compatible with the fleet dispatching problem within the city cluster. The vehicle assignment decisions in each city are relatively independent during the current horizon, but the significant influence of these actions on the supply states of other agents propagates throughout the city network in subsequent horizons, thereby influencing the overall system profit. Under the guidance of a global manager, all agents are expected to cooperate to improve the effectiveness of vehicle resource allocation to mitigate the imbalance between demand and supply.

4.1.1. Model structure

In this section, we introduce the model structure of MFuN, which maps states $\mathbf{S}_t = \{\mathbf{S}_{h,t}, \mathbf{S}_{s,t}, \mathbf{S}_{d,t}\}$ to actions $\mathbf{A}_t = \{((|\mathcal{N}_{uv}^t|)_{v:(u,v)\in\mathcal{L}}, |\bar{\mathcal{K}}_u^t|)\}_{u:u\in\mathcal{U}}$ as proposed in Section 3. The model structure of MFuN is shown as Fig. 4. The model contains a manager and several worker agents. The manager and worker agents share the embedding layer f^{percept} to obtain intermediate representation z_t of global states $\mathbf{S}_t$. We use the same module for the manager M as proposed by Vezhnevets et al. (2017), which contains a reshape layer f^{Mshape} and a recurrent neural network architecture f^{Mrnn}. The output of the manager is a goal vector $\hat{g}_t$, which is obtained by a dilated LSTM model with input as reshaped z_t and internal states h_t^{M}. The dilated LSTM model is composed of r separate groups of sub-states, where r is a dilation radius. At dispatching horizon t, only the $t\%r$ sub-states are updated, where % denotes the modulo operation, and the output is pooled across the previous c outputs to accomplish smooth variation. By updating only part of the whole states at each time, the dilated LSTM model can preserve the memories for longer periods without missing any input experience. The forward dynamics of the manager policy networks are given by Eqs. (23) and (24).

$$z_t^{\text{M}} = f^{\text{Mshape}}(z_t) \tag{23}$$

$$h_t^{\text{M}}, \hat{g}_t = f^{\text{Mrnn}}(z_t^{\text{M}}, h_{t-1}^{\text{M}}) \tag{24}$$

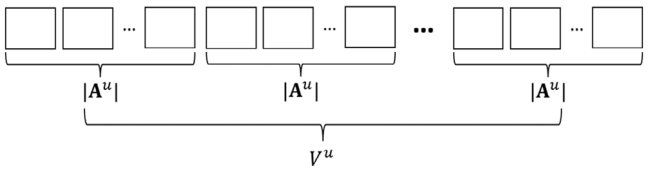

Fig. 5. The output of worker agent u.

The module of a worker agent W_u is composed of two multi-layer perceptions (MLP) $f_1^{\mathrm{W}_u}$ and $f_2^{\mathrm{W}_u}$. Parameters in each worker module are not shared since they may differ in network scales. The mathematical formulations of the policy networks of worker agent W_u are given by Eqs. (25), (26), and (27).

$$w_t^u = \phi^u\Big(\sum_{i=t-c}^{t} g_i\Big); g_t = \hat{g}_t / \|\hat{g}_t\| \tag{25}$$

$$x_t^u = f_1^{\mathrm{W}_u}(z_t \oplus \mathbf{S}_t^u) \tag{26}$$

$$p_t^u = \mathrm{Softmax}(f_2^{\mathrm{W}_u}(w_t^u x_t^u)) \tag{27}$$

For each worker agent W_u, the first MLP $f_1^{\mathrm{W}_u}$ takes the concatenation of global state representation z_t and local observation $\mathbf{S}_t^u$ as input. Here, the local observation of agent W_u is composed of local supply and demand information. Similar to the definition in Section 3.3, the local supply states are $\mathbf{S}_s^u = ((\varphi_{(0)}^{uv})_{v:(u,v)\in\mathcal{L}}, \varphi_0^u, \varphi_1^u, \varphi_2^u, \dots \varphi_{\tau_s}^u)$, including the number of currently available seat resources on the line originating from city u and idle vehicle resources at future horizons. The local demand states are $\mathbf{S}_d^u = \{\mathbf{S}_d^{uv}\}_{v:(u,v)\in\mathcal{L}} = \{(\omega_{(0)}^{uv}, \omega_0^{uv}, \hat{\omega}_1^{uv}, \hat{\omega}_2^{uv}, \dots \hat{\omega}_{\tau_d}^{uv})\}_{v:(u,v)\in\mathcal{L}}$, which summarizes the number of various unserved orders in each line originating from city u. Each worker agent is also influenced by the current global goal g_t, which is a normalized vector of the output of the manager output $\hat{g}_t$. As shown in Eq. (25), the normalized system goal g_t is then summed up with the last c normalized goals, and projected by an agent-specified linear transformation ϕ^u into a local goal vector w_t^u, which reflects the comprehension of the global goals by worker agent W_u based on their own features. The local goal vector w_t^u is then combined with local intermediate representation x_t^u via product to obtain logits vector $f_2^{\mathrm{W}_u}(w_t^u x_t^u)$. We decompose the logits vector into several parts to correspond to the multiple dimensions of actions. Each part comprises multiple real values representing the scores for different options within a specific action dimension. Subsequently, we apply the Softmax function separately to each part of the logits vector, thereby converting the real-valued scores over action options into normalized probability values for selecting the corresponding actions. Therefore, the probability vector p_t^u also consists of multiple components, each representing a probability distribution over the action space within a specific dimension of actions. The values in each component ensure a sum of 1 respectively, indicating a valid probability distribution. Finally, we can sample each dimension of the action from p_t^u independently, and then integrate each component to form the complete action vector $\mathbf{A}_t^{\mathbf{u}}$.

So far, we have presented a thorough overview of the network structure. However, there remains a question of how to ensure the feasibility of the actions sampled. In the MDP formulation proposed in Section 3.3, the actions of the agent representing idle fleet within city u are defined as $\mathbf{A}_t^u = ((|\mathcal{N}_{uv}^t|)_{v:(u,v)\in\mathcal{L}}, |\tilde{\mathcal{K}}_u^t|)$, which is composed of $|\{v\}_{(u,v)\in\mathcal{L}}| + 1$ dimensions. It is worth noting that the actions in different dimensions are not mutually independent. The summation of numbers of vehicles assigned to different lines cannot exceed $|\mathcal{K}_u^t| = |\hat{\mathcal{K}}_u^t| + |\check{\mathcal{K}}_u^{t-\tau_r-1}| + |\tilde{\mathcal{K}}_u^{t-1}| - |\tilde{\mathcal{K}}_u^t|$, which is the number of all idle vehicles in city u that are available for assignment. Thus, we cannot use Softmax to directly determine the probability distribution over the number of vehicles dispatched to each line respectively. Instead, we use a virtual fleet of size V^u to represent all available vehicle resources and determine the dispatching action of each virtual vehicle, where V^u is a predetermined constant. Each virtual vehicle selects a destination city from $|\mathbf{A}^u| = |\{v\}_{(u,v)\in\mathcal{L}}| + 1$ options, including city u and its neighbors. Hence, as shown in Fig. 5, the output p_t^u consists of V^u components, each comprising a normalized probability distribution over $|\mathbf{A}^u|$ action options. When sampling for actions, we use decision masks to handle the situations of $|\mathcal{K}_u^t| < V^u$, which represents the number of idle vehicles in reality is smaller than the predefined virtual fleet size. In such cases, the actions of the last $V^u - |\mathcal{K}_u^t|$ virtual vehicles will be masked. Then, the sampled actions of the first $\min\{|\mathcal{K}_u^t|, V_u\}$ virtual vehicles are recorded. By summing up the number of virtual vehicles dispatched to each line, we can clarify each element of $\mathbf{A}_t^u$. If the number of idle vehicles in reality exceeds V^u, the $|\mathcal{K}_u^t| - V^u$ excess vehicles will be kept in the current city at the current horizon. The actions of different virtual vehicles can be treated as mutually independent, so our method can always guarantee the feasibility of the sampled actions.

The selection of the quantity V^u is crucial for the model's performance. If V^u is so large that the number of available vehicles within city u is always less than V^u, then many components of the model outputs will always be blocked and unable to map to actions. This will significantly impact the convergence rate of the neural network. On the other hand, if V^u is so small that the number of available vehicles within the city often exceeds V^u, then numerous idle vehicles cannot be timely dispatched, leading to a waste of vehicle resources. Thus, we usually set the value of V^u close to the median between the peak supply and the mean.

4.1.2. Learning methods

We train the manager and workers with the Advantage Actor–Critic method, which learns the Actor parameters θ and the Critic parameters ψ for all agents respectively. Many value-based learning methods, such as DQN and QMix, do not support multi-discrete

action space, while the Advantage Actor–Critic method can still perform well. Notice that no gradients are propagated between workers and the manager, so the manager has to train its parameters autonomously to output latent goals for maximizing the system's expected return. Both the manager and workers can evaluate the state value, output actions, receive extrinsic rewards, and learn from interacting with the environment. For each agent, the Actor is a policy function parameterized by θ, which can provide the policy $\pi_\theta(\mathbf{A}|\mathbf{S})$. The Critic is a value function $V_\psi(\mathbf{S})$, which serves as a baseline to update the Actor's parameters. We have introduced the architecture of the Actor networks of the manager and workers in Section 4.1.1. Each Critic is established based on the corresponding Actor network but uses a different output layer to obtain single-valued state value estimation. The Critic of the manager supplements a linear layer f^{Mlinear} after the dilated LSTM to output the global state value $V_{\psi^{\text{M}}}(\mathbf{S}_t)$, i.e., $V_{\psi^{\text{M}}}(\mathbf{S}_t) = f^{\text{Mlinear}}(\hat{g}_t)$. The Critic of a worker u use a linear layer $f_3^{\text{W}_u}$ instead of the Softmax layer to obtain the local state value $V_{\psi_u^{\text{W}}}(z^t, \mathbf{S}_t)$, i.e., $V_{\psi_u^{\text{W}}}(z^t, \mathbf{S}_t) = f_3^{\text{W}_u}(f_2^{\text{W}_u}(w_t^u x_t^u))$. For each agent, the Actor and the Critic share most of the parameters within their corresponding networks. These parameters will be updated when minimizing the policy loss function and value loss function.

At each iteration, we conduct a simulation of the daily operations of the intercity ride-pooling services. The fleet dispatching decisions are made at each horizon to assign idle vehicles to lines in the simulation environment. After several iterations of interacting with the environment, we use batch gradient descent to update the policies of agents based on the transition experience of size $|\mathcal{B}|$. Let ψ^{M} and ψ_u^{W} be parameters in the Critic networks of manager and worker u, θ^{M} and θ_u^{W} be parameters in Actor networks of manager and worker u. Then the policy loss function and value loss function of the manager are defined as follows.

$$A_t^{\text{M}} = R_t + \gamma^{\text{M}} V_{\psi^{\text{M}}}(\mathbf{S}_{t+1}) - V_{\psi^{\text{M}}}(\mathbf{S}_t) \tag{28}$$

$$\mathcal{J}_{\psi^{\text{M}}} = \frac{1}{2|\mathcal{B}|} \sum_{t\in\mathcal{B}} (A_t^{\text{M}})^2 \tag{29}$$

$$\mathcal{J}_{\theta^{\text{M}}} = -\frac{1}{|\mathcal{B}|} \sum_{t\in\mathcal{B}} A_t^{\text{M}} \mathrm{d}_{\cos}(z_{t+c}^{\text{M}} - z_t^{\text{M}}, g_t(\theta^{\text{M}})) \tag{30}$$

Eq. (28) defines the advantage function of the manager, where R_t is the immediate reward of the whole platform defined in Section 3.3, and γ^{M} is a discount factor for the manager. We minimize the value loss function $\mathcal{J}_{\psi^{\text{M}}}$ and policy loss function $\mathcal{J}_{\theta^{\text{M}}}$ to update parameters of the Critic and Actor networks. Here, we adopt the definition of the policy loss function proposed by Vezhnevets et al. (2017). $\mathrm{d}_{\cos}(z_{t+c}^{\text{M}} - z_t^{\text{M}}, g_t(\theta^{\text{M}}))$ is defined as the angle between actual state transitions in c horizons and the normalized global goal g_t to measure how system state transition follows the global goal, where $\mathrm{d}_{\cos}(X, Y) = X^T Y/(|X|\,|Y|)$ indicates the cosine similarity between two vectors. On the other hand, the system goal is compared with state transition in c horizons, which provides a lower temporal resolution for the manager and enhances the foresight in our decision-making.

The worker agents are trained in a similar manner, and the corresponding value loss function and policy loss function are shown as Eqs. (31), (32), (33) and (34). The worker agent u is not only driven by its local immediate reward R_t^u but also encouraged by the intrinsic reward R_t^I to follow the goals. The local immediate reward R_t^u is defined in the formulation (6). The intrinsic reward R_t^I is the same for all workers, which is the average cosine similarities between system state transitions and the global goals in the past c horizons. The manager leverages the intrinsic rewards to motivate workers to choose actions that can facilitate the development of the system in accordance with the planned goals. Since the manager works in a lower temporal resolution, the worker's policy with a higher degree of following the goals is more likely to avoid myopic vehicle dispatching decisions and keep valid in a large latent state space. The agents have to balance the trade-off between their rewards R_t^u and the shared objective through a coefficient α, which regulates the relative weight of intrinsic reward. We set α as 0.5 in the following numerical experiments. Then based on the experience trajectory, each worker agent can be trained separately via minimizing the value loss function $\mathcal{J}_{\psi_u^{\text{W}}}$ and policy loss function $\mathcal{J}_{\theta_u^{\text{W}}}$.

$$R_t^I = \frac{1}{c} \sum_{i=1}^{c} \mathrm{d}_{\cos}(z_t^{\text{M}} - z_{t-i}^{\text{M}}, g_{t-i}) \tag{31}$$

$$A_t^{\text{W}_u} = R_t^u + \alpha R_t^I + \gamma^{\text{W}} V_{\psi_u^{\text{W}}}(z_{t+1}, \mathbf{S}_{t+1}^u) - V_{\psi_u^{\text{W}}}(z_t, \mathbf{S}_t^u) \tag{32}$$

$$\mathcal{J}_{\psi_u^{\text{W}}} = \frac{1}{2|\mathcal{B}|} \sum_{t\in\mathcal{B}} (A_t^{\text{W}_u})^2 \tag{33}$$

$$\mathcal{J}_{\theta_u^{\text{W}}} = -\frac{1}{|\mathcal{B}|} \sum_{t\in\mathcal{B}} A_t^{\text{W}_u} \log(p_{t,\theta_u^{\text{W}}}^u(z_t, S_t^u)) \tag{34}$$

4.1.3. Determine vehicle assignment

Notice that the sampled actions only indicate the number of vehicles assigned to each line and reserved at the city, the dispatching decision for each vehicle has to be determined so that the assignment in reality is nearly consistent with the dispatched vehicle numbers in $\mathbf{A}_t$. A MILP is proposed to map the numbers in $\mathbf{A}_t$ into the sets of dispatched idle vehicles for each line respectively. Let $\mathcal{E}_u$ be neighbors of the city u (including the city itself u), and a_v be the expected number of idle vehicles dispatched to the city $v \in \mathcal{E}_u$ obtained from $\mathbf{A}_t^u$. Among all idle vehicles in the city u at dispatching horizon t, there may be some vehicles reaching maximum daily work time, which are expected to return to their original cities and get off work in time. Let $\mathcal{F}_u^t$ be the set of vehicles that will be off duty in a few horizons. The objective function of the MILP includes three parts as mathematical expressions (35), (36), and (37).

$$p_1 = \sum_{v\in\mathcal{E}_u} \left| a_v - \sum_{k\in\mathcal{K}_u^t} x_{kv} \right| \tag{35}$$

$$p_2 = \sum_{v \in \mathcal{E}_u} \sum_{k \in \mathcal{F}_u^t} x_{kv} D_{vd_k} \tag{36}$$

$$p_3 = \sum_{k \in \mathcal{F}_u^t} \sum_{v \in \mathcal{E}_u \setminus \{u\}} x_{kv} \tilde{r}_k \tag{37}$$

where x_{kv} is the binary decision variable indicating whether vehicle k is dispatched to city v in this horizon. The mathematical expression (35) represents the gap between realistic assignment and sampled actions of worker agent W_u. The mathematical expression (36) represents the total distances for all vehicles in $\mathcal{F}_u^t$ to return origin cities after finishing the dispatched trip, where D_{vd_k} indicates the distance between city v and the origin city of vehicle k. Mathematical expression (37) is constructed from a fairness perspective, where $\tilde{r}_k$ is a predefined parameter representing the reward for dispatching vehicle v to work rather than keep idling. The vehicles with lower percentages of time spent in transit compared with the average level are set $\tilde{r}_k = -1$, while others receive $\tilde{r}_k = 0$. Thus, we can minimize p_3 so that vehicles that spend more time in idle state are given higher priority to be assigned to lines rather than being reserved in the city. Then the MILP can be formulated as follows.

$$\min \sum_{v \in \mathcal{E}_u} \beta_1 z_v + \beta_2 p_2 + \beta_3 p_3$$

$$\text{s.t.} \sum_{v \in \mathcal{E}_u} x_{kv} = 1 \qquad \forall k \in \mathcal{K}_u^t \tag{38}$$

$$\sum_{v \in \mathcal{E}_u} D_{vd_k} x_{kv} \le D_{ud_k} \qquad \forall k \in \mathcal{F}_u^t \tag{39}$$

$$a_v - \sum_{k \in \mathcal{K}_u^t} x_{kv} \le z_v \qquad \forall v \in \mathcal{E}_u \tag{40}$$

$$-a_v + \sum_{k \in \mathcal{K}_u^t} x_{kv} \le z_v \qquad \forall v \in \mathcal{E}_u \tag{41}$$

$$x_{kv} \in \{0,1\} \qquad \forall k \in \mathcal{K}_u^t, v \in \mathcal{E}_u \tag{42}$$

The constraints include that each vehicle can only be dispatched to exactly one city, as shown in constraint (38), and that vehicle reaching maximum daily work time is not allowed to be dispatched to cities farther from its origin city, as shown in constraint (39). The objective function is a linear combination of mathematical expressions (35)–(37). The weights β_1, β_2, and β_3 are set to firstly ensure that vehicles are dispatched consistently with $\mathbf{A}_t$, and consider other objectives afterward. We linearize the absolute terms in (35) by introducing binary auxiliary variables z_v to constraints (40) and (41). This MILP can be solved quickly using commercial software.

We conclude this subsection with the pseudo-code in Algorithm 1, which illustrates the complete process of training the proposed framework in a simulated environment.

Algorithm 1 Training process of MFuN

```
Initialize parameters ψ^M, {ψ_u^W}_u, θ^M and {θ_u^W}_u ;
for i = 1 → Iter do
    Reset the simulation environment, observe initial states S_0;
    for t = 0 → T − 1 do
        MFuN outputs g_t and {p_t^u} based on S_t and hidden states h_t^M;
        For each city, sample {A_t^u} based on {p_t^u} ;
        For each city, determine the assignment of each vehicle by solving the MILP proposed in subsection 4.1.3;
        Solve the vehicle routing problem for each line at each matching interval;
        Record rewards R_t, {R_t^u} and R_t^I, observe new environment states S_{t+1}, reserve the transition experience;
    end for
    Calculate advantages based on Equations (28), (31), and (32);
    Update parameters based on Equations (29), (30), (33), and (34).
end for
```

4.2. Lower level: Adaptive large neighborhood search heuristic

At the lower level of the framework, we solve the dynamic multi-vehicle dial-a-ride problem proposed in Section 3.4 using the ALNS heuristic at the beginning of each matching interval to generate a routing plan for each vehicle en route. The ALNS algorithm is executed for each line separately. The inputs of the algorithm are the same as the basic settings in Section 3.1. The solution comprises the routes of all vehicles traveling in this line, with each route represented as a sequence of nodes for picking up and delivering passengers until reaching the depot. During each iteration of the neighborhood search, a portion of orders in the current solution is removed by *removal operators* while the nodes corresponding to some unfulfilled orders are inserted into certain positions of the sequence by *insertion operators* to construct a new solution. The choice of operators depends on the cumulative devotion to the objective function of each operator.

The neighborhood search process is initiated with a feasible solution. The initial routes are constructed based on the orders that have already been matched, following the sequence determined in earlier periods. The pick-up and delivery nodes of the unmatched orders are then inserted into the routes using the **Regret-2 method**. This method identifies the optimal position for inserting each order into each route. This is achieved by recursively evaluating all feasible positions given the order and the route. For each order, the difference between the profit obtained from the best insertion position and the second-best insertion position is calculated as the Regret-2 value. At each iteration, the order with the highest Regret-2 value is inserted into its best position, and the Regret-2 values for the remaining unmatched orders are updated for the subsequent insertions. This process continues until all feasible insertions have been completed.

One basic operation in ALNS is to evaluate the insertion of a single order (both pickup and delivery nodes) into given feasible routes at the initialization stage and insertion stage. It is crucial to test the capacity feasibility and time window feasibility of an insertion, among which checking time window feasibility is extremely complex and time-consuming. We use the forward time slack method (Gschwind and Drexl, 2019) to accelerate the check on whether the time window constraints of each node along the sequence are violated.

The neighborhood search process is embedded in a simulated annealing metaheuristic. The initial temperature is τ_0, which decreases at each iteration as $\tau \leftarrow \tau \times c$, where c is the cooling rate. The neighborhood search process ends at the iteration with a temperature lower than the stopping temperature τ_e. ALNS learns over iterations which operators are more appropriate for the problem. Suppose there are n removal operators $\mathcal{R} = \{r_1, r_2, r_3, \dots, r_n\}$ and m insertion operators $\mathcal{I} = \{i_1, i_2, i_3, \dots, i_m\}$, where each operator o is initialized with a score $s_o = 0$ and a weight $w_o = 1$. At each iteration, the operator is selected probabilistically based on the operator weights. For example, the probability of removal operator r_i being selected is $P(r_i) = \frac{w_{r_i}}{\sum_{r_i \in \mathcal{R}} w_{r_i}}$. Let $F(S)$ be the objective function value of the current solution S, $F(S')$ be the objective function value of the new solution S', $F(S^{\text{best}})$ be the objective function value of the historical best solution S^{best}. Suppose at a certain iteration, we choose operator $r_p \in \mathcal{R}$, $i_q \in \mathcal{I}$, then there exist four situations after the removal stage and insertion stage. The details of score updates are explained in pseudo-code in Algorithm 2. The parameters are set as $\theta_1 > \theta_2 > \theta_3 > \theta_4$ such that operators leading to better solutions are enhanced adaptively in later stages. The weight of operator o is updated as $w_o \leftarrow 0.5w_o + 0.5\frac{s_o}{n_o}$, where n_o is the number of times operator o is used. The operator selection criteria reflect the flexible balance between exploration and exploitation throughout the iterations.

Algorithm 2 Adaptive large neighborhood search heuristic

Input: Order information $\mathcal{O}_{uv}$, vehicle information $\mathcal{K}_{uv}$;
Output: Routing solution S;

Construct initial routing solution S, $S^{\text{best}} \leftarrow S$;
For each operator o, $s_o \leftarrow 0$, $w_o \leftarrow 1$,$n_o \leftarrow 0$;
$\tau \leftarrow \tau_0$;
while $\tau > \tau_e$ **do**
  for $j = 1 \rightarrow$ Iter **do**
    Choose removal operator $r \in \mathcal{R}$ and insertion operator $i \in \mathcal{I}$ based on weights;
    $S' \leftarrow i(r(S))$, $n_r \leftarrow n_r + 1$, $n_i \leftarrow n_i + 1$;
    if $F(S') > F(S^{\text{best}})$ **then**
      $S^{\text{best}} \leftarrow S'$, $S \leftarrow S'$, $s_r \leftarrow s_r + \theta_1$, $s_i \leftarrow s_i + \theta_1$;
    else if $F(S^{\text{best}}) \geq F(S') > F(S)$ **then**
      $S \leftarrow S'$, $s_r \leftarrow s_r + \theta_2$, $s_i \leftarrow s_i + \theta_2$;
    else if $F(S') \leq F(S)$ **then**
      Generate random variable $\epsilon \in [0, 1]$;
      if $\epsilon \leq \exp(-\frac{F(S)-F(S')}{K\tau})$ **then**
        $S \leftarrow S'$, $s_r \leftarrow s_r + \theta_3$, $s_i \leftarrow s_i + \theta_3$;
      else
        $s_r \leftarrow s_r + \theta_4$, $s_i \leftarrow s_i + \theta_4$;
      end if
    end if
  end for
  For each operator o, $w_o \leftarrow 0.5w_o + 0.5\frac{s_o}{n_o}$;
  $\tau \leftarrow \tau \times c$;
end while
return S^{best}.

The operators used in our study are similar to Ghilas et al. (2016). The removal operators include:

(1) **Random removal operator**: randomly removes ϕ_r orders from the solution.

(2) **Shaw removal operator**: randomly removes 1 order, then removes $\phi_r - 1$ orders that are similar in locations and time window constraints.

(3) **Worst removal operator**: removes ϕ_r orders with the highest traveling cost.

(4) **Time-based removal operator**: removes ϕ_r orders with the largest pick-up time delay.

The insertion operators include:

(1) **Greedy insertion operator**: iteratively inserts the order with the maximum additional objective function value to its best feasible position of the routes, until no feasible insertion with positive profit can be made.

(2) **Distance greedy insertion operator**: iteratively inserts an order with the shortest additional distance to its best feasible position of the routes, until no feasible insertion with positive profit can be made.

(3) **Regret-m insertion operator**: similar to the **Regret-2 method** introduced for initialization above, but repeatedly insert orders with the largest differences of profit between the best insertion position and the m-best insertion position. The procedure ends when no feasible insertion with positive profit can be made.

5. Numerical experiments

In this section, we conduct numerical experiments to verify the effectiveness of our proposed framework under different supply and demand scenarios on three city networks. The first two series of experiments are conducted on toy networks and the third series of experiments are based on the realistic network and operational data in Xiamen and its surrounding cities. All computational experiments are implemented on the computation platform with Intel Core i5-12600KF CPU@3.70 GHz, 16 GB RAM and RTX3070 (8 GB) GPU. The algorithms are coded with Python 3.6. The neural network is constructed and trained with Pytorch. The mixed integer linear programs are solved by Gurobi 10.0.1.

A simulator is established to approximate the dynamics of intercity ride-pooling services, including the generation of orders and the operations of the vehicle fleet. Each vehicle starts working at a predetermined time and location. The system initially assigns vehicles to lines and then updates the routes of all vehicles in transit. Once vehicle dispatching and routing are executed, the states are updated accordingly. This procedure continues until the end of the operational time to complete a daily simulation. During the training process of the proposed reinforcement learning model, a full simulation of daily operations constitutes an episode of trajectory data for policy learning. To alleviate the computational burden and prevent memory overflow, we simplify the training process in the simulation environment in three ways. *Firstly*, during training iterations, the vehicle routing problems are only solved once in a dispatching horizon rather than solved at every matching interval. Vehicles travel along the planned routes during the horizon without intermediate route updating until the next assignment of idle vehicles. This simplification is reasonable when the newly emerging orders within a dispatching horizon are assumed to be known at the first matching process. *Secondly*, compared to the settings in the literature focus on DARP (Ghilas et al., 2016; Gschwind and Drexl, 2019), fewer iterations are performed in ALNS to solve the multi-vehicle routing problem in our numerical experiments. This approach significantly shortened the training time since ALNS needs to be repeatedly solved in the simulation. *Thirdly*, the vehicle routing problem is based on the Euclidean metric in both the toy network and the realistic network. We calibrate the detour ratio in the realistic network using trajectory data from operations to approximate the travel distances in reality.

Next, we introduce the basic settings of the simulation environment. We set the duration of a horizon to be 20 min for all subsequent experiments. In two toy networks, we consider daily intercity ride-pooling services for 40 horizons, while in the realistic network, we consider the services for 60 horizons (a daily simulation from 4:00 to 24:00). We set different fleet sizes for different experiments, and we will introduce the fleet size and their attendance settings in the corresponding subsections. We consider seven-seater vehicles in all experiments, with each vehicle capable of serving up to six passengers. Drivers always have to take a rest of 20 min after an intercity trip for safety considerations, and the maximum working time for each driver is limited as $\bar{W} = 10$ h. In experiments on the toy networks, the vehicle speed s is set to 60 km/h, and the travel cost per kilometer C is 1. In experiments on the realistic network, by calibrating based on the daily operation data, we set the average vehicle speed within the city to 25 km/h, the intercity average vehicle speed s to 70 km/h, and the travel cost per kilometer C to 0.5.

The simulation environment can also generate demand as required. We generate order information for each line respectively based on specific order arrival rates. In experiments on toy networks, we sample the widths of the two time windows for each order from a normal distribution with a mean of 40 min and std of 15 min. And the duration from the latest pick-up time $\check{U}_p$ to the latest arrival time $\hat{U}_p$ and is sampled from a normal distribution with a mean of $2 \times D_{uv}/s$ minutes and std of $0.25 \times D_{uv}/s$ minutes, where D_{uv} is the distance between two cities. The coordinates of the origin and destination nodes of each order are uniformly distributed in their respective circular cities. As for experiments on realistic networks, we will explain how to generate order information in Section 5.4. The reservation lead time (the time from order placement to the beginning of the time window for picking-up) follows a truncated normal distribution $\mathcal{N}(40, 30)$ within a range from 0 to 120 min. And the penalty rate for losing one passenger p_e is set to 0.5. Other parameters related to network structures, such as the trip fare for each line, will also be provided in corresponding subsections.

Our framework integrates the reinforcement learning method, MILP, and heuristic algorithm, leading to numerous hyperparameters for fine-tuning. We conducted several numerical experiments to identify good parameter combinations. Here, we present the values of the essential algorithm parameters employed in the subsequent experiments. We first focus on the training procedure of the reinforcement learning method at the upper level of the framework. In each subsequent experiment, the neural networks are trained for 3000 episodes, and the test results come from trajectories of 20 episodes based on the well-trained neural networks. In MFuN, we set $\gamma^M = 0.99$ for the manager agent and $\gamma^W = 0.95$ for worker agents. For the manager, the number of looking forward horizons is set to $c = 3$, and the dilation radius for manager LSTM is set to $r = 5$. We use RMSprop optimizer for manager and workers with a learning rate of the policy optimizer as 2.5×10^{-4}. We use batch gradient descent to update the policies of agents and the batch size $|\mathcal{B}|$ is set to be 5 episodes. In MFuN, we use the neural networks with hidden layer size shown as Fig. 6. d_{input} is the length of $\mathbf{S}_t$ and $d^u_{action} = V^u \times |\mathbf{A}^u|$ is the length of the output probability vector. The size of hidden layers in the neural networks

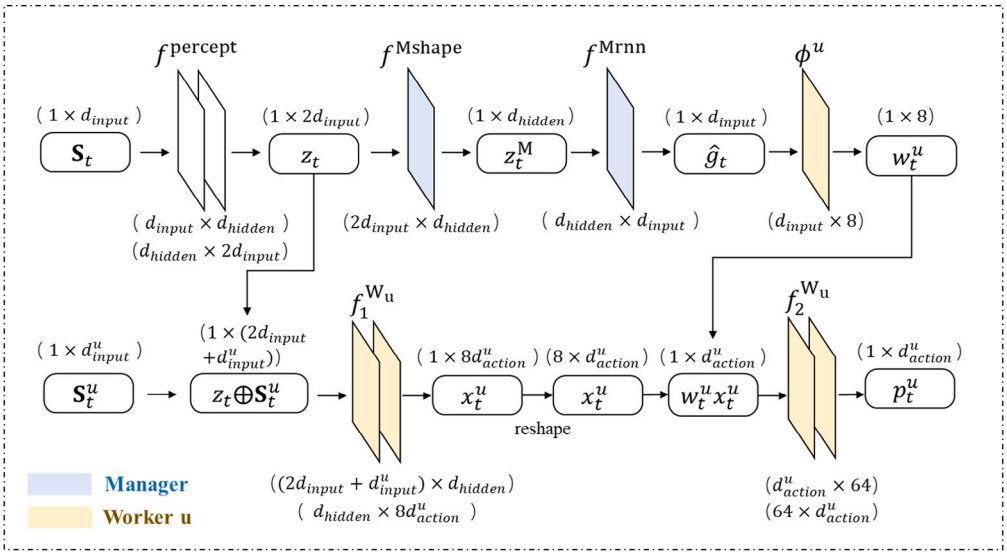

Fig. 6. Neural network hidden layer size in MFuN.

scales up as the number of cities and lines increases. We set d_{hidden} to 128 in experiments on toy networks and 256 in experiments on realistic networks.

There also exists hyper-parameters for the MILP in Section 4.1.3. We set the weights $\beta_1, \beta_2, \beta_3$ in the objective function to 5, 0.02, and 1 respectively. This parameter setting considers the distinct scales of each component in the objective function, to firstly ensure that vehicles are dispatched consistently with $\mathbf{A}_t$, and consider other objectives afterward. As for control parameters in the ALNS, we set the hyper-parameters to balance the trade-off between computational time and the algorithmic performance. For the outer loop, we set initial temperature $\tau_0 = 100$, stopping temperature $\tau_e = 5$, and cooling rate $c = 0.2$. And we set the number of iterations in the inner loop to 5, indicating the operator weights are updated every five times of neighborhood search. The parameters for updating operator scores are set as $\theta_1 = 20$, $\theta_2 = 12$, $\theta_3 = 6$, $\theta_4 = 2$. At each neighborhood search, ϕ_r, the number of orders removed, is set as one-fourth of the total number of unserved orders in the current solution.

To better evaluate the effectiveness of the proposed MFuN framework on vehicle resource allocation among cities, we introduce a Myopic vehicle dispatching strategy as the benchmark. At each horizon, the Myopic approach determines the idle vehicle assignment for each city respectively. For each city, the Myopic method first estimates the number of vehicles required for serving currently unmatched orders on each line separately. If the total number of idle vehicles within a city is sufficient to fulfill the assignment, vehicles are randomly assigned to lines such that the number of vehicles dispatched to each line is nearly consistent with the planned number, while the remaining vehicles are reserved for the next dispatching. If there are not enough idle vehicles for this assignment, numbers of vehicles dispatched to each line are proportionally reduced until all idle vehicles are deployed. The Myopic method also uses ALNS to solve the vehicle routing problem at the lower level of the framework. Myopic method offers clear insights into the performance of decision-making based on the current states and exhibits stronger interpretability in contrast to RL methods. Thus we compare the optimization performance of the proposed MFuN framework with the Myopic method in Sections 5.2, 5.3, and 5.4. We also propose several reinforcement learning methods as benchmarks to show the advantage of MFuN in policy training. The details of these RL methods and their performances are presented in Section 5.1.

The evaluation of the proposed methods will be based on several key indices, including the average daily profit and the order fulfillment ratio. In the context of intercity passenger transportation, the platforms also emphasize the vehicle utilization rate, which represents the percentage of time vehicles spend in transit. In the subsequent sections, we will compare the performance of different methods based on these indices. This comparative analysis aims to evaluate the system's overall performance and illustrate the specific characteristics and properties of each method.

5.1. Comparative analysis of network architectures

We first focus on the training process of the reinforcement learning method at the upper level of the framework. To validate the learning effectiveness of the MFuN network structure, we use three baselines with different network architectures to conduct a comparative analysis.

- **FuN**: The first baseline is the centralized decision feudal networks (FuN), and we use the Advantage Actor–Critic algorithm to train the networks. This model leverages a similar network architecture as MFuN but only consists of a single manager and a single worker. The worker networks jointly generate the dispatching decisions for all cities, so its outputs can be interpreted as the concatenation of the outputs of all workers in MFuN.

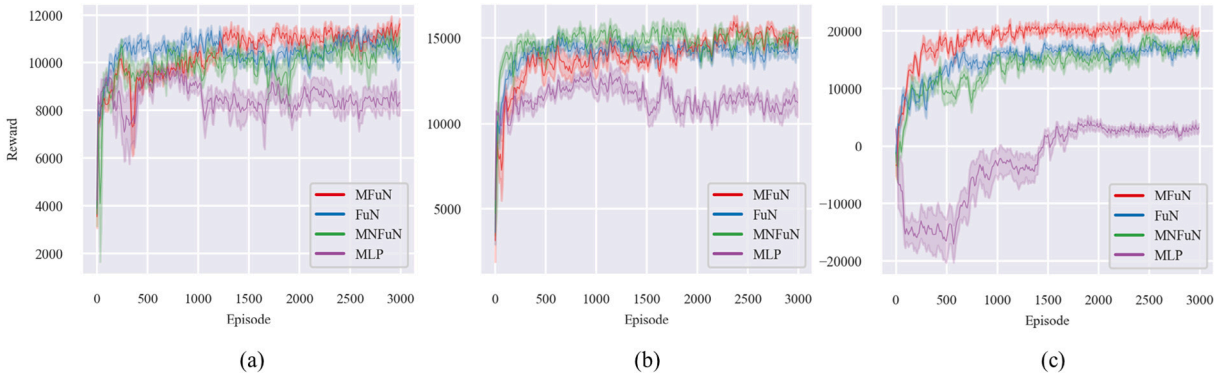

Fig. 7. Learning curves of different RL methods in different networks. (a) The 2-city-2-line toy network. (b) The 3-city-4-line toy network. (c) Realistic network. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- **MNFuN**: The second baseline is the multi-agent 'non-feudal' feudal networks (MNFuN). MNFuN has the same network architecture as MFuN. The 'non-feudal' represents that no intrinsic reward is used and the manager output g_t is trained with gradients coming directly from the worker. In this framework, there is no clear boundary between the manager and the workers. The manager can only be regarded as the predecessor network of the worker because it cannot receive learning signals from the environment and get trained alone. We also use the Advantage Actor–Critic algorithm to train the MNFuN.
- **MLP**: The third baseline uses a multi-layer perceptron (MLP) as the network and trains the model by the Proximal Policy Optimization algorithm (PPO). In the experiments on toy networks, we use an MLP that consists of three hidden layers of size 128, while in experiments on realistic networks, the hidden layer size is set as 256. Similar to FuN, this model also makes centralized decisions in a multi-discrete manner.

The outputs of different neural networks are then used to generate the agent actions by sampling and drive the simulation environment in the same way as the MFuN framework. By comparing the network architecture of MFuN with these baselines, we aim to assess its advantages in learning an intelligent policy in the stochastic dynamic vehicle resource allocation problem.

Fig. 7(a) and (b) show the learning curves of the models on relatively small networks, which contain only 2 and 4 lines respectively. The demand and supply pattern on two toy networks will be introduced in subsequent subsections. The intercity distance is set as 60 km, and the trip fare is also set as 60. The red curves in the figures represent the reward obtained by the MFuN method over episodes, and green, blue, and purple curves represent three baselines respectively. Benefiting from the guidance of hierarchical reinforcement learning, the training results of MFuN, FuN, and MNFuN are significantly more stable than MLP. The results of MFuN and FuN are basically similar in this example, however, the advantage of multi-agent decision becomes more obvious when the network complexity increases. Fig. 7(c) shows the learning curve on a realistic network with 7 cities and 12 lines. Compared with smaller networks, the advantage of hierarchical network architecture over fully connected networks is more pronounced. The learning curves of MLP exhibit dramatic fluctuations, while the training process of the hierarchical framework is more stable. Additionally, MFuN also achieves a higher daily profit compared to FuN in the task environment with large-scale networks. Centralized decision approaches tend to exhibit relatively poor performance when the joint action space of agents expands exponentially. This highlights the advantage of MFuN in scalability because each worker agent can learn its policy autonomously under the guidance of global goals, allowing for better performance as the number of agents grows. In networks with more cities and intercity lines, MFuN notably ensures effective cooperation among agents to achieve a pronounced advantage in the global return.

The comparison with baselines also highlights the advantage of MFuN in associating the actions of agents with subsequent extrinsic rewards and state transitions. We can gain insights into the policies learned by different models from observing the vehicle dispatching actions in the simulation environment on a 2-city-2-line network. Two intercity lines (line 1 and line 2) are operated between cities A and B. Line 1 (from A to B) experiences a steady low demand flow throughout the day while line 2 (from B to A) exhibits a demand peak as shown in Fig. 9. A fleet of 20 vehicles originating from city A is available for the intercity ride-pooling services of line 1 and line 2. A more detailed introduction to the demand pattern of this toy network will be provided in the next subsection. Fig. 8 illustrates the vehicle dispatching patterns under the MFuN, MNFuN, FuN, and Myopic methods respectively. The upper figures show the average number of departures of idle vehicles at both lines within each two horizons (40 min), and the lower figures show the average number of lost orders (green), matched orders (red), and the orders not matched in this period but still available (cyan). Compared to the Myopic approach, all three reinforcement learning methods dispatch more vehicles from city A to city B, even when both lines are experiencing low demand. This proactive dispatching strategy ensures more sufficient vehicle resources available in city B ahead of the demand peak to reduce order loss. Among three reinforcement learning approaches, MFuN dispatches the most vehicles to line 2 during peak hours, resulting in the highest daily profits. The abundant vehicle supply of city B from 320 min to 440 min under MFuN's dispatching policy can be attributed to its accurate identification of periods with higher profits, so more vehicles are prepared at city B in advance. This underscores MFuN's heightened capability to associate the actions of each agent at each period with the expected global return in a long-term dynamic environment.

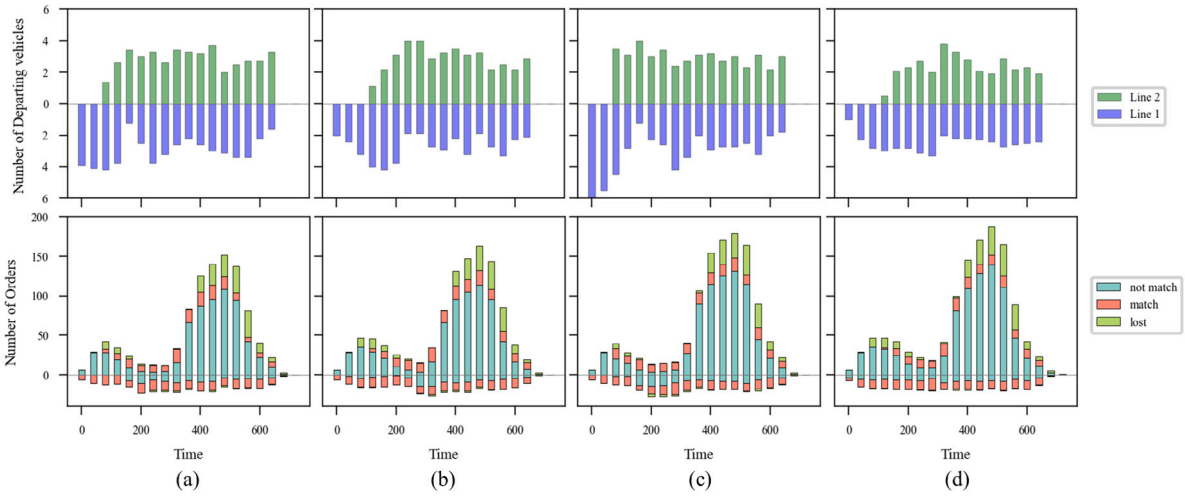

Fig. 8. Number of departures and orders in each period. (a) MFuN. (b) MNFuN. (c) FuN. (d) Myopic. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

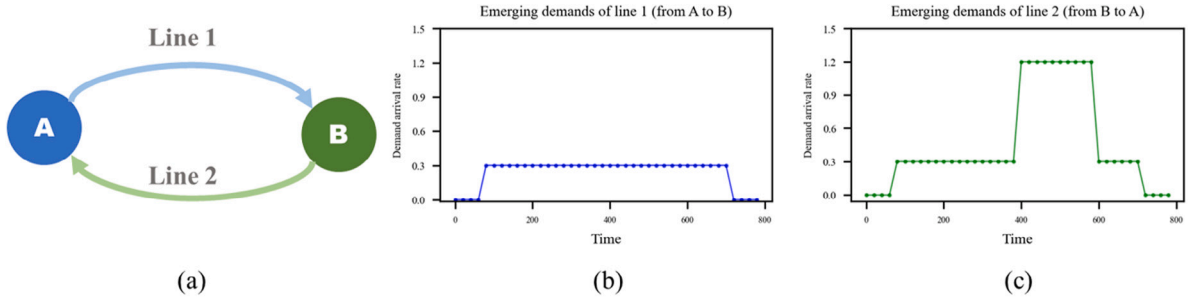

Fig. 9. A 2-city-2-line toy network. (a) Topology of the network. (b) Demand from city A to B. (c) Demand from city B to A.

5.2. A 2-city-2-line toy network

The 2-city-2-line toy network is the simplest and most basic intercity transportation network. Numerical experiments based on this network can demonstrate the effectiveness of the proposed framework on temporal scheduling. The dispatching decisions in the 2-city-2-line toy network illustrate how various methods adapt to fluctuations in supply and demand by only adjusting the departure time of each vehicle. Fig. 9(a) depicts the network topology of two cities, which have a radius of 15 km and the distance between two cities is 60 km. Fig. 9(b) portrays the demand of two lines. As introduced before, there is a relatively low demand for line 1 throughout the day, while there is a demand peak lasting for 200 min on line 2, which is four times higher than the demand during low-demand periods. Notice that the demand arrival rate here indeed represents the arrival rate for the beginning of the pick-up time windows, so the platform will receive the reservation earlier due to the reservation lead time. In this series of experiments, a fleet of 20 homogeneous seven-seater vehicles is utilized. The vehicles start their operations in city A at the initial moment and are available to serve ride requests throughout the day.

5.2.1. Sensitivity analysis of trip fare

In this subsection, we examine the impact of the one-way pooled-ride trip fare on the vehicle dispatching strategy. We assume equal trip fares for both directions, denoted as $R^{AB} = R^{BA}$. The trip fare values are varied, specifically set at 30, 45, 60, 75, and 90 for each passenger. Given that the one-way traveling cost is at least 60, a vehicle must transport at least two passengers to attain profitability when the trip fare is relatively low. However, as the trip fare increases, the condition for vehicle profitability with respect to the occupancy rate becomes less stringent, allowing for more flexible dispatching strategies.

As depicted in Fig. 10, it can be observed that the Myopic method exhibits little sensitivity to changes in trip fare. Under this method, the one-way trip fare does not directly influence the dispatching schedule of vehicles. However, it does impact vehicle routing, as higher trip fares encourage picking up more passengers with longer detour distances within the city. Consequently, this leads to a slight increase in the order fulfillment ratio. In contrast, the MFuN framework demonstrates more intelligent vehicle scheduling. It consistently achieves higher daily profits and order fulfillment ratios compared to the Myopic method across various

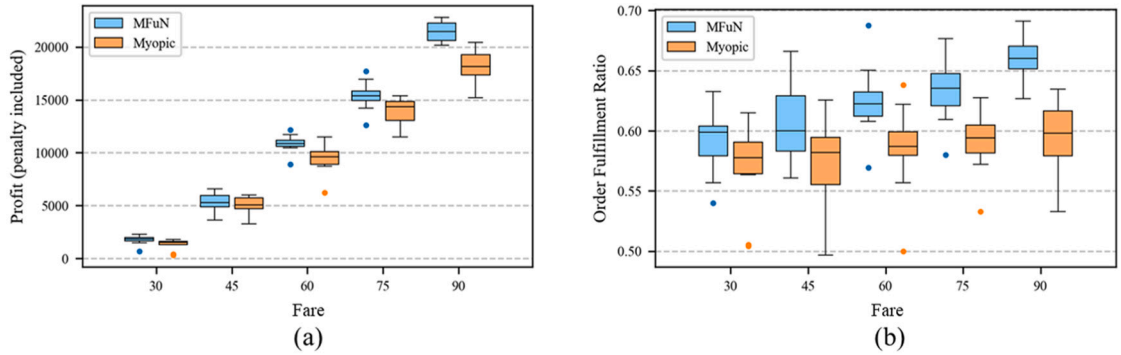

Fig. 10. Performance of MFuN and Myopic in the 2-city-2-line toy network under different trip fares. (a) Profit. (b) Order fulfillment ratio.

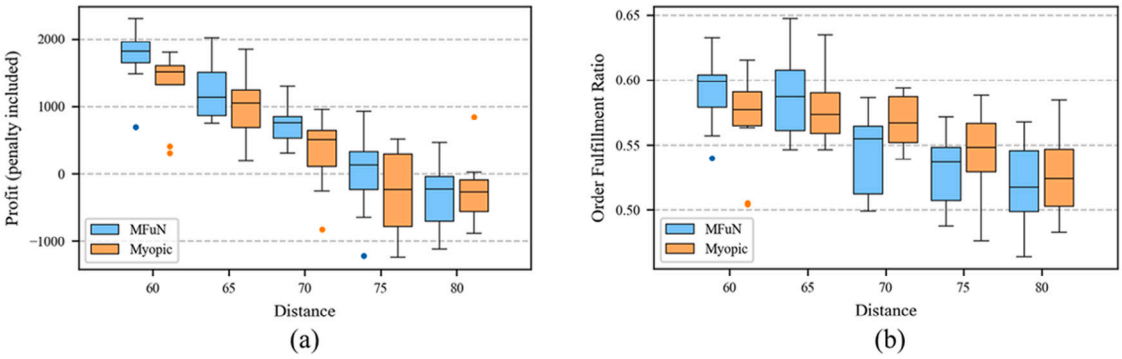

Fig. 11. Performance of MFuN and Myopic in the 2-city-2-line toy network with different inter-city distances. (a) Profit. (b) Order fulfillment ratio.

trip fare settings. Additionally, the increase in trip fare has a more pronounced positive impact on MFuN, particularly in improving the order fulfillment ratio. Under the high trip fare scenarios, MFuN achieves a significant improvement of 16.8% in the order fulfillment ratio during demand peak periods, and a 7.1% improvement in the daily average order fulfillment ratio, compared to the Myopic method.

5.2.2. Sensitivity analysis of the distances between cities

In this subsection, we focus on the differences in dispatching strategies under different inter-city distances, ranging from 60 to 80, while keeping the one-way trip fare fixed at 30. Considering a one-way trip fare of 30 and an inter-city distance of 60 to 80, a vehicle must carry at least 2 to 3 passengers on a one-way trip to attain profitability. As depicted in Fig. 11, both the average daily profit and the order fulfillment ratio gradually decline as the distance between cities increases, regardless of the method employed. However, the MFuN approach demonstrates advantages in terms of profit in all scenarios, although it exhibits a lower order fulfillment ratio in cases where the distance is relatively large. The MFuN framework aims to enhance overall system profit by increasing the average number of passengers per trip, even if it means some orders may not be served due to a more conservative dispatching strategy. Furthermore, when the distance between cities is large, MFuN optimally utilizes the abundant demand during peak periods. This accumulation of vehicle resources in advance allows for a higher occupancy rate during peak periods, leading to improved efficiency. In contrast to Section 5.2.1, where MFuN intensifies the frequency of departures to serve more orders in scenarios with high trip fares, in cases with a relatively large inter-city distance, MFuN reduces the frequency of departures to enhance one-way profitability. This showcases the MFuN framework's ability to propose diverse and efficient dispatching strategies tailored to different demand and supply scenarios.

5.3. A 3-city-4-line toy network

The 3-city-4-line toy network contains one central city (A) and two neighboring cities (B, C). At the beginning of each horizon, the idle vehicles in the central city can be assigned to lines to any of the neighboring cities immediately or continue to wait for the next dispatching. Compared to experiments conducted in the 2-city-2-line network in Section 5.2, this subsection is devoted to demonstrating the effectiveness of the proposed framework on spatially allocating vehicle resources. Fig. 12(a) depicts the network topology, where the distance from city A to city B and C is both 60 km, and the trip fares are set to 30 for all lines. Line 1 and line 3 both originate from city A, while line 2 and line 4 originate from city B and city C respectively. Fig. 12(b) depicts the demand

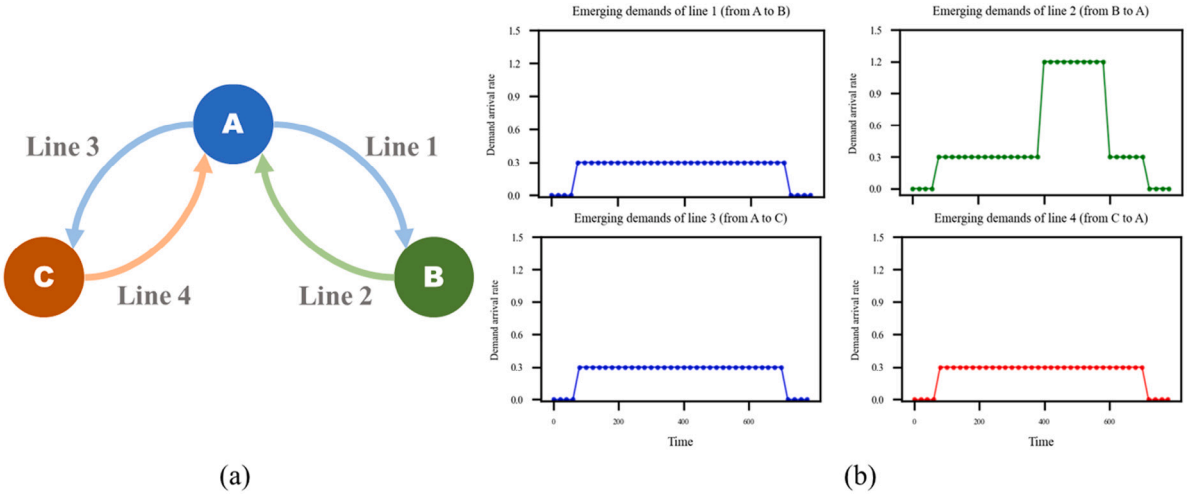

Fig. 12. The 3-city-4-line toy network. (a) Topology of the network. (b) Demand of all lines.

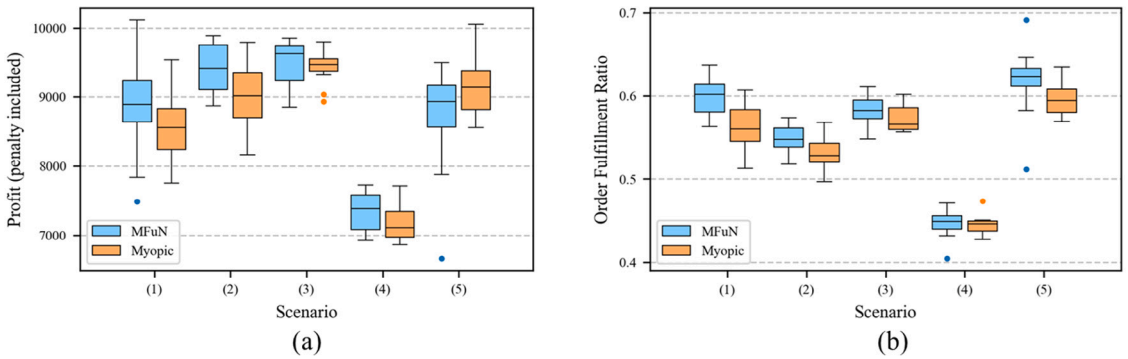

Fig. 13. Performance of MFuN and Myopic in the 3-city-4-line toy network under five demand and supply scenarios. (a) Profit. (b) Order fulfillment ratio.

fluctuation on four lines. Among the four lines, a demand peak is observed during the evening periods specifically from city B to city A, whereas the other three lines experience relatively low demand throughout the day. In this series of experiments, a fleet of 30 homogeneous vehicles is considered.

Based on the network topology shown in Fig. 12(a), we design 5 supply and demand scenarios as follows to evaluate the spatial resource allocation performance.

1. All vehicles are initially located in city A and the demand pattern is demonstrated in Fig. 12(b).
2. All vehicles are initially located in city A. There also exists a demand peak on line 4 based on demands demonstrated in Fig. 12(b), but the intensity is half of the demand peak on line 2.
3. All vehicles are initially located in city A. The demand fluctuation is based on demands in scenario (2) while the demand peak occurs at the same time on lines 1 and 3 instead of lines 2 and 4.
4. The demand pattern is the same as Fig. 12(b), while all vehicles are initially located in city B.
5. The demand pattern is the same as Fig. 12(b), while all vehicles are initially evenly distributed in three cities.

Fig. 13 are box plots on daily profits and order fulfillment ratio of MFuN and Myopic method under five scenarios. The daily revenue of MFuN is relatively higher than the values of Myopic among the most supply and demand scenarios, basically attaining 3% to 7% improvement compared to the Myopic method. MFuN also maintains a relatively satisfactory order fulfillment ratio compared to the Myopic method.

However, the optimization mechanism differs across the five scenarios. Fig. 14 illustrate the average daily profit and the average number of fulfilled orders of two methods in peak periods and off-peak periods under various scenarios. Under scenarios (1) and (2), the advantage of MFuN is mainly reflected in serving more orders during high-demand periods. The MFuN framework dispatches more vehicles to city B compared to city C before and during the peak periods, which is the origination of the high-demand line. Before and during peak periods, the MFuN framework dispatches a larger number of vehicles to city B, which is the origin of the high-demand line. This allocation strategy of vehicle resources has a slight impact on service capacity during off-peak periods but

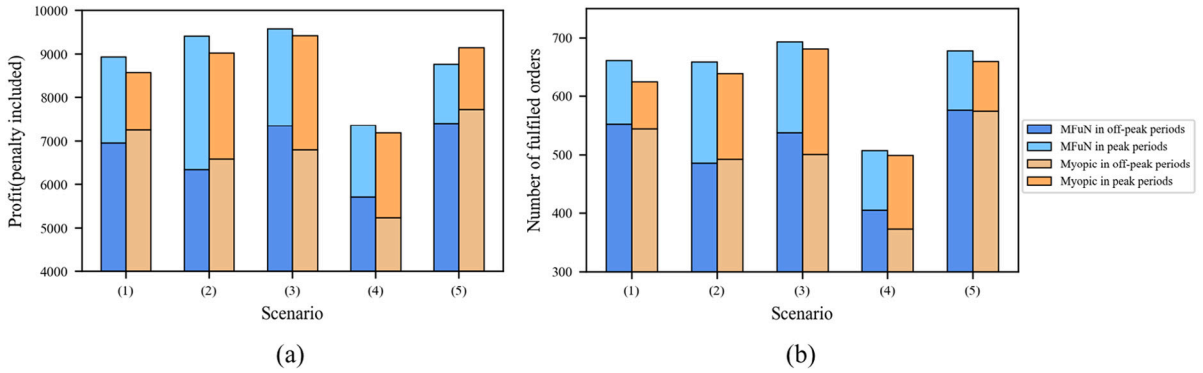

Fig. 14. Performance of MFuN and Myopic in the 3-city-4-line toy network during peak and off-peak periods under five demand and supply scenarios. (a) Profit. (b) Number of fulfilled orders.

ensures a relatively abundant availability of vehicles during the demand peak. There are 17% more vehicle departures from city B to fulfill 20% more orders in high-demand lines during the peak periods under the dispatching of the MFuN framework compared to the Myopic method.

In scenarios (3) and (4), the advantage of the MFuN framework during peak periods is less apparent. However, MFuN excels in providing more services primarily in low-demand lines during off-peak periods, leading to improved daily profits. This is primarily attributed to the demand peak under scenarios (3) and (4) occurring on lines originating from cities where vehicles are initially concentrated. Under the dispatching strategy of the Myopic method, fleet operations are insufficient in the early periods of the day. As a result, a certain number of idle vehicles accumulate at the initial city during low-demand periods before the peak demand emerges. In contrast, the MFuN framework proactively dispatches more vehicles to the neighboring cities in the early stages, which exhibits more advantages in terms of fulfilling orders in low-demand lines during the off-peak periods.

Additionally, when vehicles are evenly distributed initially under scenario (5), no significant advantages are observed for the performance of the MFuN strategies compared to the Myopic method. In this particular scenario, MFuN incurs higher traveling costs, resulting in slightly lower overall profits compared to the Myopic method.

This series of experiments highlights that the optimization effect of the MFuN framework lies primarily in its ability to exploit flexible vehicle resource allocation to mitigate spatial imbalances between supply and demand. The MFuN framework excels in scenarios where there is a significant imbalance between the demand on a specific line and the supply in its originating city, such as during peak periods in scenarios (1) and (2). In these cases, MFuN demonstrates clear advantages in efficiently dispatching fleets to mitigate the effects of supply shortages. However, in other scenarios, the advantages of the MFuN framework become less apparent due to the relatively balanced supply and demand conditions. These results highlight the adaptability of the MFuN framework to address spatial imbalances and effectively allocate resources where they are most needed. By strategically dispatching vehicles and optimizing fleet operations, MFuN can mitigate the negative impacts of supply shortages and enhance the overall performance of intercity ride-pooling services.

5.4. Realistic network

After verifying the effectiveness of the proposed framework from the perspective of temporally vehicle scheduling and spatial fleet allocation in Sections 5.2 and 5.3 respectively, we conduct numerical experiments based on a realistic network of Xiamen and its surrounding cities in this subsection. The Xiamen city cluster is situated on the southeast coast of China. The core urban area, Xiamen Island, has a developed economy and thriving market, which is closely interconnected with surrounding cities, forming a network of transportation and economic activities. For our study, we obtain the operational data from an intercity ride-pooling platform operating in Xiamen during June and July 2022. The dataset encompasses information from 130,112 orders across 12 different lines connecting Xiamen with six surrounding cities. The surrounding cities vary in the distance to Xiamen Island. The city farthest from Xiamen is Xianyou, which is 169 km away, while the nearest city Longhai is only 37 km away. To cater to the intercity travel demands within the city cluster, a fleet of 291 vehicles is deployed and operated for intercity ride-pooling services in reality. Fig. 15 shows the distribution of origins and destinations of all intercity ride-pooling orders within an hour in the operational data.

In this subsection, we replicate the realistic operational settings by utilizing a simulator, and we also generate the training dataset based on the passenger behavior observed in the realistic demand data by employing several data-driven techniques. First, we calculate the average intensity of newly emerging demands for each line within every 10 min, based on the realistic demand data. Using these time-dependent parameters, we generate orders in the simulation environment by modeling the order arrival process as a non-homogeneous Poisson process. Additionally, we project the latitude and longitude of each realistic order's origin and destination into Cartesian coordinates, from which we sample the location information for orders in the simulator. The travel time between origin and destination is calculated using the Euclidean distance metric. However, to account for the detouring behavior observed in the operational data, we calibrate the detour ratio in urban areas and on intercity roads respectively based on realistic trajectory

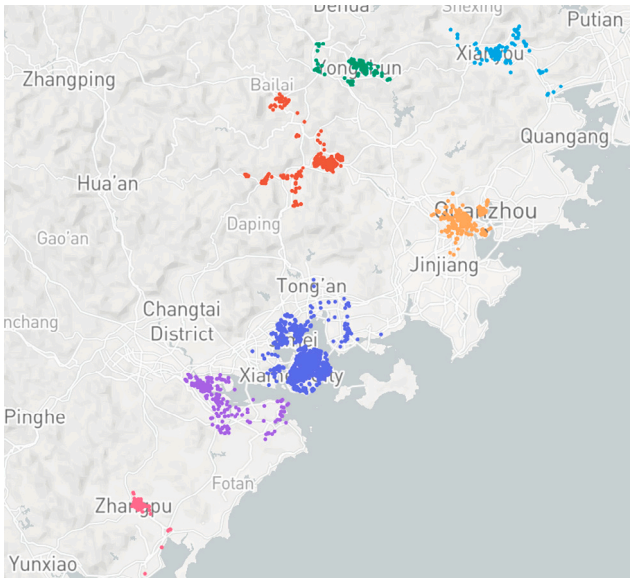

Fig. 15. Distribution of realistic demand origins and destinations within an hour.

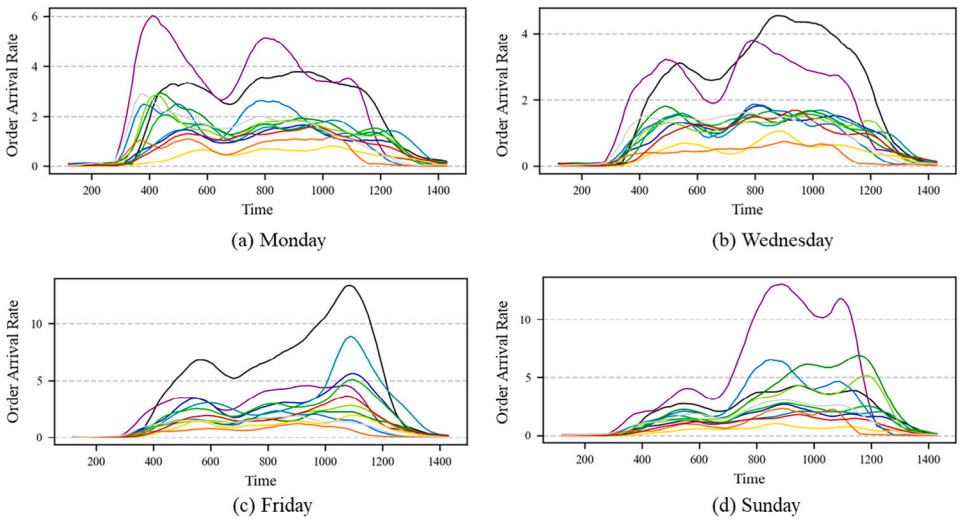

Fig. 16. Daily fluctuation of demand for each line. (a) Monday. (b) Wednesday. (c) Friday. (d) Sunday.

data. This adjustment ensures that the simulated travel times reflect the real-world detouring patterns. We also generate a hard time window for picking each order. The width of the pick-up time window follows a normal distribution with a mean of 60 min. To ensure a satisfactory passenger experience and avoid excessive detours, we set a latest arrival time for each order.

The one-way pooled-ride trip fare of each line in the simulator is consistent with the average level in reality, ranging from 35 to 100 for different lines. The fleet size in the simulation environment is also consistent with the average level in reality. Due to significant variations in intercity travel demand on different days of the week, drivers' attendance rates and times to start work also exhibit differences across days. We set the number of vehicles starting service from different cities during each period of the day based on the average situation from real data. In the subsequent experiments, we used a fleet size of 192 vehicles for Monday experiments, 176 vehicles for Wednesday experiments, 215 vehicles for Friday experiments, and 221 vehicles for Sunday experiments. By utilizing the ALNS hyper-parameters settings during training, the average total computational time for each episode is 34.1 s. Notice that in the experiments of realistic scenarios, each episode consists of 60 horizons of vehicle dispatching and routing for 12 lines, so on average 0.047 s suffice to complete one iteration of fleet dispatching and vehicle routing for a single line. This indicates that from the perspective of computational time, our algorithm has ample potential to be applied in real-time operations.

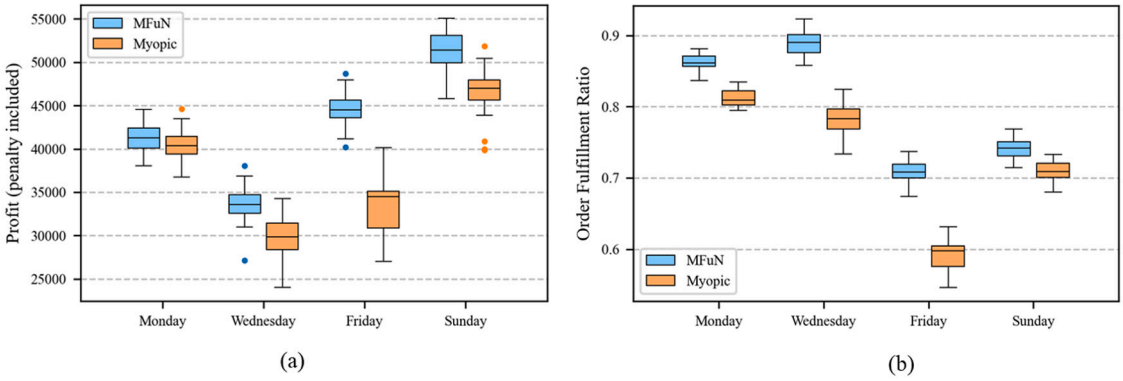

Fig. 17. Performance of MFuN and Myopic in the realistic network. (a) Profit. (b) Order fulfillment ratio.

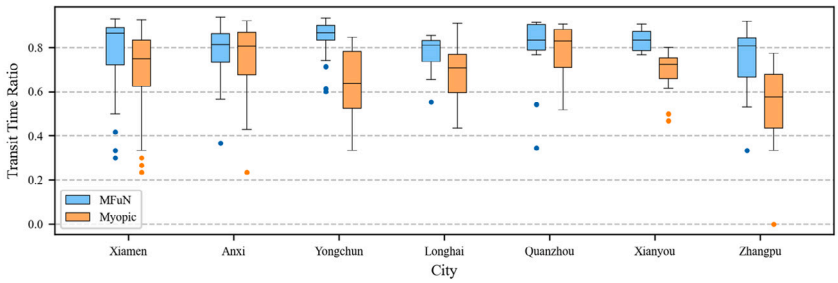

Fig. 18. Average percentage of time in transit for vehicles originating from each city under the Sunday scenario.

5.4.1. Performance under different demand and supply scenarios

The operational data reveals that the intercity travel demands for each line follow a weekly cycle. The travel demands on the same weekday of each week exhibit similar patterns, while those on different weekdays show significant variations. Fig. 16 illustrates the daily fluctuations of emerging order numbers per 10 min for 12 lines on Monday, Wednesday, Friday, and Sunday. For each week, there is a substantial demand from surrounding cities to the central city on Sunday afternoons and early Monday mornings. Conversely, on Friday afternoons, there is an almost equal amount of demand from the central city to surrounding cities. There also exists demand peaks in the morning and evening of each day, although these peaks have lower intensity compared to the weekly fluctuation pattern. Furthermore, it is worth noting that the demand varies in scale across different lines, as shown by different curves in Fig. 16. The complex demand patterns contribute to significant spatiotemporal variations in both demand and supply. The ride-pooling platform must intelligently allocate vehicle resources among the lines and optimize the matching of supply and demand through flexible scheduling, taking into account the specific temporal characteristics of the demand patterns. We train models for fleet operations under the demand scenarios of four different weekdays respectively to illustrate the effectiveness of the proposed MFuN framework, as shown in Fig. 17.

It is observed that the MFuN outperforms the Myopic method under demand and supply patterns of various weekdays, attaining improvement on both average daily profit and order fulfillment ratio. The advantages of MFuN are particularly pronounced when there is a significant imbalance between demand and supply, such as during peak periods on Fridays. In this scenario, MFuN demonstrates a remarkable 27% improvement in daily profits compared to the Myopic method. Under the scenario of Monday where the demand peak emerges at the beginning of a day, MFuN may not have enough time to effectively allocate resources to accommodate the sudden surge in demand. Under the Wednesday scenario, MFuN attains satisfactory order fulfillment rates when demand is relatively sparse compared to the weekend.

The advantage of MFuN is reflected in the intelligent spatiotemporal allocation of vehicle resources. From a spatial perspective, MFuN dispatches all available resources in the system in a more flexible manner. Fig. 18 shows the percentage of time in transit for vehicles originating from each city under the Sunday scenario. Under the MFuN dispatching strategy, vehicles from different cities have similar transit times, indicating a balanced allocation of resources. On the other hand, under the Myopic dispatching approach, vehicles originating from cities with lower demand, such as Yongchun and Zhangpu, may experience longer idle times. The proactive dispatching strategy employed by MFuN leads to improved overall utilization of vehicle resources, which aligns with the platform operator's expectations. Furthermore, the revenue distribution among drivers is also more balanced under the MFuN framework, as demonstrated in Fig. 19, which indicates a more systematic task assignment facilitated by MFuN. These improvements highlight the effectiveness of agent cooperation within the multi-agent feudal networks, further enhancing the operational outcomes of the ride-pooling platform.

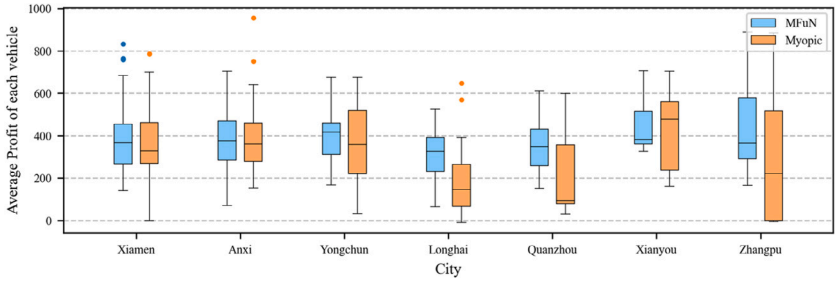

Fig. 19. Average daily revenue for drivers originating from each city under the Sunday scenario.

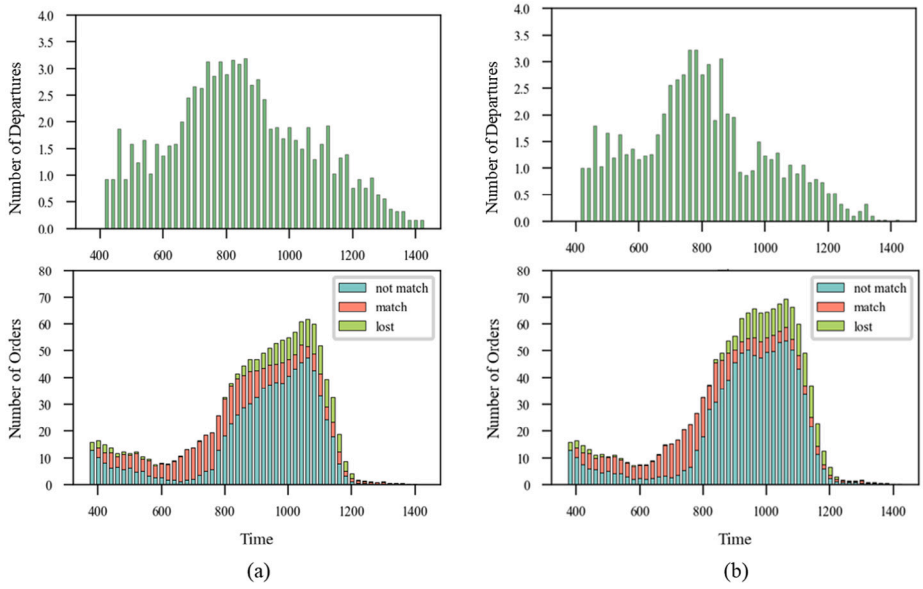

Fig. 20. Number of vehicle departures and orders for a line in each period. (a) MFuN. (b) Myopic.

From a temporal scheduling perspective, MFuN demonstrates its ability to identify and respond to potential fluctuations in demand and supply within specific cities (agents). This adaptability allows MFuN to make informed decisions based on anticipated dynamics. Fig. 20 presents the number of departures and orders for a line with a demand peak in the afternoon. Under the MFuN framework, the demand peak of this line is recognized, prompting the dispatching of a greater number of vehicles to the originating city. As a result, vehicles can depart continuously during peak periods to fulfill orders. In contrast, the Myopic method dispatches only a limited number of vehicles to the originating city during the peak periods, leading to a shortage of available vehicles and a higher rate of order loss in subsequent periods.

5.4.2. Sensitivity analysis of supply and demand level

In this subsection, we perform sensitivity analyses to evaluate the performance of fleet operational strategies under varying supply and demand levels. The experiments on different supply levels vary on the number of vehicles applied in each line. Five supply quantity levels are set, ranging from 60% to 140% based on the fleet under the Sunday scenario in realistic operational data. The performance of both methods in terms of average daily profit and order fulfillment ratio is presented in Fig. 21. A series of experiments are also conducted to explore the impact of different demand densities on the performance of the fleet operational strategies. The demand densities vary from shrinking by 40% to amplifying by 40% based on the demand dynamics under the Sunday scenario in realistic operational data. The performance comparisons under various demand levels are shown in Fig. 22.

These two series of experiments reveal consistent patterns in the performance of the two methods. The MFuN framework demonstrates superior performance over the Myopic method when the demand levels are relatively low. This advantage arises from the ability of MFuN to intelligently allocate vehicles to high-demand lines, thereby maximizing the system's profit. Even in scenarios where the overall demand level is low or the supply level is high, there are still peak periods and lines that lack sufficient vehicles. In such cases, MFuN excels in optimizing the allocation of vehicles to high-demand lines, whereas the Myopic method often results in a large number of idle vehicles in low-demand regions.

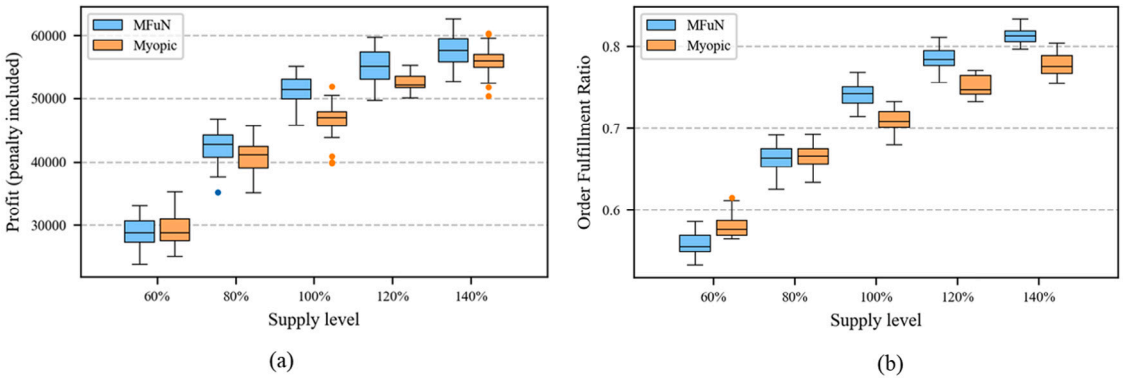

Fig. 21. Performance of MFuN and Myopic under different supply levels. (a) Profit. (b) Order fulfillment ratio.

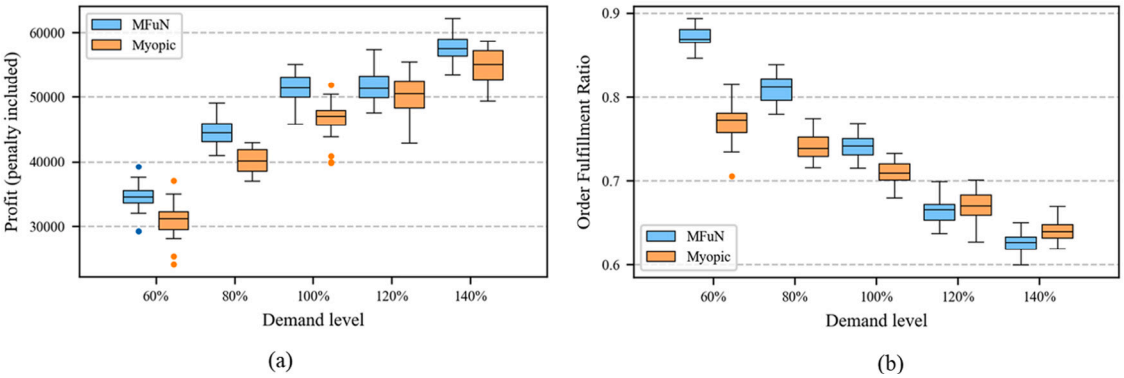

Fig. 22. Performance of MFuN and Myopic under different demand levels. (a) Profit. (b) Order fulfillment ratio.

However, under conditions where the supply level is relatively low or the demand level is relatively high, vehicle resources can always be in high utilization. In these situations, the Myopic method already guarantees that vehicles are utilized to their fullest capacity, leaving little room for MFuN to further improve the matching of demand and vehicle resources.

6. Conclusion

This study addresses the challenging problem of online fleet operations in on-demand intercity ride-pooling services within city clusters, characterized by a complex embedded structure. The fleet operations involve assigning idle vehicles in each city to lines and optimizing the vehicle routing for each line dynamically. The assignment of vehicles to intercity lines can be regarded as a stochastic dynamic vehicle resource allocation problem, while the online order-matching and vehicle routing process is a variant of dynamic DARP. The integrated online optimization framework of vehicle resource allocation and dynamic vehicle routing is barely touched in the literature.

To address this gap, we propose a two-level framework that combines reinforcement learning and heuristic approaches. A novel multi-agent hierarchical reinforcement learning method named multi-agent feudal networks is adopted in the upper level of the framework. Each worker agent corresponds to a city and collaboratively makes decisions on vehicle assignments under the guidance of a manager agent. The outputs of the worker agents, represented in a multi-discrete manner, are interpreted and mapped by a mixed-integer linear programming (MILP) formulation that considers the heterogeneity of vehicles. In the lower level of the framework, we solve the dynamic DARP using an adaptive large neighborhood search heuristic based on the assigned vehicles for each line.

We conduct extensive numerical experiments to evaluate the effectiveness of our proposed framework based on two toy networks and a realistic network in Xiamen. The experiments on the toy networks demonstrate the efficacy of our framework in temporal scheduling and spatial allocation. Furthermore, our proposed framework outperforms the myopic method across various demand and supply scenarios in the realistic network with 7 cities and 12 lines, attaining improvements in average daily profit and order fulfillment ratio. The optimization effect of MFuN is primarily observed in mitigating the spatial–temporal imbalance between supply and demand, with relatively balanced scenarios showing fewer advantages for MFuN.

In future studies, we plan to extend the scope of our research from on-demand orders to include orders with different reservation lead times. The platform can design price mechanisms to influence the stochastic demand and improve the fleet operation strategy

Table A.1
Notations for the fleet dispatching and vehicle routing model.

Notations	Explanation
Sets	
$\mathcal{U}$	The set of cities
$\mathcal{L}$	The set of intercity lines
$\mathcal{T}$	The set of dispatching horizons
$\mathcal{K}_u^t$	The set of available vehicles for assignment at horizon t at city u
$\hat{\mathcal{K}}_u^t$	The set of vehicles that newly enter the system at horizon t at city u
$\tilde{\mathcal{K}}_u^t$	The set of vehicles that exit the system at horizon t at city u
$\check{\mathcal{K}}_u^t$	The set of arrived vehicles at horizon t at city u
$\bar{\mathcal{K}}_u^t$	The set of reserved vehicles at horizon t at city u
$\tilde{\mathcal{K}}_u^t$	The set of vehicles that exit the system due to reaching limited work duration at horizon t at city u
$\mathcal{N}_{uv}^t$	The set of vehicles dispatched from city u to city v at horizon t
$\mathcal{K}_{uv}$	The set of vehicles en route of line (u, v)
$\mathcal{O}_{uv}$	The set of orders in the matching pool and in service for line (u, v)
$\mathcal{O}_{uv}^1$	The set of orders for line (u, v) that passengers have been picked
$\mathcal{O}_{uv}^2$	The set of orders for line (u, v) that have been matched but not picked
$\mathcal{O}_{uv}^3$	The set of orders in matching pool of line (u, v)
$\mathcal{V}_k$	The set of vehicle locations at the beginning of a matching interval
$\mathcal{V}_f$	The set of destination nodes of all orders
$\mathcal{V}_s$	The set of origin nodes of the unserved orders
Parameters	
$\bar{W}$	Maximum daily work time for a driver
R^{uv}	Trip fare of line (u, v)
C	Traveling cost per unit of distance
p_e	The penalty rate for losing one passenger
v_k	Location of vehicle k at the beginning of a matching interval
n_p	Number of passengers of order p
s_p	Origin node of order p
f_p	Destination node of order p
$[\check{L}_p, \check{U}_p]$	Time window of origin node of order p
$[\hat{L}_p, \hat{U}_p]$	Time window of destination node of order p
k_p	Matched vehicle of order $p \in O_{uv}^1 \cup O_{uv}^2$
E	The location of the depot in the destination city
D_{ij}	The distance of arc (i, j) in the graph for routing
W	The capacity of the vehicle
s	The speed of the vehicle
Variables	
s_k^t	The number of passengers that vehicle k picked in the trip begins at horizon t
e_{uv}^t	The number of lost passengers in line (u, v) at horizon t
δ_k^t	The total distance of the trip of vehicle k begins at horizon t
e_{uv}^t	The number of lost passengers in line (u, v) at horizon t
Decision variables	
x_{ij}^{pk}	Binary variable indicating whether vehicle k passes arc (i, j) with passengers of order p
y_{ij}^k	Binary variable indicating whether vehicle k passes arc (i, j)
d_{pk}	Binary variable indicating whether vehicle k serves order p
u_{kj}	The time for vehicle k to arrive at node j

to provide services to hybrid demand including on-demand orders and orders reserved in advance. Another area of investigation is the planning of intercity ride-pooling services within city clusters, which encompasses determining the optimal fleet size, pricing mechanisms, and operation lines. This planning can be constructed on the market equilibrium of intercity passenger transport, taking into account the competition with other intercity transit modes.

CRediT authorship contribution statement

Jinhua Si: Conceptualization, Methodology, Validation, Visualization, Writing – original draft, Writing – review & editing, Data curation, Investigation. **Fang He:** Conceptualization, Methodology, Resources, Supervision, Validation, Writing – review & editing, Funding acquisition, Investigation. **Xi Lin:** Conceptualization, Investigation, Methodology, Supervision, Validation, Writing – original draft. **Xindi Tang:** Conceptualization, Methodology, Supervision, Validation, Writing – review & editing, Funding acquisition.

Declaration of competing interest

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

Data availability

The authors do not have permission to share data.

Acknowledgments

This research was supported partially by grants from the National Natural Science Foundation of China [grant numbers 72371141, 72301305] and the Seedling Scholars Support Program of Central University of Finance and Economics [grant number XMXZ2306].

Appendix. Notations

See Table A.1

References

Agatz, N., Erera, A., Savelsbergh, M., Wang, X., 2012. Optimization for dynamic ride-sharing: A review. European J. Oper. Res. 223 (2), 295–303.
Ahilan, S., Dayan, P., 2019. Feudal multi-agent hierarchies for cooperative reinforcement learning. arXiv preprint arXiv:1901.08492.
Alonso-Mora, J., Wallar, A., Rus, D., 2017. Predictive routing for autonomous mobility-on-demand systems with ride-sharing. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, IEEE, Vancouver, BC, pp. 3583–3590.
Bongiovanni, C., Kaspi, M., Cordeau, J.-F., Geroliminis, N., 2022. A machine learning-driven two-phase metaheuristic for autonomous ridesharing operations. Transp. Res. E: Logist. Transp. Rev. 165, 102835.
Braekers, K., Kovacs, A.A., 2016. A multi-period dial-a-ride problem with driver consistency. Transp. Res. B 94, 355–377.
Cordeau, J., 2006. A branch-and-cut algorithm for the dial-a-ride problem. Oper. Res. 54 (4), 573–586.
Czioska, P., Kutadinata, R., Trifunović, A., Winter, S., Sester, M., Friedrich, B., 2019. Real-world meeting points for shared demand-responsive transportation systems. Public Transp. 11, 341–377.
Dayan, P., Hinton, G.E., 1992. Feudal reinforcement learning. Adv. Neural Inf. Process. Syst. 5.
Demir, E., Bektaş, T., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the pollution-routing problem. European J. Oper. Res. 223 (2), 346–359.
Fang, C., Yu, D., 2017. Urban agglomeration: An evolving concept of an emerging phenomenon. Landsc. Urban Plan. 162, 126–136.
Fielbaum, A., Bai, X., Alonso-Mora, J., 2021. On-demand ridesharing with optimized pick-up and drop-off walking locations. Transp. Res. C 126, 103061.
Ganji, S., Ahangar, A., Awasthi, A., Jamshidi Bandari, S., 2021. Psychological analysis of intercity bus passenger satisfaction using Q methodology. Transp. Res. A 154, 345–363.
Ghilas, V., Demir, E., Van Woensel, T., 2016. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. Comput. Oper. Res. 72, 12–30.
Godfrey, G.A., Powell, W.B., 2002. An adaptive dynamic programming algorithm for dynamic fleet management, II: Multiperiod travel times. Transp. Sci. 36 (1), 40–54.
Goeke, D., 2019. Granular tabu search for the pickup and delivery problem with time windows and electric vehicles. European J. Oper. Res. 278 (3), 821–836.
Gschwind, T., Drexl, M., 2019. Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. Transp. Sci. 53 (2), 480–491.
Gschwind, T., Irnich, S., 2015. Effective handling of dynamic time windows and its application to solving the dial-a-ride problem. Transp. Sci. 49 (2), 335–354.
Guo, X., Caros, N.S., Zhao, J., 2021. Robust matching-integrated vehicle rebalancing in ride-hailing system with uncertain demand. Transp. Res. B 150, 161–189.
Guo, J., Long, J., Xu, X., Yu, M., Yuan, K., 2022. The vehicle routing problem of intercity ride-sharing between two cities. Transp. Res. B 158, 113–139.
Guo, G., Xu, Y., 2020. A deep reinforcement learning approach to ride-sharing vehicle dispatching in autonomous mobility-on-demand systems. IEEE Intell. Transp. Syst. Mag. 14 (1), 128–140.
Guo, G., Zhang, T., 2020. A residual spatio-temporal architecture for travel demand forecasting. Transp. Res. C 115, 102639.
Guo, Y., Zhang, Y., Yu, J., Shen, X., 2020. A spatiotemporal thermo guidance based real-time online ride-hailing dispatch framework. IEEE Access 8, 115063–115077.
Ho, S.C., Szeto, W.Y., Kuo, Y.H., Leung, J.M.Y., Petering, M., Tou, T.W.H., 2018. A survey of dial-a-ride problems: Literature review and recent developments. Transp. Res. B 111, 395–421.
Huang, D., Gu, Y., Wang, S., Liu, Z., Zhang, W., 2020. A two-phase optimization model for the demand-responsive customized bus network design. Transp. Res. C 111, 1–21.
Jiao, Y., Tang, X., Qin, Z.T., Li, S., Zhang, F., Zhu, H., Ye, J., 2021. Real-world ride-hailing vehicle repositioning using deep reinforcement learning. Transp. Res. C 130, 103289.
Jin, J., Zhou, M., Zhang, W., Li, M., Guo, Z., Qin, Z., Jiao, Y., Tang, X., Wang, C., Wang, J., et al., 2019. Coride: joint order dispatching and fleet management for multi-scale ride-hailing platforms. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. pp. 1983–1992.
Ke, J., Zheng, H., Yang, H., Chen, X.M., 2017. Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. Transp. Res. C 85, 591–608.
Kullman, N.D., Cousineau, M., Goodson, J.C., Mendoza, J.E., 2022. Dynamic ride-hailing with electric vehicles. Transp. Sci. 56 (3), 775–794.
Lee, E., Cen, X., Lo, H.K., 2021. Zonal-based flexible bus service under elastic stochastic demand. Transp. Res. E: Logist. Transp. Rev. 152, 102367.
Lei, C., Jiang, Z.T., Ouyang, Y.F., 2020. Path-based dynamic pricing for vehicle allocation in ridesharing systems with fully compliant drivers. Transp. Res. B 132, 60–75.
Liu, Y., Ouyang, Y., 2021. Mobility service design via joint optimization of transit networks and demand-responsive services. Transp. Res. B 151, 22–41.
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. Transp. Res. E: Logist. Transp. Rev. 161, 102694.

Lowe, R., Wu, Y.I., Tamar, A., Harb, J., Pieter Abbeel, O., Mordatch, I., 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. Adv. Neural Inf. Process. Syst. 30.
Luo, Z., Liu, M., Lim, A., 2019. A two-phase branch-and-price-and-cut for a dial-a-ride problem in patient transportation. Transp. Sci. 53 (1), 113–130.
Ma, Z., Koutsopoulos, H.N., 2022. Near-on-demand mobility. The benefits of user flexibility for ride-pooling services. Transp. Res. C 135, 103530.
Mao, C., Liu, Y., Shen, Z.-J.M., 2020. Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach. Transp. Res. C 115, 102626.
Melis, L., Sörensen, K., 2022a. The real-time on-demand bus routing problem: The cost of dynamic requests. Comput. Oper. Res. 147, 105941.
Melis, L., Sörensen, K., 2022b. The static on-demand bus routing problem: large neighborhood search for a dial-a-ride problem with bus station assignment. Int. Trans. Oper. Res. 29 (3), 1417–1453.
MOT, 2022. Statistical communiqué of the People's Republic of China on the transport industry. Accessed 25 May 2022. https://xxgk.mot.gov.cn/2020/jigou/zhghs/202205/t20220524_3656659.html.
Naccache, S., Côté, J.-F., Coelho, L.C., 2018. The multi-pickup and delivery problem with time windows. European J. Oper. Res. 269 (1), 353–362.
Ouyang, Y., Yang, H., Daganzo, C.F., 2021. Performance of reservation-based carpooling services under detour and waiting time restrictions. Transp. Res. B 150, 370–385.
Qin, G., Luo, Q., Yin, Y., Sun, J., Ye, J., 2021a. Optimizing matching time intervals for ride-hailing services using reinforcement learning. Transp. Res. C 129, 103239.
Qin, Z., Tang, X., Jiao, Y., Zhang, F., Xu, Z., Zhu, H., Ye, J., 2020. Ride-hailing order dispatching at didi via reinforcement learning. INFORMS J. Appl. Anal. 50 (5), 272–286.
Qin, X., Yang, H., Wu, Y., Zhu, H., 2021b. Multi-party ride-matching problem in the ride-hailing market with bundled option services. Transp. Res. C 131, 103287.
Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 40 (4), 455–472.
Schwieterman, J., Antolin, B., Bell, C., 2021. On the brink: 2021 outlook for the intercity bus industry in the united states.
Simonetto, A., Monteil, J., Gambella, C., 2019. Real-time city-scale ridesharing via linear assignment problems. Transp. Res. C 101, 208–232.
Sun, Q., Chien, S., Hu, D., Chen, G., Jiang, R.-S., 2020b. Optimizing multi-terminal customized bus service with mixed fleet. IEEE Access 8, 156456–156469.
Sun, P., Veelenturf, L.P., Hewitt, M., Van Woensel, T., 2020a. Adaptive large neighborhood search for the time-dependent profitable pickup and delivery problem with time windows. Transp. Res. E-Logist. Transp. Rev. 138.
Tafreshian, A., Abdolmaleki, M., Masoud, N., Wang, H., 2021. Proactive shuttle dispatching in large-scale dynamic dial-a-ride systems. Transp. Res. B 150, 227–259.
Tang, X.D., Li, M., Lin, X., He, F., 2020. Online operations of automated electric taxi fleets: An advisor-student reinforcement learning framework. Transp. Res. C 121.
Tang, X., Qin, Z., Zhang, F., Wang, Z., Xu, Z., Ma, Y., Zhu, H., Ye, J., 2019. A deep value-network based approach for multi-driver order dispatching. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1780–1790.
Tong, Y., She, J., Ding, B., Chen, L., Wo, T., Xu, K., 2016. Online minimum matching in real-time spatial data: experiments and analysis. Proc. VLDB Endow. 9 (12), 1053–1064.
Tsai, T.-H., 2020. Self-evolutionary sibling models to forecast railway arrivals using reservation data. Eng. Appl. Artif. Intell. 96, 103960.
Tuncel, K., Koutsopoulos, H.N., Ma, Z., 2023. An integrated ride-matching and vehicle-rebalancing model for shared mobility on-demand services. Comput. Oper. Res. 159, 106317.
Vansteenwegen, P., Melis, L., Aktaş, D., Montenegro, B.D.G., Vieira, F.S., Sörensen, K., 2022. A survey on demand-responsive public bus systems. Transp. Res. C 137, 103573.
Vezhnevets, A.S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., Kavukcuoglu, K., 2017. Feudal networks for hierarchical reinforcement learning. In: International Conference on Machine Learning. PMLR, pp. 3540–3549.
Wang, H., Yang, H., 2019. Ridesourcing systems: A framework and review. Transp. Res. B 129, 122–155.
Wu, Y., Poon, M., Yuan, Z., Xiao, Q., 2022. Time-dependent customized bus routing problem of large transport terminals considering the impact of late passengers. Transp. Res. C 143, 103859.
Xu, Z., Li, Z., Guan, Q., Zhang, D., Li, Q., Nan, J., Liu, C., Bian, W., Ye, J., 2018a. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 905–913.
Xu, Z., Li, Z.X., Guan, Q.W., Zhang, D.S., Li, Q., Nan, J.X., Liu, C.Y., Bian, W., Ye, J.P., Acm, 2018b. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In: 24th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. KDD, pp. 905–913.
Yang, H., Qin, X., Ke, J., Ye, J., 2020. Optimizing matching time interval and matching radius in on-demand ride-sourcing markets. Transp. Res. B 131, 84–105.
Yu, X., Shen, S., 2020. An integrated decomposition and approximate dynamic programming approach for on-demand ride pooling. IEEE Trans. Intell. Transp. Syst. 21 (9), 3811–3820.