Transportation Research Part C 148 (2023) 104030

Contents lists available at ScienceDirect

Transportation Research Part C

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

Dynamic vehicle routing problem for flexible buses considering stochastic requests☆

Wanjing Ma [a,b], Lin Zeng [a], Kun An [b,*]

[a] Urban Mobility Institute, Tongji University, Shanghai, China
[b] Key Laboratory of Road and Traffic Engineering of Ministry of Education, Tongji University, 4800 Cao'an Road, Shanghai 201804, China

ARTICLE INFO

Keywords:
Flexible bus
Dynamic routing
Passenger–vehicle matching
Two-stage stochastic programming
Adaptive large neighbourhood search

ABSTRACT

Flexible buses provide on-demand services to one or more local communities in a specific geographical area. Bus routes can be adjusted dynamically according to real-time passenger demand in a cost-effective manner. This study investigated the dynamic bus-routing problem considering stochastic future passenger demand. A two-stage stochastic programming model was formulated to minimise the total vehicle travel time cost and penalty for rejecting requests. A rolling horizon scheme was adopted to handle the dynamic changes in passenger requests and vehicle routes. A vector-similarity-based clustering and adaptive large neighbourhood searching (VSC-ALNS) algorithm was developed to solve this problem. Vehicles and passengers were matched and clustered into groups based on vector similarity, and vehicle routes were generated using an adaptive large-neighbourhood search algorithm for each cluster. The effectiveness of the proposed method was evaluated in four cases with different demand intensities using Shanghai taxi order data. The results indicate that flexible buses are more suitable for moderate demand cases, ranging from 20 to 50 requests per square kilometre per hour.

1. Introduction

Ground public transport is as one of the major modes of urban mobility, serving over 51 % passenger trips in Hong Kong, 25 % in Shanghai, and 18 % in London. Bus services can be divided into two categories: fixed route transit (FRT) and demand-responsive transit (DRT) (Engels, 2004). The FRT operates on fixed routes with fixed schedules that are unresponsive to actual demands and may cause waste of seats. In contrast, DRT provides demand-responsive services, such as airport shuttle services, and support for disadvantaged groups (Daganzo, 1984; Li and Quadrifoglio, 2009).

The most representative problem in DRT systems is the dial-a-ride problem (DARP), which aims to determine vehicle routes and timetables for passengers who specify their requests via telephone (Ho et al., 2018). DARP has various applications, including personnel transportation for elderly and disabled persons and airport shuttle services. Passengers with similar route and time preferences can share the same vehicle according to the vehicle capacity. The objectives of DARP include minimising the vehicle travel distance and time under the constraints of the vehicle capacity. Moreover, a certain period of advance booking is necessary for availing services under DARP.

☆ This article belongs to the Virtual Special Issue on "Emerging On-Demand Passenger and Logistics Systems: Modelling, Optimization, and Data Analytics".

* Corresponding author.
E-mail address: kunan@tongji.edu.cn (K. An).

https://doi.org/10.1016/j.trc.2023.104030
Received 30 June 2022; Received in revised form 6 December 2022; Accepted 14 January 2023
Available online 31 January 2023

Ride sharing is a particular form of shared mobility, which is different from that of DARP. In DARP, drivers are professionally qualified and operate out of one or more stations, whereas in ride sharing, each driver drives a private vehicle and only serves passengers on similar routes. In other words, ride sharing facilitates shared rides between drivers and passengers with similar origin–destination pairings. A typical example is carpooling, which determines carpool people with the same origin and/or destination, usually-one day or several hours in advance. Once determined, the vehicle route does not change during the entire trip until all the carpool passengers arrive. Passenger-preferred time window constraints should also be considered (Carotenuto and Martis, 2017). Traditional carpooling often requires long-term commitment, and is not a DRT service. Technological advancements in the mobile Internet services have significantly boosted the development of large-scale online shared-mobility platforms, and on-demand responsive real-time ride-sourcing services have become possible. Ride-sourcing companies provide prearranged and on-demand transportation services through online platforms that connect private drivers with passengers (Shaheen and Cohen, 2019). Ride sourcing includes the use of vehicles with others and the use of vehicles alone. Ride sourcing with pooling, also known as ride splitting, is an emerging shared mobility service in which private vehicle owners provide on-demand services (Jin et al., 2018). Ride splitting allows passengers with a similar route to share a ride and split fares. A driver can accept a passenger who is willing to share the ride first and then seek the second passenger on the way. Whether to serve a second passenger is determined by the path similarity between the two passengers. Hence, the vehicle route is significantly restricted by the first passenger, and the ride-splitting platform seeks a second passenger with a similar destination or route that coincides with the first passenger in the course of serving the first passenger. One vehicle trip ends when all rideshare passengers arrive at their destinations and the vehicle becomes available. The objective typically maximises the matching rate between vehicles and passengers (Zhan et al., 2021) considering the maximum detour constraints. Here, Fig. 1 illustrates the categories of shared mobility.

Recently, DRT systems comprising buses and vans have emerged. These services include customised and flexible buses. A customised bus collects passengers' travel demands in advance, usually-one month earlier, and customises bus routes by aggregating similar demands. The objectives include vehicle distance or travel cost minimisation (Becker et al., 2020; Lv et al., 2020; Winter and Nittel, 2006), or fleet size minimisation while satisfying vehicle capacity constraints (Ben-Dor et al., 2019; Liang et al., 2016). Huang et al. (2020) proposed a new optimisation model for the network design problem of a demand-responsive customised bus consisting of dynamic and static phases.

A flexible bus can be regarded as a DRT service provider with online booking for the general public (Flusberg, 1976). Passengers send their travel requests through an online platform and buses serve passengers according to the routes generated by the platform. A flexible bus picks up and drops off passengers flexibly in areas without being constrained by routes. Thus, flexible buses are more cost-efficient than taxies and more convenient than conventional fixed-route buses (Atasoy et al., 2015; Liu and Ceder, 2015). The key is to match passengers and vehicles as well as plan vehicle routes, with the aim of minimising operating costs while ensuring low waiting and travel times for passengers. There have been a large number of variants in flexible bus services. Its operating modes vary according to the vehicle type, that is, passenger car, minibus, and coach, passenger reservation scheme, that is, whether reservation is required or not and how long in advance, and frequency of bus-route adjustment, that is, every day or every minute.

Compared to ride sourcing, a flexible bus usually assigns several passengers to one bus simultaneously. Passengers get on and off continuously, and therefore, there is no "first passenger" or specific ending time for the bus trip, thereby reducing empty driving miles. Moreover, the flexible bus service typically uses minibuses with a larger capacity, whereas ride sourcing uses private cars. An increase in the vehicle capacity makes it more difficult to solve the vehicle-routing problem. Compared to traditional ridesharing, a flexible bus does not require making a commitment in advance to form a fixed carpool, and can rely on the platform to form a real-time carpool. Compared to a customised bus, a flexible bus can change its route in real time and serve dynamic demands. Based on whether all request information is known before making a decision, the flexible bus service routing problem is classified into static and dynamic (Ho et al., 2018) problems. In the static problem, all vehicles and passengers must provide travel information before the system optimises the matches (Xiao et al., 2015; Xu et al., 2015). Therefore, the operator has sufficient time to determine the optimal bus schedule. In contrast, the dynamic problem requires that the travel information of vehicles and passengers is sent to the system in real time (Duan et al., 2020; Tafreshian and Masoud, 2020). The vehicle schedule is often updated in real time; thus, the problem is more

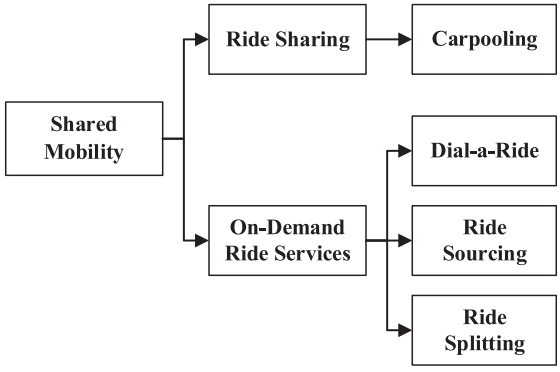

Fig. 1. Categories of shared mobility.

computationally challenging. Previous studies have adopted rolling horizon optimisation to address the dynamic nature of the problem (Agatz et al., 2011; Liang et al., 2020; SteadieSeifi et al., 2021; Syed et al., 2019). The operation of an entire day is discretised into several time slices. Rolling horizon optimisation solves the vehicle–passenger matching for each time slice, converting the dynamic problem into a static problem. However, rolling horizon optimisation only utilises the information in the current time slice, which is often short-sighted, and the impact of future requests on the vehicle route is overlooked, resulting in vehicle detours. Therefore, it is necessary to consider deterministic requests in the current time slice and stochastic requests in the future to alleviate this short-sighted problem.

A problem is deterministic if the information received is certain when making a decision; otherwise, it is stochastic (Soeffker et al., 2022). For example, requests obtained through reservations are deterministic, and future requests that have not yet occurred are stochastic (An, 2020). The flexible bus routing problem generally involves randomness in the request information, including travel time uncertainty (Liang et al., 2018; Liu et al., 2020; Vodopivec and Miller-Hooks, 2017), departure or arrival location uncertainty (Zhao et al., 2021), and passenger number uncertainty (Guo et al., 2021). Some existing studies have investigated the static bus routing and scheduling problem considering the randomness in passenger demand and travel time. The uncertainty parameters are assumed to follow a certain probability distribution, which is available to decision-makers (Zhou et al., 2019). Li et al. (2016) proposed two two-stage stochastic programming models that considered the stochastic travel time and delivery locations. An adaptive large neighbourhood search algorithm was used to solve the problem. Lee et al. (2021) proposed a zonal-based flexible bus service mode that considered the stochastic demand and detour time. An and Lo (2014) proposed a service reliability-based formulation for public transport networks. Another group of studies focused on the dynamic vehicle-routing problem under request information uncertainty. Schilde et al. (2011) proposed four heuristic algorithms to solve the dynamic DARP under stochastic pick-up location requests. Ulmer (2019) analysed the merits of reactive route re-optimisation and the prediction of stochastic requests in a dynamic vehicle-routing scheme. Ning et al. (2021) solved the first-mile transportation problem that minimised passenger waiting and travel times using an improved ant-colony algorithm under a mixture of stochastic and deterministic requests. Vehicle routing considering demand randomness was investigated for the dial-a-ride system, where passengers and vehicles were matched in a first-come first-serve way, or for fixed route buses, where the demand was static. Few studies have considered the impact of future uncertain demand on the dynamic vehicle routing for flexible buses.

The dynamic flexible bus routing problem, which considers stochastic future requests, is a notorious NP-hard problem. Exact algorithms can be developed to pursue the optimal global solution, such as the branch-and-price algorithm (Hasan et al., 2020) and the branch-and-cut algorithms (Bongiovanni et al., 2019). However, exact algorithms are usually only applicable to small instances. For larger-scale cases, it fails because of the explosion of the dimensions. Heuristics or metaheuristics can search for near-optimal solutions efficiently, and thus, they have been widely adopted in existing research. These include insertion-based methods, ant-colony algorithm (Daganzo et al., 2020), and tabu search algorithm (Kirchler and Calvo, 2013). Alonso-Mora et al. (2017) proposed a greedy algorithm that dynamically generated optimal routes for real-time high-capacity ridesharing. Chen et al. (2020) proposed a dynamic vacant car-passenger meeting model and proposed an algorithm based on approximate dynamic programming, which can effectively improve the objective function. Lyu et al. (2019) proposed a customised bus-line planning framework to optimise bus stop locations, bus routes, and timetables simultaneously. A heuristic solution framework that includes a clustering method, bus stop deployment algorithm, and dynamic routing and timetabling algorithm was developed to solve the model. Solution algorithms are required for the application of large-scale real-time flexible bus systems. Researchers have also tried to use machine learning and deep learning to solve the ride sharing problem. Ke et al. (2022) established a two-stage framework to solve online matching between idle drivers and waiting passengers, which incorporates a combinatorial optimization and multi-agent deep reinforcement learning methods.

In this study, we aimed to investigate the dynamic flexible bus routing problem by considering stochastic future requests. Requests gradually appear throughout the day and must be scheduled in real-time. Buses travel along scheduled routes in the study region with passengers on-board. The system must match requests and vehicles serving passengers and reroute vehicles while fulfilling the time window constraints of both on-board passengers and new passengers. The impact of stochastic requests on generalised costs should be considered during path planning. To this end, we developed a two-stage stochastic programming model that minimised the sum of the vehicle operating costs and penalty of unserved passengers. Furthermore, we developed a vector similarity clustering method to identify request groups in similar directions. An adaptive large neighbourhood search (ALNS) was then applied to determine the vehicle routes in each cluster. The solution approach was applied to a real-world network to demonstrate solution efficiency and quality.

The main contributions of this study can be summarized as follows:

- We proposed a flexible-bus operating mode that used minibuses with median vehicle capacity. The bus route can be adjusted according to the dynamic demand, which is different from existing customised buses and ridesharing.
- We investigated a dynamic flexible bus routing problem considering deterministic current requests and uncertain future requests under time window and seat-capacity constraints. A two-stage stochastic programming model with recourse was developed to minimise the sum of the vehicle operating costs and the penalty of unserved passengers.
- We proposed a vector-similarity-based clustering and adaptive large-neighbourhood searching (VSC-ALNS) solution algorithm, wherein requests and vehicles are clustered based on vector similarity calculated using spatiotemporal distance. Subsequently, an ALNS is applied to obtain the vehicle routes.
- We illustrated the performance of the proposed solution method based on real taxi data from Shanghai. Our algorithm outperformed the other baseline algorithms in solving large-scale problems with over 3639 requests per hour.

- We compared the proposed flexible bus mode with conventional bus, taxi, ride sharing mode proposed by Alonso-Mora et al. (2017) and provided remarks on the advantages and application conditions of each mode.

The remainder of this paper is organised as follows. In Section 2, we describe the dynamic flexible bus routing problem and formulate a two-stage stochastic programming model with recourse. In Section 3, the VSC-ALNS heuristic algorithm is proposed. Section 4 presents the computational results. Finally, Section 5 concludes the study and provides suggestions for future research.

2. Model formulation

This section presents the operating mode of the flexible bus and the mathematical model formulated for the dynamic vehicle-routing problem for flexible buses.

2.1. Problem description

We considered an urban area in which flexible bus services are provided. Passengers send their requests to the operating platform through mobile applications in real-time, specifying their preferred departure time, origin, and destination. Based on the received requests and real-time vehicle status, the platform determines the sequences of passenger pick-ups and drop-offs as well as the routes of each vehicle with the aim of generalised cost minimisation. Over time, en-route vehicles with passengers on-board may have to reroute to fulfil more requests if there are extra seats. When new requests are received, the platform adjusts the vehicle routes by considering

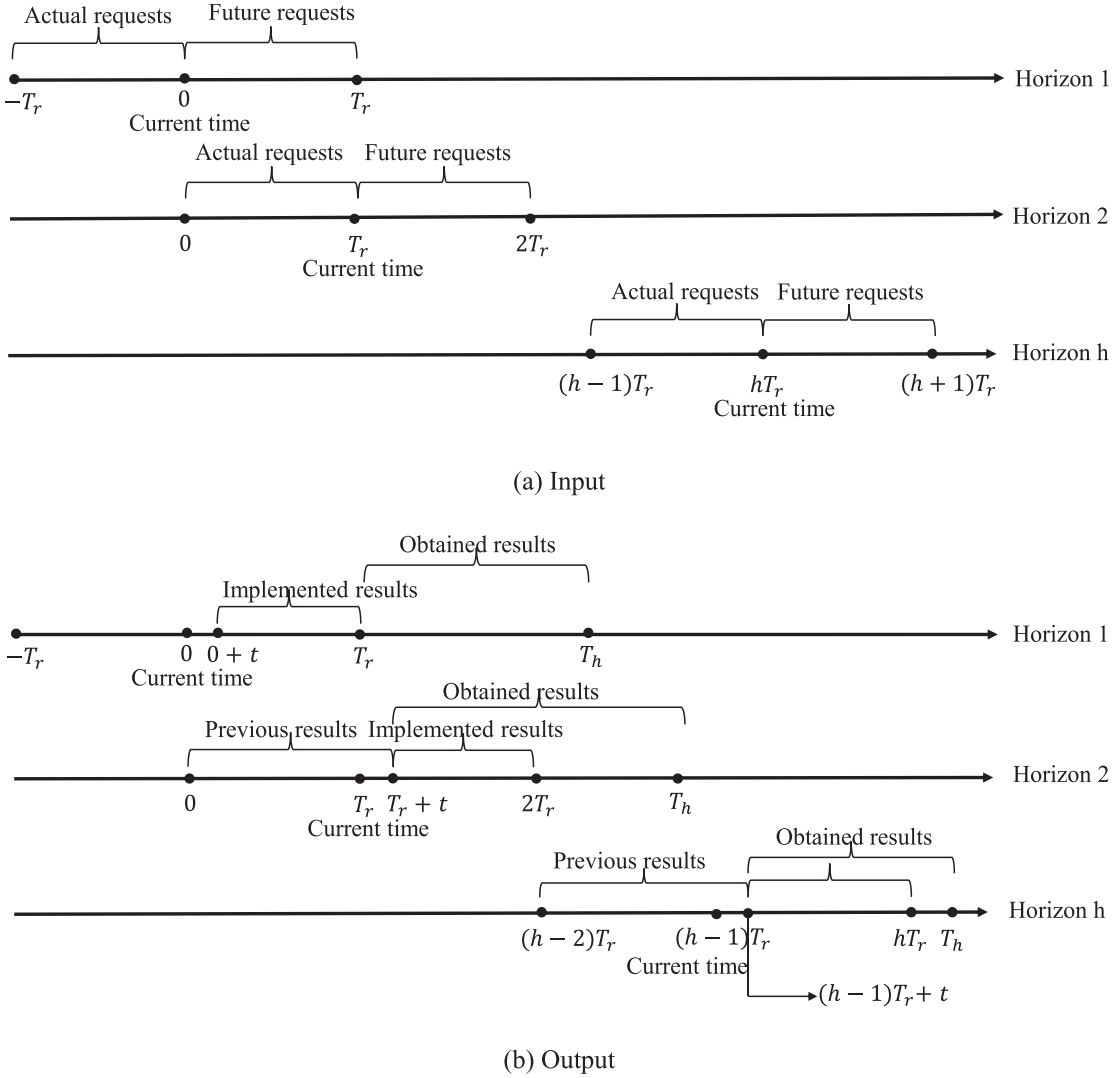

Fig. 2. Schematic of the rolling horizon optimization.

the time window constraints of the new requests and the incurred delay in the arrival time of the on-board passengers. The objective is to fulfil as many requests as possible at the lowest cost. During the optimisation process, the allocated requests from the previous steps must be served by the same vehicle as scheduled. For new requests, the platform needs to determine whether this request can be served, which vehicle to use, and the updated route of the vehicle.

Only requests until the current time are submitted by the passengers and become known to the platform. The future requests are unknown to the platform yet might have a significant impact on the efficiency of the obtained bus routes. For example, consider vehicle V and request order A, where order A is far away from vehicle V. If only the current order A is considered, the cost of serving order A by vehicle V is too high, and order A will be rejected. However, there is a high possibility that order B will occur in the next timestep, and the pickup and drop-off nodes of order B are in the route of vehicle V serving order A. At this time, serving orders A and B via vehicle V is profitable. Therefore, it is important to incorporate the impacts of future uncertain requests in the optimisation of vehicle–passenger matching and vehicle routes.

Because passengers submit requests in real time, the platform must respond as soon as possible and update the vehicle routes periodically. Here, we adopted a rolling-horizon framework to address the dynamic vehicle-routing problem. The study horizon was divided into several time steps with a duration of p min each. The platform executed the optimisation algorithm at the end of each time step and updated the status of vehicles and passengers. Requests that were scheduled or the passengers that were on-board in previous iterations were served by the same vehicle. However, new requests could be served or rejected in the current iteration, and the rejected requests would be transferred to the waiting pool until vehicle assignment in the next iteration. As passengers exit the system when the maximum waiting time δ is reached, a longer time interval p would aid in accumulating more requests in one optimisation, leading to lower operating costs at the expense of a longer passenger waiting time and longer computational time. Therefore, setting an appropriate time interval is required to strike a balance between the platform and the passengers' interests.

Here, Fig. 2 illustrates the rolling horizon optimisation considering stochastic future requests. One horizon includes two time steps, and each time step has p min. More time steps in a horizon can be considered if computational resources are sufficient. For illustration purposes, we only considered future requests in one time step. Time 0 was defined as the current moment. For horizon 1, we collected the new requests from $-T_r$ to the current time 0, current vehicle status, vehicle routes, and also considered stochastic requests that might occur from 0 to T_r as the input. Once the solutions were obtained by solving the model, the system output the vehicle-routing results from $0+t$ to T_r, as shown in Fig. 2(b), where t is the computational time. The vehicles implemented the results of the path from $0+t$ to T_r, which did not serve stochastic requests. Then, the horizon rolled forward with time length T_r and arrived at the next horizon. When it rolled to horizon h, we considered the deterministic requests submitted from $(h-1)T_r$ to hT_r and stochastic requests that might be submitted from hT_r to $(h+1)T_r$. The range of route results implemented was hT_r+t to $(h+1)T_r$.

Regarding future requests, the pickup and drop-off nodes and number of seats required could be estimated by statistical methods using historical demand data and other real-time information. With the development of deep learning, neural networks can make short-term travel origins and destinations (ODs) predictions of requests, considering spatiotemporal distribution characteristics. Qiu et al. (2019) proposed a novel contextualised spatial–temporal network to predict taxi origin–destination demand. Ke et al. (2021) proposed a spatiotemporal encoder–decoder residual multi-graph convolutional network for ride-hailing OD-based demand prediction. In the existing studies, the accuracy rate of OD prediction for inter-site requests has reached 70 %. In this study, future uncertain requests were represented as a number of probabilistic scenarios. A demand scenario was obtained using demand prediction methods based on historical order data together with real-time information. We reasonably converted the accuracy of the prediction results into a stochastic scenario set based on existing prediction methods. As demand prediction was not the focus of this study, we do not discuss the details here, but only consider the probabilistic prediction results (i.e., pickup and drop-off locations and departure time) as inputs.

Here, Fig. 3 illustrates passenger allocation and vehicle routing for flexible buses in one optimisation horizon. In the first diagram, there are two vehicles in the system. At the end of time step t, passenger $P1$, who is filled with black, is already scheduled to be taken by

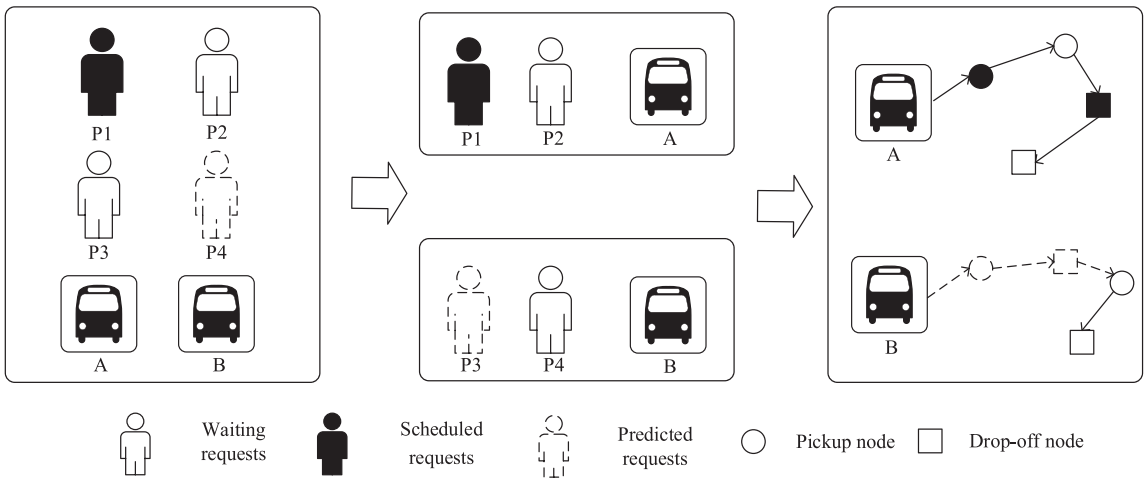

Fig. 3. Schematic of passenger allocation and vehicle routing of the flexible bus system.

vehicle A in previous time steps on or before time step $t-1$. Passengers $P2$ and $P3$ represent new requests during time step t and are deterministic requests. Passenger $P4$ represents the predicted request at time step $t+1$. After optimisation, in the second diagram, vehicle A serves passengers $P1$ and $P2$, and vehicle B serves passenger $P3$ and the possible future stochastic passenger $P4$. The third diagram shows the real driving route of the vehicle, and the route of the dashed line in vehicle B is temporarily not executed. However, when vehicle B is on the road, the stochastic request $P4$ may appear, and vehicle B will be able to serve $P4$.

2.2. Mathematical model

2.2.1. Mathematical formulation

There are two critical participants in the flexible bus system: passengers (presented by order requests) and vehicles. For each optimisation horizon, we denoted R as the set of requests in the flexible system. Each request $r \in R, r = \{1, \cdots, |R|\}$ was represented as a vector:(q_r, p_r, d_r), where q_r is the number of passengers in this request, p_r is the pickup node, and d_r is the drop-off node, and $|R|$ is the number of requests considered in this horizon. The requests were divided into three types according to their service status: (1) waiting requests; i.e., requests that were received during this time step and those that were rejected in the previous time step but still existed in the waiting pool, $r \in R_{waiting}$; (2) scheduled requests; i.e., requests that were assigned to certain vehicles. Those passengers could already be on-board or could be waiting to be picked up by the assigned vehicle, $r \in R_{scheduled}$; (3) predicted requests; i.e., requests that might occur in the future, $r \in R_{predicted}$. There were another two groups of requests, that is, the finished requests that were delivered and the lost requests that had left the system as they exceeded the waiting time threshold. Because these passengers were no longer in the system and could not affect the vehicle routes, we simply excluded them from the mathematical model.

The attributes of a vehicle include its current position, capacity, and scheduled routes. We denoted K as the set of vehicles available during the current time step. Each vehicle $k \in K$, $k = \{1, \cdots, |K|\}$ was represented as a vector: $(q^k_{max}, V^k_o, V^k_d)$, where q^k_{max} denotes the remaining number of seats, V^k_o represents the vehicle's current position, V^k_d represents the virtual destination of the vehicle, and $|K|$ is the number of vehicles. A vehicle returns to a virtual destination after an entire day of operation. The distance from the other locations to the virtual destination was set to 0.

To describe the dynamic vehicle route in the programming language better, we unified the requests and vehicles in a fully con-

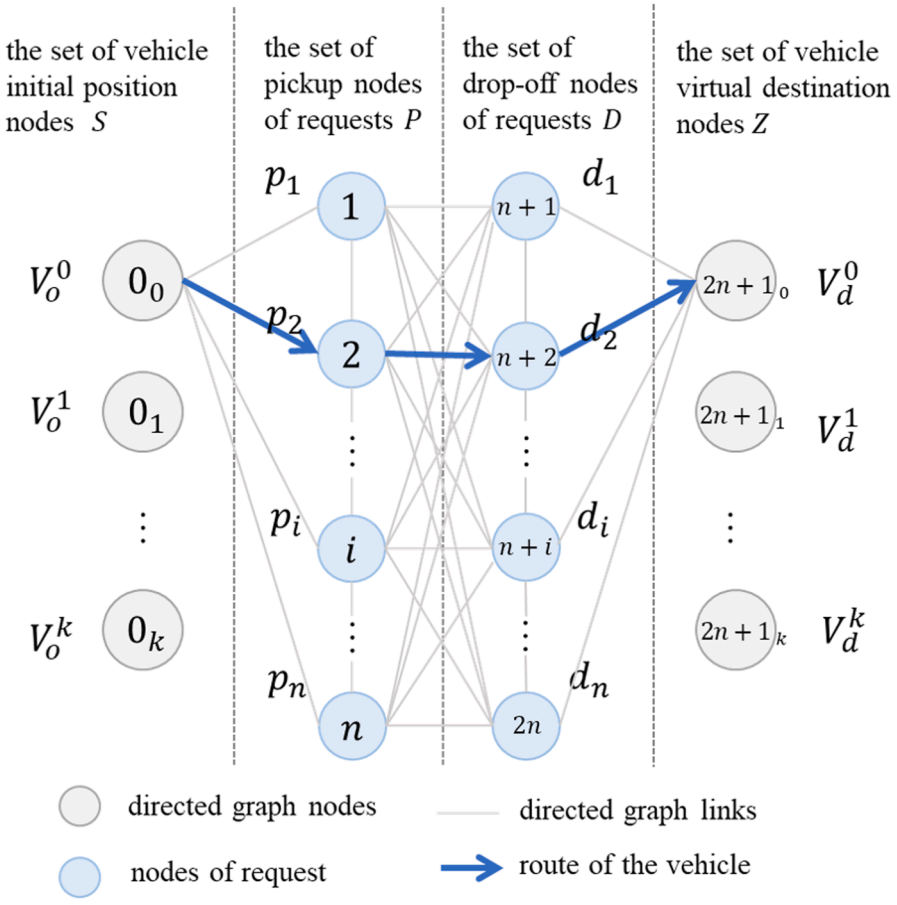

Fig. 4. Schematic of the relationship between actual stations and nodes within the model.

nected network. For a request, the pickup location p_r was encoded as node i, where $i = r$, and the drop-off location d_r was coded as node $i+n$ in the network. The initial location of a vehicle in this optimisation horizon was encoded as node 0_k, and its virtual destination location was encoded as node $2n + 1_k$. The vehicle route could be captured as a vehicle visiting a sequence of nodes consisting of the vehicle's initial position node, other nodes corresponding to the origins or destinations of the requests served by the vehicle, and the virtual destination node.

Let the node set be $N = P \cup D \cup S \cup Z$, where P is the set of pickup nodes of requests, D is the set of drop-off nodes, S is the set of vehicle initial position nodes, and Z is the set of vehicle virtual destination nodes. Therefore, there are $2|R| + 2|K|$ number of nodes in N, and $P_{waiting}$, $P_{scheduled}$, and $P_{predicted}$ are the pickup nodes for the waiting, scheduled, and future predicted requests, respectively, where $P = P_{waiting} \cup P_{scheduled} \cup P_{predicted}$. The link set between the pairs of nodes is $L = N \times N$. The directed graph $G = (N, L)$ consists of the node set N and arc set L.

Here, Fig. 4 illustrates the relationship between the vehicle initial position, the actual pickup/drop-off station of the passenger, and the nodes of the digraph in the model. The initial position of vehicle k was modelled as node 0_k, the pickup node p_i of request i was modelled as node i, the drop-off node d_i was modelled as node $n + i$, and the virtual destination node of vehicle k, V_d^k, was modelled as node $2n + 1_k$. Considering the blue arrow shown in the figure as an example, the vehicle numbered 0 starts from its current position V_o^0, goes to pickup node p_2 of the request numbered 2, then goes to drop-off node d_2 of passenger 2, and finally goes to virtual destination node V_d^0 of vehicle 0. In the model, these were expressed as the links between nodes 0_0 and 2, nodes 2 and $n + 2$, and nodes $n+2$ and $2n + 1_0$, respectively. The pickup and drop-off stations of each request, regardless of their actual station, were represented as independent nodes in the model. When multiple passengers got on and off at the same station at the same time, the actual stations were modelled as multiple nodes in the model, and the nodes in the model pointed to the same actual location.

The parameters used in the model are as follows. Parameter u_i^k was used to describe the matching relationship between vehicle k and node i, ensuring that the scheduled request was served by the predetermined vehicle. This was defined for scheduled requests $r \in R_{scheduled}$ only. If vehicle k passes through node i, $u_i^k = 1$, and 0 otherwise. Here, c^k denotes the unit travel cost, c^0 denotes the penalty

Table 1
Notation.

Sets and indices	
R	Set of requests, $r \in R$
K	Set of buses, $k \in K$
N	Set of nodes, $i \in N$
P	Set of pickup nodes, $i \in P$
D	Set of drop-off nodes, $i \in D$
S	Set of vehicle initial positions, $i \in S$
Z	Set of vehicle virtual destinations, $i \in Z$
L	Set of arcs, $(i, j) \in L$
G	Directed network
$R_{waiting}$	Set of waiting requests, $r \in R_{waiting}$
$R_{scheduled}$	Set of scheduled requests, $r \in R_{scheduled}$
$R_{predicted}$	Set of predicted requests, $r \in R_{predicted}$
$P_{waiting}$	Set of pickup nodes of waiting requests, $i \in P_{waiting}$
$P_{scheduled}$	Set of pickup nodes of scheduled requests, $i \in P_{scheduled}$
$P_{predicted}$	Set of pickup nodes of predicted requests, $i \in P_{predicted}$
Parameters	
q_r	Number of passengers in request r, $r \in R$
p_r	Pickup location of request r, $r \in R$
d_r	Drop-off location of request r, $r \in R$
q_{max}^k	Remaining seats in vehicle k, $k \in K$
V_o^k	Initial position of vehicle k, $k \in K$
V_d^k	Virtual destination of vehicle k, $k \in K$
u_i^k	Matching relationship between vehicle k and node i, $i \in P \cup D$, $k \in K$
$A = \{A_i^m\}$	Matching relationship between actual location m and node i, $i \in P \cup D$
c^k	Travel cost of vehicle k, $k \in K$, unit: yuan/minute
c^0	Penalty for rejecting a request
s_i	Service time of node i, $i \in P \cup D$
t_{ij}	Vehicle travel time between node i and node $ji.j \in N$
w_{max}	Maximum waiting time of a vehicle at a station
tep_i	Earliest pickup time of the passengers who board a bus at node i, $i \in P \cup D$
tlp_i	Latest pickup time of the passengers who board a bus at node i, $i \in P \cup D$
tld_i	Latest drop-off time of the passengers who get off a bus at node i, $i \in P \cup D$
a_{max}	Passenger's maximum detour time coefficient
Decision variables	
x_{ij}^k	Equal to 1 if arc (i, j) is traveled by vehicle k, $i.j \in N$
q_i^k	Number of passengers in vehicle k when the vehicle arrives at node i, $i.j \in N$, $k \in K$
w_i^k	Extra waiting time of vehicle k at node i, $i.j \in N$, $k \in K$
a_i^k	Time when vehicle k arrives at node $i.i.j \in N$, $k \in K$

value for rejecting the request, s_i denotes the service time of a pickup or drop-off node i, $i \in P \cup D$, t_{ij} denotes the vehicle travel time between the two nodes, and w_{max} denotes the maximum waiting time of the vehicle at a station. For pickup node $i \in P$, the time window was $[tep_i, tlp_i]$, where tep_i represents the passenger's earliest pickup time and tlp_i represents the passenger's latest pickup time. For $i \in D$, the time window was $[ted_i, tld_i]$, where ted_i represents the passenger's earliest drop-off time and tld_i represents the passenger's latest drop-off time. We set $ted_i = 0$ when there were no constraints on the earliest drop-off time, and $a_{max} \geq 1$ denotes the coefficient for passenger detour, which represents the passenger's tolerance for increased travel time compared with the shortest travel time.

Here, x_{ij}^k is a binary decision variable, $x_{ij}^k = 1$, if arc (i,j) is travelled by vehicle k, and 0 otherwise. q_i^k is a continuous variable that represents the number of passengers in vehicle k when it arrives at node i. Furthermore, w_i^k is a continuous variable representing the extra waiting time of vehicle k at node i, and a_i^k is a continuous variable representing the time when vehicle k arrives at node i. The notation used in this study is listed in Table 1.

2.2.2. Two-stage stochastic programming model

The introduction of future demand makes the flexible bus routing problem a decision-dependent stochastic optimisation problem that is challenging to solve. We propose a two-stage stochastic programming model to determine vehicle routes at the current time step for each horizon.

In the first stage of the model, vehicle routes are determined to serve the current request before the stochastic future request information is revealed. For the future demand, we assume that the prediction error obeys a normal distribution. A set Γ of scenarios is generated according to the prediction error. The probability of each scenario is ρ_γ, and the sum of all scenario probabilities equals to 1. Each scenario contains only one set of predicted requests. In the second-stage problem, once the stochastic requests are known, vehicles have to adjust the vehicle route to service those requests while maintaining the passenger–vehicle matching relationship obtained from the first-stage problem unchanged. Therefore, we calculate the recourse costs for the second-stage problem, including the operating costs of serving future requests and the penalty for rejecting them. For a given first-stage solution (i.e., vehicle–passenger matching and vehicle route), each future demand scenario corresponds to one recourse cost. The expected value of recourse cost is the objective function of the second-stage problem. Overall, the two-stage stochastic program aims to minimise the sum of the costs of serving the current requests obtained in stage-1 and the expected recourse costs obtained in stage-2.

With future demand described by a set of demand scenarios, the two-stage stochastic program can be transformed into a sample-average-based mixed-integer linear program (SA-MILP). SA-MILP is executed repeatedly for each horizon and periodically updates the vehicle routes.

Objective function (1) consists of three terms: the cost of vehicle travel time, the cost of rejected requests, and the expected recourse costs of stochastic requests in the second stage. b_1, b_2, and b_3 are all positive weight coefficients that define the relative importance of these components.

The first-stage problem schedules vehicles to minimise operating costs and serves as many requests as possible. The requests involved in the optimisation at this stage are $r \in R_{scheduled} \cup R_{waiting}$.

Stage-1

$$\min \sum_{k\in K}\sum_{i\in N}\sum_{j\in N} b_1 c^k t_{ij} x_{ij}^k + \sum_{i\in P} b_2 c^0 \left(1 - \sum_{k\in K}\sum_{j\in P\cup D} x_{ij}^k\right) + b_3 E[Q(x,w,a,q,\xi)] \tag{1}$$

$$\sum_{j\in N} x_{ij}^k = u_i^k, \ \forall i \in P_{scheduled}, k \in K \tag{2}$$

$$\sum_{j\in N} x_{ij}^k \leq 1, \ \forall i \in P_{waiting}, k \in K \tag{3}$$

$$\sum_{j\in N} x_{ij}^k - \sum_{j\in N} x_{n+i,j}^k = 0, \ \forall i \in P, k \in K \tag{4}$$

$$\sum_{j\in N} x_{ij}^k - \sum_{j\in N} x_{ji}^k = \begin{cases} 1 & i = 0 \\ 0 & i \in P \cup D \\ -1 & i = 2n+1 \end{cases} \tag{5}$$

$$q_i^k + q_i - M\left(1 - x_{ij}^k\right) \leq q_j^k, \ \forall i,j \in N, k \in K \tag{6}$$

$$q_i^k \leq q_{max}^k, \ \forall i \in N, k \in K \tag{7}$$

$$a_i^k + w_i^k + s_i + t_{ij} - M\left(1 - x_{ij}^k\right) \leq a_j^k, \ \forall i,j \in N, k \in K \tag{8}$$

$$\sum_{k\in K}\sum_{i\in P\cup D} A_i^m w_i^k \leq w_{max}, \ \forall i,j \in P \cup D, k \in K \tag{9}$$

$$tep_i \le a_i^k + w_i^k,\ \forall i \in P, k \in K \tag{10}$$

$$a_i^k \le tlp_i,\ \forall i \in P, k \in K \tag{11}$$

$$a_i^k \le tld_i,\ \forall i \in D, k \in K \tag{12}$$

$$a_{n+i}^k - a_i^k \le a_{max} t_{i,n+i},\ \forall i \in P, k \in K \tag{13}$$

$$a_i^k + w_i^k + s_i + t_{i,n+i} \le a_{n+i}^k,\ \forall i \in P, k \in K \tag{14}$$

$$\mu_i - \mu_j + M x_{ij}^k \le M - 1,\ \forall i, j \in N, i, j \ne 0, i \ne j, k \in K \tag{15}$$

$$x_{ij}^k = 0,\ if\ i = j \tag{16}$$

Stage-2

$$\min E[Q(x, w, a, q, \xi)] = \rho_\gamma \sum_\gamma \left(\begin{array}{c} \sum_{k\in K}\sum_{i\in N}\sum_{j\in N} b_1 c^k t_{ij} x_{ij}^k(\gamma) \\ + \sum_{i\in P} b_2 c^0 \left(1 - \sum_{k\in K}\sum_{j\in P\cup D} x_{ij}^k(\gamma)\right) \end{array} \right)$$

s.t. (4)–(16)

$$\sum_{j\in N} x_{ij}^k = u_i^k,\ \forall i \in P_{scheduled,\gamma}, k \in K \tag{18}$$

$$\sum_{j\in N} x_{ij}^k \le 1,\ \forall i \in P_{waiting} \cup P_{predicted,\gamma}, k \in K \tag{19}$$

Constraint (2) guarantees that scheduled requests must be served by the assigned vehicles from the last iteration. Constraint (3) ensures that each request is served exactly once per vehicle. Constraint (4) ensures that the same vehicle visits the pickup and drop-off nodes of the same request. Constraint (5) is a flow conservation constraint that ensures that the incoming flow at a node is equal to its outgoing flow. Constraint (6) calculates the vehicle load after visiting a node, M is a large number. Constraint (7) represents the vehicle capacity constraint. Constraint (8) calculates the arrival time of a vehicle after visiting a node. Constraint (9) ensures that the waiting time of the vehicles at each station does not exceed the maximum waiting time. Because the request-based coding method was adopted in this study, the actual pickup or drop-off positions of multiple requests may be at the same physical station. Thus, we used adjacency matrix $\boldsymbol{A}$ to calculate the vehicle waiting time at an actual station, where $\boldsymbol{A}$ is a matrix composed of the elements A_i^m, m is the actual station involved in this iteration, $i \in P \cup D$, and $A_i^m = 1$, if the physical station corresponding to node i is m, $A_i^m = 1$, and 0 otherwise. Constraints (10)–(12) define time windows. Constraints (10) and (11) restrict the vehicle arriving at the passenger pickup node to no earlier than the passenger's earliest pickup time and no later than the passenger's latest pickup time. Constraint (12) specifies a drop-off time window. Constraint (13) restricts the maximum detour time for each passenger: Constraint (14) causes the vehicle to visit the pickup node of the request first and then the drop-off node. Constraint (15) eliminates sub-loops, which avoids the occurrence of partial nodes in the vehicle route. Constraint (16) deletes the variables corresponding to the arcs that connect meaningless arcs.

In the second stage, the objective function $E[Q(x, w, a, q, \xi)]$ is used to minimise the expected recourse cost incurred by the predicted requests for the given current request-vehicle assignment results obtained from stage-1. In constraint (18), the scheduled requests include those scheduled in stage-1. In constraint (19), the requests include the waiting requests left from stage-1 and the predicted requests $r \in R_{predicted}$. Furthermore, ξ is a random parameter, and its realisation represents one future demand scenario γ.

3. Solution algorithm

The dynamic vehicle-routing problem is a typical NP-hard problem, specifically when considering stochastic future requests. The computational time increased exponentially with the number of requests and vehicles. Commercial solvers typically fail to handle large problems; thus, they cannot be directly applied to practical flexible bus systems. We propose a customised VSC-ALNS heuristic algorithm consisting of a vector-similarity (VS)-based clustering algorithm and an adaptive large-neighbourhood searching algorithm. The VS-based clustering algorithm classified requests and vehicles with similar route directions and distances into groups first to reduce the problem size. ALNS was then utilised to solve the vehicle-routing problem in each cluster. Parallel computing was utilised to handle future stochastic requests, which helped to further reduce the computation time by decomposing the large problem into a number of independent smaller problems.

3.1. Vector similarity-based clustering algorithm

3.1.1. Preparation

(1) Node vector.

We connected a pair of nodes i_1 and i_2 in the network defined in Section 2.2.1, and the directed line can be regarded as a vector λ, with the direction of i_1 pointing to i_2. There are two types of vectors. The request vector λ_r is the line directed from the pickup node to the drop-off node. For a vehicle with a scheduled route, the vehicle path vector λ_k is defined as the line directed from the vehicle's current position to the next planned arrival node. Here, Fig. 5 shows an illustration of the vectors.

(2) Spatiotemporal distance.

We calculated the spatiotemporal distance $\mathrm{SPTDist}(i_1, i_2)$ between two nodes using different vectors in the network. A node has two attributes: location and time. For request nodes, the location attribute is their geographic pickup or drop-off location. The earliest pickup time and latest drop-off time are the time attributes of the pickup and drop-off nodes, respectively. For vehicle-related nodes, the location attribute of the current position node was defined as its geographic location, and the time attribute was the current time.

For any two nodes i_1 and i_2 from different vectors in the network with their location attributes x_1 and x_2 and their time attributes t_1 and t_2, the spatiotemporal distance from i_1 to i_2 was calculated as follows:

$$\mathrm{SPTDist}(i_1, i_2) = \sqrt{\left(\frac{\mathrm{Dist}(x_1, x_2)}{w_d}\right)^2 + \left(\frac{t_1 - t_2}{w_t}\right)^2} \tag{20}$$

where $\mathrm{Dist}(x_1, x_2)$ is the geographic distance between two nodes, and w_d, w_t are normalisation factors that balance the importance of the spatial and temporal distances.

(3) Vector angles.

We defined the intersection angle of the two path vectors as the vector angle, which ranges from 0 to π. The cosine of the vector angle is equal to the ratio of the product of the vectors to the product of the two vector modules. The formula used is as follows:

$$\cos < \lambda_1, \lambda_2 > = \frac{\lambda_1 \lambda_2}{(|\lambda_1| \bullet |\lambda_2|)} \tag{21}$$

3.1.2. Vector similarity clustering rules

The clustering algorithm aimed to select request vectors with similar directions that were sufficiently close to each other. These requests could be served by the same vehicle and were clustered into a group. Then, we searched for vehicle vectors with similar path directions and placed them into the same group. Vehicle-request clustering decomposes a large-scale problem into several independent small-scale problems that are handled by parallel computing.

Here, Fig. 6 illustrates two types of vectors that can be grouped into clusters. In Fig. 6(a), a small vector angle indicates that the two vectors have similar direction. The spatiotemporal distance between the two starting nodes $\mathrm{SPTDist}(O_1, O_2)$ is short, and the ending nodes $\mathrm{SPTDist}(D_1, D_2)$ is also short. Therefore, the two vectors can be grouped together. In Fig. 6 (b), the ending node of vector λ_1 is close to the starting node of vector λ_2, and the two vectors can be connected in series and grouped into one cluster as well.

To identify the vectors that can be grouped, we propose three simple criteria based on vector similarity rules.

(1) Rule #1: Vector similarity in directions.

The cosine value of the intersection angle between the two vectors was calculated. If $\cos < \lambda_1, \lambda_2 >$ is greater than μ and less than 1, the two vectors are considered to have similar directions.

$$\mu \leq \cos < \lambda_1, \lambda_2 > \leq 1 \tag{22}$$

(2) Rule #2: Vector similarity for parallel trips.

If the spatiotemporal distance between the starting (ending) nodes of the two vectors is small, then the two vectors are similar in parallel. This can be determined using the following two constraints:

$$\mathrm{SPTDist}(O_1, O_2) \leq \theta_1 \tag{23}$$

$$\mathrm{SPTDist}(D_1, D_2) \leq \theta_2 \tag{24}$$

where θ_1 and θ_2 denote the maximum SPTDist differences. Constraint (23) limits the spatiotemporal distance between the starting

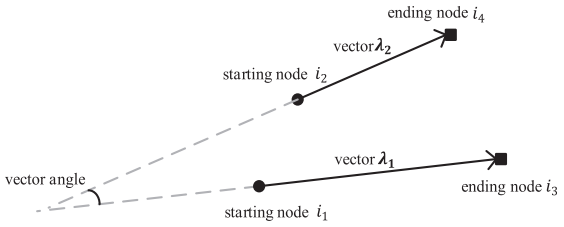

Fig. 5. Illustration of vectors.

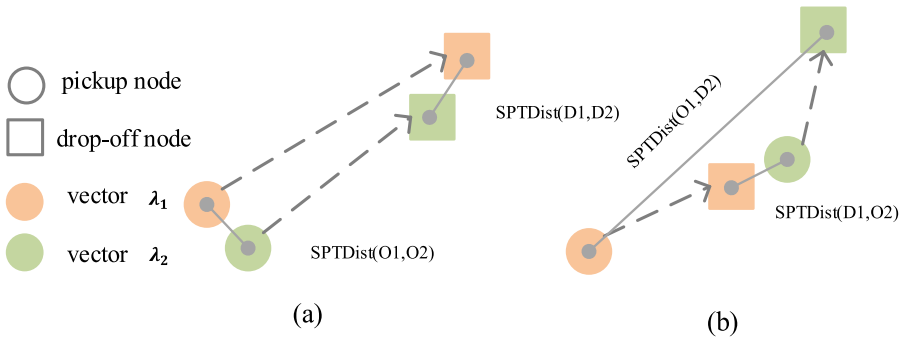

Fig. 6. Illustration of the relative location between two vectors.

nodes of the two vectors, and Constraint (24) limits the distance between the ending nodes of the two vectors.

(3) Rule #3: Vector similarity for series trips.

If the spatiotemporal distance between the ending node of one vector and the starting node of the other vector is small, the two vectors are similar in series. For example, we assume that λ_1 can be served by a vehicle first and λ_2 can be continuously served by the vehicle after λ_1. Therefore, the spatiotemporal distances $\mathrm{SPTDist}(O_2, D_1)$ between the ending node of λ_1 and the starting node of λ_2 must be close to each other. To prevent λ_2 from waiting too long, the sum of the service distances of λ_1 and λ_2 must not be too large.

Rule #3 can be expressed as follows:

$$min\left\{\mathrm{SPTDist}(O_1, D_2),\ \mathrm{SPTDist}(O_2, D_1)\right\} \le \gamma_1 \tag{25}$$

$$max\left\{\mathrm{SPTDist}(O_1, D_2),\ \mathrm{SPTDist}(O_2, D_1)\right\} \le \gamma_2 \tag{26}$$

Constraint (25) enables requests to be served individually, and γ_1 is the maximum allowable distance between the adjacent pickup node and the drop-off node. Constraint (26) restricts the total distance between the two requests such that the second request does not have to wait too long, and γ_2 is the maximum distance between the pickup and drop-off nodes that are not adjacent to each other.

3.1.3. Request clustering

The VS-based clustering algorithm was used to group waiting requests into a number of request clusters. The algorithm includes three parts: a direction similarity check, request vector clustering, and isolated request allocation.

Step 0: Initialization.

Let C_m be the set of requests that can be grouped in cluster m, and C_{temp}, the temporary set of requests that can be grouped in cluster m. The total number of clusters was determined using this procedure. Next, initialize $m = 1$.

Step 1: Direction similarity check.

Randomly select request r_0 from the pool of waiting requests. For any other requests $r_k \in R$ in the waiting pool, the vector cosine values $\cos < \lambda_{r_0}, \lambda_{r_k} >$ between requests r_0 and r_k. If r_k fulfils $\mu \le \cos < \lambda_1, \lambda_2 > \le 1$ (Rule #1), the request is put into set C_{temp}. This procedure is repeated for all $r_k \in R_{waiting}$. We can obtain a temporary request cluster C_{temp}.

Step 2: Request vector clustering.

Step 2.1 For the requests in C_{temp}, we randomly select a pair of requests (r_0, r_k) and calculate their spatiotemporal distances $\mathrm{SPTDist}(O_{r_0}, O_{r_k})$ and $\mathrm{SPTDist}(D_{r_0}, D_{r_k})$. If Rule #2 is satisfied, r_0 *and* r_k are moved from C_{temp} to the fixed cluster set C_m. Otherwise, calculate the spatiotemporal distance $\mathrm{SPTDist}(O_{r_0}, D_{r_k})$ and $\mathrm{SPTDist}(O_{r_k}, D_{r_0})$. If Rule #3 is satisfied, r_0 and r_k are moved from C_{temp} to cluster C_m. Repeat this step until all $r_k \in C_{temp}$ have been checked for r_0. Because clustering is conducted using request r_0 as a baseline, we considered r_0 as the central request $r_{m,center}$ in cluster C_m.

Step 2.2 If there are still requests in C_{temp}, the cluster number is increases by one, that is $m = m + 1$. We randomly select another request pair r_1 and r_k, check Rules #2 and #3 individually for each $r_k \in C_{temp}$. The request that fulfils Rule #2 or Rule #3 is clustered into C_m. Similarly, r_1 is considered the central request for this cluster. This process is repeated until there are no requests in C_{temp}.

Step 2.3 For the cluster set C_m obtained in this iteration, if there are two or more requests in C_m, all requests in C_m are removed from set R, and C_m is added to Θ. If there is only one request in C_m, set C_m is destroyed, and the single request in C_m is moved to isolated request set R_{iso}.

Step 3: Set $m = m + 1$. Repeat Steps 1 and 2 until set R is empty.

Step 4: Isolated request allocation.

Isolated requests, which do not belong to any existing cluster, may appear during the process. For each isolated request $r_k \in R_{iso}$, we calculate the spatiotemporal distance $\mathrm{SPTDist}(O_r, O_{r_{m,center}})$ between r_k's pickup node and the central request $r_{m,center}$'s pickup node in cluster C_m. Isolated request r_k is placed in the cluster with the smallest distance.

The pseudocode is shown in Fig. 7.

Algorithm 1. Request clustering

Input: (1) A set of requests $R = \{r_1, \ldots, r_n\}$ $r \in R_{waiting}$

Output: (1) A set of clusters $\Theta = \{C_1: r_1, \ldots, C_m: r_k\}$

Algorithm

$\Theta \leftarrow \emptyset$ $m = 1$

While R != $\emptyset$ do

    $C_{temp} \leftarrow \emptyset$, Randomly select $r_0 \leftarrow C_m$

    Calculate $\cos < \boldsymbol{\lambda}_{r_0}, \boldsymbol{\lambda}_{r_k} >$ for each $r_k \in R$

    Select r_k such that $\mu \leq \cos < \boldsymbol{\lambda_1}, \boldsymbol{\lambda_2} > \leq 1$ copy $r_k \leftarrow C_{temp}$ #Rule 1

    For each r_k in C_{temp} do

        Calculate $\mathrm{SPTDist}(O_{r_0}, O_{r_k})$ and $\mathrm{SPTDist}(D_{r_0}, D_{r_k})$

        If $\mathrm{SPTDist}(O_{r_0}, O_{r_k}) \leq \theta_1$ and $\mathrm{SPTDist}(D_{r_0}, D_{r_k}) \leq \theta_2$ #Rule 2

            $C_m = C_m + r_k$

        Else

            If max $\{\mathrm{SPTDist}(O_{r_0}, D_{r_k}), \mathrm{SPTDist}(O_{r_k}, D_{r_0})\} \leq \gamma_1$ and

                min $\{\mathrm{SPTDist}(O_{r_0}, D_{r_k}), \mathrm{SPTDist}(O_{r_k}, D_{r_0})\} \leq \gamma_2$ #Rule 3

            $C_m = C_m + r_k$

            Else

                Pass

            End if

        End if

    End for

    If len $(C_m) \geq 2$:

        $C_m \leftarrow \Theta$, Remove r from R for each $r \in C_m$, $m = m + 1$, $C_m \leftarrow \emptyset$

        $r_{m,center} = r_0$

    Else

        $r \leftarrow R_{iso}$, Remove r from R, $C_m \leftarrow \emptyset$

    End if

End while

For each request r in R_{iso} do #Isolate requests

    Calculate $\mathrm{SPTDist}(O_{r_{m,center}}, O_{r_k})$ for each $r_k \in \Theta$

    $C_m = C_m + r$ if min $\mathrm{SPTDist}(O_{r_{m,center}}, O_{r_k})$ $r_k \in C_m$

End for

Fig. 7. Pseudocode of the request-clustering algorithm.

3.1.4. Vehicle allocation

To date, multiple request clusters $\Theta = \{C_m\}$ and the central request of each cluster $R_c = \{r_{m,center}\}$ have been obtained. Next, we must allocate suitable vehicles to different clusters to serve the passengers. The vehicle allocation algorithm has three major parts: direction similarity check, vehicle assignment, and vehicle redistribution.

The input of the vehicle allocation algorithm include the request cluster Θ, set of central requests R_c, set of vehicle path vectors K_1,

Algorithm 2. Vehicle allocation

Input: (1) A set of clusters $\Theta = \{C_1: r_{1,center}, \dots, C_m: r_{m,center}\}$

(2) A set of scheduled vehicles $K_1 = \{k_1: \lambda_1, \dots, k_v: \lambda_v\}$

(3) A set of free vehicles $K_2 = \{k_{v+1}: \lambda_{v+1}, \dots, k_n: \lambda_n\}$

Output: (1) The matching relationship between vehicles and clusters

$$\Omega = \{C_1: [k_1, \dots], , \dots, C_m: [k_n, ..,]\}$$

Algorithm

01:$\Omega = \{C_1: [], \dots, C_m: []\}$ $R_C = \{r_{1,center}, \dots, r_{m,center}\}$

02: While K_1 != $\emptyset$ do

03: Calculate $\cos < \boldsymbol{\lambda}_{r_{m,center}}, \boldsymbol{\lambda}_k >$ for each $r_{m,center} \in R_C$ #Rule 1

04: Sort Θ according to the $\cos < \boldsymbol{\lambda}_{r_{m,center}}, \boldsymbol{\lambda}_k >$

05: For each cluster C_m in Θ do

06: If $q_c > \Sigma q_k * \alpha$:

07: Calculate $\text{SPTDist}(O_k, O_{r_{m,center}})$ and $\text{SPTDist}(D_k, D_{r_{m,center}})$

08: If $\text{SPTDist}(O_k, O_{r_{m,center}}) \le \theta_3$ and $\text{SPTDist}(D_k, D_{r_{m,center}}) \le \theta_4$

09: $C_m: [, ..,] + k$ #Rule 2

10: Remove k from K_1

11: Break

12: End if

13: Else

14: Calculate $\text{SPTDist}(O_{r_{m,center}}, D_k)$ and $\text{SPTDist}(O_k, D_{r_{m,center}})$

15: If max $\{\text{SPTDist}(O_{r_{m,center}}, D_k), \text{SPTDist}(O_k, D_{r_{m,center}})\} \le \gamma_3$ and

16: min $\{\text{SPTDist}(O_{r_{m,center}}, D_k), \text{SPTDist}(O_k, D_{r_{m,center}}) \le \gamma_4$ #Rule 3

17: $C_m: [, ..,] + k$

18: Remove k from K_1

19: Break

20: Else

21: Pass

22: End if

23: Else

24: Pass #next cluster

25: End if

26: End for

27: if k is not assigned to any cluster

28: Remove k from K_1 and put k into K_2

27: End while

28: For each vehicle k in K_2 do # isolated vehicles

29: Calculate $\text{SPTDist}(O_{r_{m,center}}, O_k)$ for each $r_{m,center} \in R_C$

30: $C: [k_n, ..,] + k$ if min $\text{SPTDist}(O_{r_{m,center}}, O_k)$ $r_{m,center} \in R_C$

31: Remove k from K_2

32: End for

Fig. 8. Vehicle-allocation algorithm pseudocode.

and set of idle vehicles K_2. The output is the matching relationship between vehicles and request clusters. Here, Ω was initialised as a hash table that was used to record the matching relationship between the vehicles and the demand clusters. Multiple vehicles could be assigned to a cluster.

Step 1: Direction similarity check.

Randomly select vehicle $k \in K_1$. For a central request $r_{m,center} \in R_c$, we calculate the vector cosine values $\cos < \lambda_{r_{m,center}}, \lambda_k >$ between request $r_{m,center}$ and vehicle k. Repeat this procedure for all $r_{m,center} \in R_c$. The clusters in Θ are sorted in ascending order according to the angle cosine value. (Rule #1).

Step 2: Vehicle assignment.

Step 2.1. For vehicle k, from the top of the ordered list of clusters obtained from step 1, check whether the number of seats provided by the allocated vehicle $\Sigma q_k{}^*\alpha$ satisfies the demand of cluster q_c. If yes, the vehicle assignment for this cluster is completed, and this cluster is removed from the cluster set Θ. The next cluster from the ordered list is selected and Step 2.1. If no, proceed to Step 2.2.

Algorithm 3. Outline of ALNS

Input: (1) s_{init}

(2) Parameters

Output: (1) s_{best}

Algorithm

01: $iteration\ Time\ = 0; not\ Improved\ Time = 0; s_{best} \leftarrow s_{curr}$

02: $s_{curr} \leftarrow s_{init}$ #Initialization

03: Repeat

04: Destroy operator s_{curr} by roulette-wheel selection

05: Repair operator s_{curr} by roulette-wheel selection

06: Obtain s_{temp} and update the number of using operator

07: If $f(s_{temp}) > f(s_{best})$

08: $s_{best} \leftarrow s_{curr} \leftarrow s_{temp}\ not\ Improved\ Time = 0$

09: Update operator scores by ω_1

10: Else if $f(s_{temp}) > f(s_{curr})$

11: $s_{curr} \leftarrow s_{temp}\ not\ Improved\ Time + 1$

12: Update operator scores by ω_2

13: Else if $f(s_{temp}) \leq f(s_{curr})$ but accepted by the simulated annealing method

14: $s_{curr} \leftarrow s_{temp}\ not\ Improved\ Time + 1$

15: Update operator scores by ω_3

16: Else

17: Reject solution and update operator score by ω_4

18: End if

19: If $(iteration\ Time\ + 1)\%100 = 0$

20: Update the weights and scores of all operators

21: End if

22: $iteration\ Time\ + 1$

23: Until $iteration\ Time\ > n$

24: Return s_{best}

Fig. 9. Outline of ALNS.

Step 2.2. Calculate the distances $\mathrm{SPTDist}(O_k, O_{r_{m,center}})$ and $\mathrm{SPTDist}(D_k, D_{r_{m,center}})$ between the central request $r_{m,center}$ of the selected cluster and vehicle k. If Rule #2 is satisfied, vehicle k is placed in the vehicle list corresponding to cluster C_m and k is removed from set K_1. Terminate and proceed to step 2.4. Otherwise, calculate the spatiotemporal distance $\mathrm{SPTDist}(O_{r_{m,center}}, D_k)$ and $\mathrm{SPTDist}(O_k, D_{r_{m,center}})$. If Rule #3 is satisfied, vehicle k is placed in the vehicle list corresponding to cluster C_m, and k is removed from set K_1. Terminate and proceed to Step 2.4. If neither of the two rules are fulfilled, repeat Steps 2.1 and 2.2 until k is assigned or all clusters in the ordered list have been checked.

Step 2.3 If vehicle k is not assigned to any clusters after all the clusters have been checked, the vehicle is placed into K_2, waiting for redistribution.

Step 2.4 Select another vehicle k in K_1 and return to Step 1 until all the vehicles in K_1 have been checked.

Step 3: Vehicle redistribution.

The unallocated and idle vehicles in K_2 were redistributed. For each vehicle $k \in K_2$, we calculated the spatiotemporal distance $\mathrm{SPTDist}(O_{r_{m,center}}, O_k)$ between the current position of vehicle k and each cluster's central request $r_{m,center}$'s pickup node in cluster C_m. Vehicle k was assigned to the cluster with the smallest spatiotemporal distance.

The details of the algorithm are shown in Fig. 8.

3.2. Adaptive large neighbourhood search algorithm

We obtained vehicle-request clusters using the VS-based clustering algorithm. Next, for each cluster, we used ALNS to determine whether to serve the requests and vehicle routes with the aim of generalised cost minimisation. An outline of the ALNS is shown in Fig. 9. First, we obtained an initial solution using the nearest vehicle assignment. In each iteration, the ALNS destroyed the current solution using a destroy operator and then constructed a new solution using a repair operator. The probability of selecting the operators depended on their historical performance. A new solution was accepted as the next current solution if it satisfied the specified acceptance criteria.

3.2.1. Initial solution construction

The nearest-vehicle dispatching heuristic was used to construct the initial solution. In a vehicle-request cluster, we calculated the geographic distances between the request pickup node and the available vehicle's position and sorted the request-vehicle pairs in the ascending order of distance. From the top of the list, the pickup node of the request was inserted at the first feasible insertion position on the vehicle's existing path. The drop-off node was inserted into the vehicle path after the pickup node. If there was no feasible point for the request to be added to the vehicle's path, we moved down to the next request-vehicle pair with the second shortest distance and attempted to insert the request to that vehicle. This process was repeated until all requests were checked. A request was rejected if no vehicle could serve it. The initial solution contained multiple vehicle routes and a list of unserved passenger requests.

3.2.2. Destroy and repair operators

The next step was to destroy the initial solution using the destroy operators and repair the solution using repair operators to obtain a better solution. We used three types of destruction operators.

(1) Random removal: This operator randomly removes several requests from the current vehicle route. Although this method can accidentally destroy a suitable solution, it helps avoid trapping in the local optima.

(2) Worst removal: This operator removes the request that leads to the largest cost savings for vehicle-route operation. A request is removed by deleting its pickup and drop-off nodes from the vehicle route and directly reconnecting the affected adjacent nodes. The removed request is placed back to the request waiting pool and inserted into other positions to be assigned to a different vehicle or to be rejected.

(3) Shaw removal: We calculated the similarity of every-two requests in a vehicle's route and identified the two requests with the highest similarity in a route. We randomly removed one request from the two requests in the vehicle route to increase the diversity of the solutions. The similarity was calculated based on the differences between the two requests in terms of spatiotemporal distance and passenger demand. The similarity measure $\Phi(i,j)$ of two requests i,j is calculated using Equation (27).

$$\begin{aligned}\Phi(i,j) = \tau_1\left(d_{P(i),P(j)} + d_{D(i),D(j)}\right) + \tau_2\left(\left|T_{P(i)} - T_{P(j)}\right|\right. \\ \left. + \left|T_{D(i)} - T_{D(j)}\right|\right) + \tau_3\left|D_i - D_j\right|,\end{aligned} \tag{27}$$

where $d_{P(i),P(j)}$ and $d_{D(i),D(j)}$ are the differences in distance between the pickup and drop-off nodes of the two requests; $T_{P(i)} - T_{P(j)}$ and $T_{D(i)} - T_{D(j)}$ are the differences between the pickup and drop-off times, respectively; $D_i - D_j$ is the difference in the number of seats booked in the request; and $\tau_1, \tau_2,$ and τ_3 are weights. Note that a small value of Φ indicates that the requests are similar.

After the destroy operation, we obtained the shortened vehicle routes, removed requests, and unserved requests. We used the repair operator to insert the unserved requests and removed requests back to the vehicle routes. Three repair operators were used in the study.

(1) Random insertion: This operator randomly selects a request in the waiting pool and a vehicle and searches for a feasible position to insert the request to the vehicle route. If there is no feasible position, another random vehicle is selected and checked. If no vehicle has a feasible insertion position, then the request is rejected and remains in the waiting pool.

(2) Greedy insertion: The increase in the route operating cost (the first term in the objective function) was calculated after inserting the request. The greedy insertion operator inserts the request to the route that could lead to a minimum operating cost increase of the route.

(3) Regret-2 insertion: The regret heuristic considers both the first-best insertion point and the second-best position. The regret-2 insertion operator calculates the added costs by inserting the request to the first-best vehicle route (the one found in the greedy algorithm) and the second-best vehicle route. The regret-2 value represents the difference in operating costs between the two routes. The request with the largest regret value is then inserted into the first-best position.

3.2.3. Solution acceptance principle and operator weight update

To avoid the algorithm from being trapped in a local optimum, simulated annealing was embedded in the ALNS framework. If the new solution was better than the current solution, it was accepted as the new current solution. Otherwise, a non-improved solution was accepted with a probability of

$$p(s_{new}, s_{curr}) = min\left\{e^{(s_{new}-s_{curr})/T_k}, 1\right\}, \tag{28}$$

where T_t is the temperature in the current iteration t, $T_t = T_0 c^t$, where T_0 is the initial temperature, and c^t is the cooling rate in the iteration t.

To select the destroy and repair operators more effectively, as in Ropke and Pisinger (2006), the destroy and repair operators were selected through a roulette process. Initially, the operator weights were all equal to 1. In iteration $t+1$, the probability of operator d being selected is

$$P_d^{t+1} = P_d^t(1 - r_p) + r_p \pi_d / \sigma_d,$$

where P_d^t is the probability of operator d being selected in iteration t, r_p is the roulette parameter, π_d is the score of operator d, and σ_d is the number of times operator d is selected. An adaptive method was adopted to update the score of the selected operator. The operator score d was updated according to the historical performance of the operator. If the temporary solution was better than the global optimal solution, the destroy and repair operators were assigned a score ω_1. If the temporary solution was worse than the global optimal solution but better than the current solution, the destroy and repair operators were assigned a score ω_2. If the temporary solution was worse than the current solution but was still accepted, the destroy and repair operators were assigned a score ω_3. If the temporary solution was not accepted, the destroy and repair operators were assigned a score ω_4. Operators that obtained better solutions obtained higher scores and were more likely to be selected for the next iteration by the roulette.

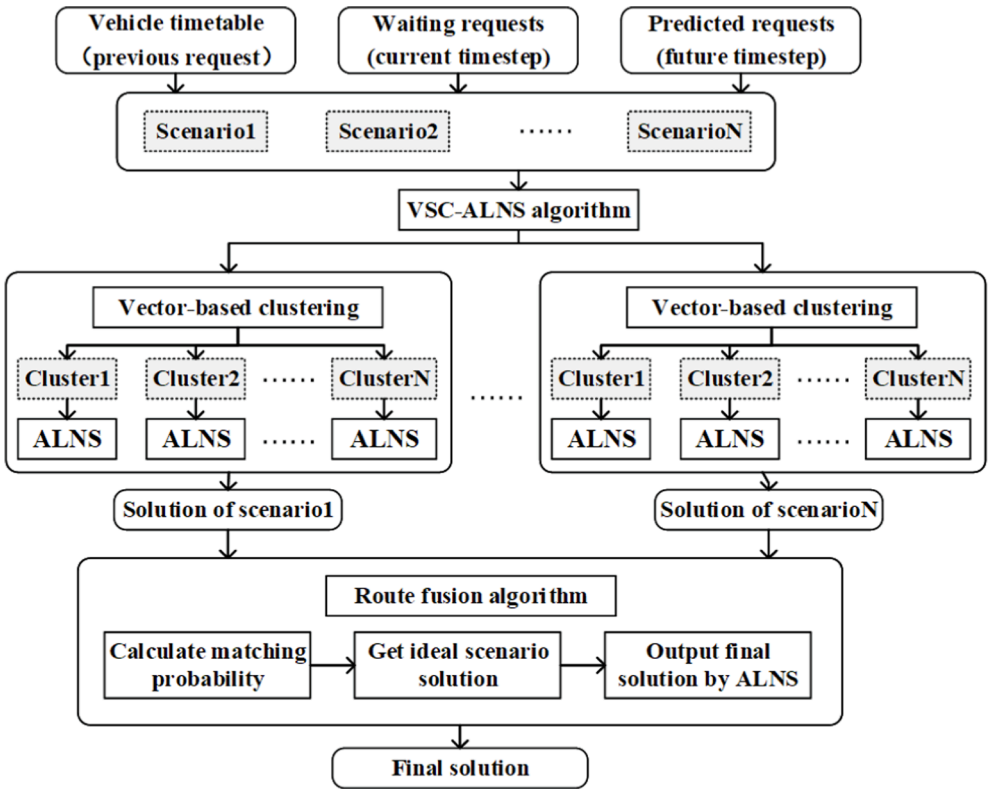

Fig. 10. Parallel computing framework.

3.3. Parallel computing framework

Parallel computing is used to solve large-scale problems that can be decomposed into a number of independent small-scale problems. Multiple small-scale problems can be solved in parallel using multiple threads, which can greatly expedite the solution procedure. We used parallel computing at two locations in our model. First, future uncertain requests were represented by probabilistic demand scenarios. One demand scenario included known current requests and predicted future requests. The current requests were the same for different scenarios. The predicted future requests varied among scenarios. Deterministic demand scenarios were handled independently using parallel computing. Second, in one demand scenario, the clustering algorithm was used to decompose the model into a number of independent vehicle-request clusters that could be handled by parallel computing. The parallel-computing framework is shown in Fig. 10.

The solution method consists of three parts: demand-scenario generation, scenario-based route calculation, and executable route generation. In the demand-scenario generation, historical demand data were used to predict future demand. In the scenario-based route calculation, actual vehicle states and known requests at the current time step and one deterministic future demand scenario were used as inputs. For one demand scenario, the VS-based clustering algorithm was used to generate multiple request-vehicle clusters. Then, the ALNS algorithm was used for each cluster to obtain the passenger–vehicle assignment and tentative vehicle routes. Different scenarios were computed in parallel. Note that we intentionally separated the demand scenarios and ignored the

Algorithm 4. Route-fusion algorithm

Input: (1) Scenario solution $S = \{s_1, \dots, s_m$

(2) Scenario and its probability $\Gamma = \{\gamma_1 : \rho_1, \dots, \gamma_m : \rho_m\}$

(3) Request set $R = \{r_1, \dots, r_n\}$

Output: (1) s_{final}

Algorithm

01: $W_1 \leftarrow \emptyset,\ W_2 \leftarrow \emptyset,\ s^{ideal} \leftarrow \emptyset,\ M \leftarrow \{ms^\gamma = 0, \dots\}$

02: For γ in S do

03: If request r is served by vehicle k # k is the vehicle number

04: $\omega^{\gamma}_{(r,k)} = \rho_\gamma,\ W_1 \leftarrow \omega^{\gamma}_{(r,k)}$

05: Else if r is not served

06: $\omega^{\gamma}_{(r,k)} = 0,\ W_1 \leftarrow \omega^{\gamma}_{(r,k)}$

07: End if

08: $\omega_{(r,k)} = \sum_{\gamma \in \Gamma} \omega^{\gamma}_{(r,k)}\ \forall\, r \in R,\ if\ \omega^{\gamma}_{(r,k)}\ in\ W_1$

09: $W_2 \leftarrow \omega_{(r,k)}$

10: For r in R do

11: $\omega^{ideal}_{(r,k)} = 1,\ k = k \leftarrow max\ \omega_{(r,k)}$

12: $s^{ideal} \leftarrow \omega^{ideal}_{(r,k)}$

13: End for

14: For s in S do

15: For r in R do

16: If $\omega^{\gamma}_{(r,k)} = \omega^{ideal}_{(r,k)} \times \rho_\gamma$

17: $ms^\gamma \mathrel{+}= 1$

18: obtain s_{final} by ALNS according to $max\ ms^\gamma$

Fig. 11. Route-fusion algorithm pseudocode.

correlation between them in this step. Hence, the generated vehicle routes may be infeasible and cannot be directly applied in practice. In the last step, route fusion was used to determine the executable passenger–vehicle matching and vehicle routes. We proposed a route fusion heuristic to identify the most cost-efficient solution. The executive vehicle routes only serviced the current known requests, and the predicted requests, which did not occur yet, were removed. The algorithm concept was to calculate the matching probability $\omega^{\gamma}_{(r,k)}$ in each scenario and obtain a set of solutions with the highest probability as the ideal matching relation. According to the matching relationship, the ALNS was used to obtain the executable vehicle routes of the final output.

The route-fusion algorithm is shown in Fig. 11. The inputs were the scenario-based solutions generated by the VSC-ALNS method, including the request-vehicle assignment, vehicle route, and probability of each scenario. The output was the actual vehicle route to be executed to service existing requests. The details are as follows:

Step 1: Calculate the matching probability $\omega^{\gamma}_{(r,k)}$ for the request $r \in R$ and vehicle $k \in K$ in scenario γ. If request r is served by vehicle k in scenario γ, then the matching probability $\omega^{\gamma}_{(r,k)} = \rho_{\gamma}$, where ρ_{γ} is the occurrence probability of scenario γ. For rejected requests or unmatched request–vehicle pairs, $\omega^{\gamma}_{(r,k)} = 0$. The individual matching probabilities $\left\{\omega^{\gamma}_{(r,k)}\right\}$ are stored in set W_1.

Step 2: For the non-zero vehicle-request pairs in W_1, calculate the sum of the matching probabilities in all scenarios $\omega_{(r,k)} = \sum_{\gamma}\omega^{\gamma}_{(r,k)}$. The cumulative matching probability $\{\omega_{(r,k)}\}$ is stored in set W_2. We sort the $\omega_{(r,k)}$ in descending order. From the top of the sorted $\omega^{*}_{(r,k)}$, request r is assigned to vehicle k with the maximum matching probability. If request r has already been assigned, we move down to the next pair in $\omega^{*}_{(r,k)}$. If the remaining seats of vehicle k are exhausted, we move down to the next one in the sorted list until all requests have been matched, or until the number of remaining seats is zero. Then, we obtain the ideal solution-matching relationship $\omega^{ideal}_{(r,k)}$.

Step 3: According to the ideal matching relationship, the ALNS is used to recalculate the vehicle routes and passenger service time.

4. Case study

To demonstrate the applicability of the proposed models and algorithms, they were applied to a series of practical networks in Shanghai, China. We also compared the performance of the proposed flexible bus service with that of the conventional bus service, taxies, and ride-sharing services. The proposed solution algorithms were programmed in Python 3.7 and run on a server with 12-thread 1.60 GHz CPUs and 4 GB RAM.

4.1. Experimental setup

4.1.1. Data description

We used real taxi trip data from Shanghai, China in the experiments. The trip data covered Xuhui and Jiading Districts (see Fig. 12) for a typical day on 30 April 2021. The data contained the order ID, order start and end times, and longitudes and latitudes of the trip origins and destinations. The study area in Xuhui District is the CBD of Shanghai, which has a high travel demand, whereas the study area in Jiading District is an industrial area, which has a relatively low travel demand. The two areas were divided into zones of 1 km^2, and the orders were aggregated in the zonal centroid. We chose two typical study horizons for comparative analysis: the morning peak during 8:00–9:00 and the other is the mid-night during 0:00–1:00. Four cases were generated: peak hours in Xuhui, off-peak hours in Xuhui, peak hours in Jiading, and off-peak hours in Jiading. Table 2 lists the number of requests and bus fleet sizes used in the four cases.

The buses were uniformly distributed in the study area at the beginning of operation, and we employed homogeneous minibuses with 10 seats in the experiment. The bus fleet size was set as the number of requests divided by vehicle capacity.

Fig. 12. Schematics of the study areas: (a) Xuhui District; (b) Jiading District.

Table 2
Case properties.

Case	Region name	Time	Number of requests	Bus fleet size
1	Jiading	00:00–01:00	47	30
2	Jiading	08:00–09:00	488	50
3	Xuhui	00:00–01:00	1396	150
4	Xuhui	08:00–09:00	3639	360

4.1.2. Parameter setting

The parameters in the model were set as follows: The unit operating cost c^k was 1 yuan/min, and the penalty for rejecting request c^0 was 30 yuan. The maximum detour coefficient a_{max} was 3. The maximum waiting time of the vehicle at the station was $w_{max} =$ 5 min. The service time s_i of a passenger (passengers getting on and off a bus) was 0.2 min. The time interval of one optimisation horizon T was 5 min. The flexible bus capacity q was 10. The maximum waiting time after a passenger submits a request was set to 15 min. The weights of the vehicle operation cost and request rejection penalty were set to 1; i.e., $b_1 = b_2 = b_3 = 1$. The time window of a request was 15 min, and the earliest departure time was the passenger order submission time.

The parameters used in the solution algorithm were set as follows: In the VS-based clustering algorithm, the spatial weight w_d was 1, the temporal weight w_t was 3, and the maximum cosine value of the angle between the two vectors was $\mu = 0.87$. The clustering rule-related parameters were $\theta_1 = \theta_2 = \gamma_1 = 500$ and $\gamma_2 = 4000$. The parameters in the ALNS algorithm are as follows: the initial temperature T_0 was 100, the cooling rate c^t was 0.7, the end temperature T_{end} was 10, the maximum number of iterations L was 100, and the operator scores were $\omega_1 = 20, \omega_2 = 12, \omega_3 = 6,$ and $\omega_4 = 2$.

4.2. Result analysis of the four cases

The four cases represent different travel-intensity levels. A higher demand and larger fleet size make the problem more challenging to solve. Six performance measures were adopted: generalised costs (the objective value), request rejection rate, total vehicle travel time, average passenger-detour-time, average passenger-waiting-time, and computational time. The passenger detour time was calculated as the difference between the actual travel time of the on-board passengers and the shortest path travel time. Passenger waiting time refers to the difference between the expected boarding time and the actual boarding time.

We also solved the dynamic vehicle-routing problem using the commercial solvers Gurobi and ALNS. The results are presented in Table 3. The VSC-ALNS algorithm outperformed the other two algorithms in terms of computational time. For Case 1, all three algorithms obtained the optimal solution. With an increase in the problem size, Gurobi was unable to obtain a feasible solution for Cases 2, 3, and 4 in 13 h. The computation times of the ALNS algorithm were 271 s and 2043 s, which were 127.5 % and 287.7 % higher those of the VSC-ALNS algorithm, respectively. Only the VSC-ALNS successfully solved Case 4.

In Cases 1 and 2 with less than 500 requests per hour, the average computation time for one horizon by VSC-ALNS is 10 s and 120 s respectively, which is smaller than the duration of the optimization horizon, 5 min. The variance of calculating time for each horizon in Case 1 is 5.2 s, and the variance of calculating time for each horizon in Case 2 is 10.4 s. It fulfills the real-time application requirement. In Cases 3 and 4, the average computation time of one horizon by VSC-ALNS is 8.8 min and 46.6 min respectively. Since we used only one server Dell PowerEdge T430 with 12 threads in the experiment, the algorithm could not reach its full potential. We expect the computation time can be reduced to less than 5 min if high performance server is used. At present, the test score of the latest CPU AMD

Table 3
Performance comparison of different demands.

Case		Case 1	Case 2	Case 3	Case 4
Objective Function Value (yuan)	Gurobi	419	—	—	—
	ALNS	419	7406	19,344	—
	VSC-ALNS	419	5614	15,043	44,223
Rejection Rate (%)	Gurobi	2.1 %	—	—	—
	ALNS	2.1 %	28.5 %	25.2 %	—
	VSC-ALNS	2.1 %	18.4 %	17.4 %	20.5 %
Total Vehicle Travel Time (minute)	Gurobi	389	—	—	—
	ALNS	389	3235.7	8814.4	—
	VSC-ALNS	389	2914.6	7753.0	21843.3
Average Passenger-Detour-Time (minute)	Gurobi	—	—	—	—
	ALNS	1.7	2.0	3.5	—
	VSC-ALNS	1.7	1.4	3.8	3.6
Average Passenger-Waiting-Time (minute)	Gurobi	6.5	—	—	—
	ALNS	6.5	9.6	9.0	—
	VSC-ALNS	6.5	8.8	8.4	8.4
Average Computational Time for one horizon (s)	Gurobi	104	—	—	—
	ALNS	25	271	2043	—
	VSC-ALNS	10	120	527	2794

Remark: The cases where a feasible solution could not be obtained within 50,000 s is marked as '—'.

EPYC 7773X is 89614, and the test score of the experimental server CPU Intel Xeon E5-2603 v4 in this research is 4695, 18 times higher. We conservatively estimate that using the latest CPU can improve the computing efficiency by 10 times. Meanwhile, for computationally intensive tasks, it is estimated that C++ can improve computing speed by 50–200 times compared to python, and Java can improve computing speed by 60–100 times compared to python. Therefore, we believe that the real-time application of the algorithm is feasible if we further increase the computing power.

Regarding the solution quality, the VSC-ALNS algorithm obtained equal or lower generalised costs (the objective values) in all four cases compared to the other two algorithms. In Case 1, the VSC-ALNS obtained the same global optimal solution as the exact solver Gurobi. In Cases 2 and 3, the generalised costs obtained by VSC-ALNS were 32.2 % and 28.6 % lower than those obtained by ALNS, respectively.

Subsequently, we analysed only the solutions obtained by the VSC-ALNS in the four cases. The request rejection rate in Case 1 was as low as 2 % when there are sufficient vehicles. However, the request rejection rate increased rapidly to approximately 20 % for Cases 2, 3, and 4. The average passenger-detour-time ranged from 1.7 to 3.8 min, and the average passenger-waiting-time ranged from 6.5 to 8.8 min. The passenger waiting time included both the system response time from the request submission to the vehicle pick-up confirmation and the waiting time from pickup confirmation to the actual passenger boarding time. If we change the operating mode of the flexible bus from an instant response to requiring reservation at least 5 min in advance, the waiting time can be further reduced.

4.3. Comparison with different shared mobility modes

In this section, we compare the performance of the flexible bus with three transport service modes: conventional buses, taxis, and ride-sharing vehicles. These were tested under the same demand scenarios. The vehicle fleet had the same number of seats in the different modes.

For conventional buses, we set up a grid bus network with five horizontal bus lines and six vertical bus lines (see Fig. 13). The bus frequency was set to 8 min. For the ride-hailing/taxi mode, the taxis were initially randomly distributed in the area, and the taxis were assumed to be employed under the online ride-hailing service via a third-party platform. Passengers send requests through mobile phones, and the platform assigns a nearby taxi to service these requests. A taxi can serve only one order at a time. For the ridesharing mode, we used the well-known sharable network algorithm proposed by Alonso-Mora et al. (2017) published on PNAS (referred to as NY-2017 hereinafter).

We analysed the differences between the four travel modes for four cases with different demand intensities. The results are presented in Table 4. Under Case 1, with extremely low demand, the taxis outperformed the other four modes. Taxis with the smallest vehicle capacity could fully cover travel demand with a higher efficiency than shared mobility for extremely low demand at midnight. In Case 2, with approximately 500 requests, the flexible bus performed better. The request rejection rate was only 18 % for the flexible bus, i.e., three times lower than that for conventional bus. Moreover, the average passenger-waiting-time of the flexible bus was 1.4 min, which is approximately-six times lower than that of NY-2017 and the conventional bus. In Case 3, with 1300 requests, the flexible bus performed better. The request rejection rate was only 17 %, whereas the other modes suffered at least 48 % rejection rate. Simultaneously, the total vehicle travel time was ten times lower than that of taxis and 5 % lower than that of NY-2017. This demonstrates that the flexible bus can handle medium to large demand intensities. It is almost as cost-efficient as conventional buses, and provides a high service level to passengers as taxis. In Case 4 with 3600 requests, the conventional bus obtained the lowest total vehicle travel time of 5280 min using only 66 50-seat buses, saving approximately 60 % of the operational costs compared to the other modes. However, the passenger rejection rate was 28 %, which is 7 % higher than that of the flexible bus. The detour time was 10 min, which is lower than that of NY-2017 but 7 min higher than that of the flexible bus. The results demonstrate the economics of the scale for ground public-transport systems. A conventional bus is the most cost-efficient way to satisfy high and stable travel demands. However, passengers still have to compromise for some detours.

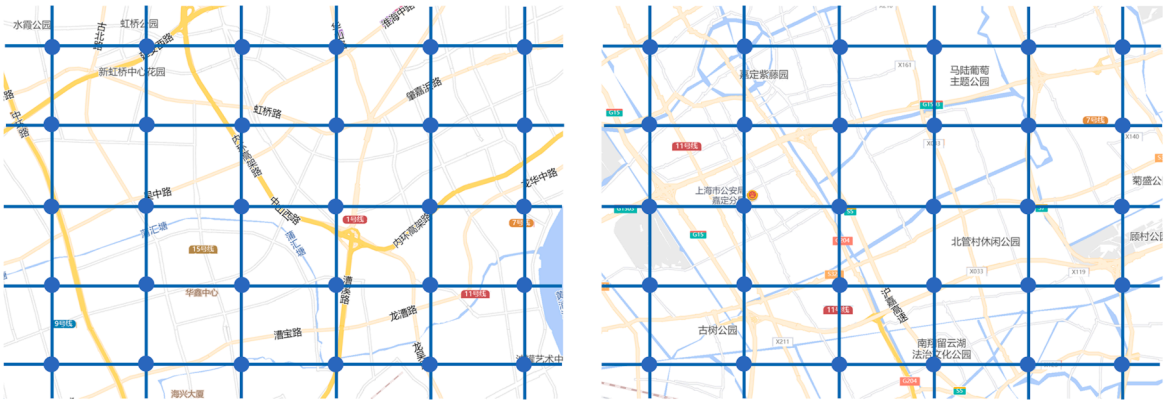

Fig. 13. Schematics of the conventional bus lines in Xuhui and Jiading Districts.

Table 4
Performance comparison of different travel modes.

Case		Case 1	Case 2	Case 3	Case 4
Total number of seats (the number of vehicle*vehicle capacity)	Conventional bus	44*50	44*50	44*50	66*50
	Taxi	75*4	125*4	375*4	900*4
	NY-2017	30*10	50*10	150*10	360*10
	Flexible bus	30*10	50*10	150*10	360*10
Rejection Rate (%)	Conventional bus	44.6 %	63.4 %	58.2 %	27.6 %
	Ride-hailing	0	13.7 %	47.5 %	33.9 %
	NY-2017	2.1 %	22.3 %	49.9 %	39.5 %
	Flexible bus	2.1 %	18.4 %	17.4 %	20.5 %
Total Vehicle Travel Time (min)	Conventional bus	2640	2640	2640	5280
	Taxi	4500	7500	22,500	54,000
	NY-2017	476	2825	8146	19,953
	Flexible bus	389	2914	7753	21,843
Average Passenger-Detour-Time (min)	Conventional bus	10.0	8.8	9.5	5.7
	Taxi	0	0	0	0
	NY-2017	2.6	9.9	12.3	11.5
	Flexible bus	1.7	1.4	3.8	3.6
Average Passenger-Waiting-Time (min)	Conventional bus	6.9	5.2	8.2	5.7
	Taxi	0.6	4.2	5.0	4.5
	NY-2017	4.7	6.7	8.4	7.3
	Flexible bus	8.5	8.8	8.4	8.4

In summary, the conventional bus is more suitable for high-demand areas with 121 requests/km^2/h. Taxis were more suitable in low-demand areas. Because a taxi receives one request at the same time and chooses the shortest path, the average passenger-detour-time is always zero. When travel intensity increases, the taxi fleet size also increases significantly, leading to higher operational costs. Because we maintained the same number of seats in the four modes for a fair comparison, taxis apparently did not have enough vehicles, leading to over 50 % passenger rejection rate. The flexible bus was more suitable in moderate-demand areas, ranging from 20 to 50 requests/km^2/h. Flexible buses could serve 80 % of the requests when the number of requests was equal to the number of seats in the fleet. Furthermore, the flexible bus could significantly shorten the average passenger-detour-time and had a moderate average passenger-waiting-time. Compared with conventional buses, flexible buses can provide more directed services. Compared to taxis, the flexible bus significantly reduced the operational costs because of the nature of shared trips, and no significant loss in service quality was observed. For NY-2017, the overall performance of the algorithm was inferior to that of the flexible bus.

4.4. Sensitivity analysis

4.4.1. Effect of vehicle fleet size and vehicle type

Given the demand intensity in a study area, fleet size is a key factor affecting the flexible bus system performance. The parameters and settings in Case 3, the Jiading district in the morning peak hours, were used in the subsequent sensitivity analyses.

In Fig. 14, the X-axis represents the bus fleet size and the Y-axis represents the different performance measurement indexes. With an increase in fleet size, the request rejection rate decreased substantially from 75 % to less than 10 %. When there were over 70 buses, the request rejection rate remained stable at approximately 8 %. Further increasing the fleet size was no longer effective in reducing the rejection rate. The average passenger-waiting-time and average passenger-detour-time decreased slightly as the fleet size increased. Moreover, the average passenger-waiting-time decreased rapidly, and the average passenger-detour-time decreased rapidly when the fleet size was 10–60 vehicles and stabilized at 0.5 min with more than 60 vehicles. The average passenger-detour-time slightly decreased from 10.8 min to 8.7 min. The value of the objective function, which considered the cost of rejecting passengers, continued

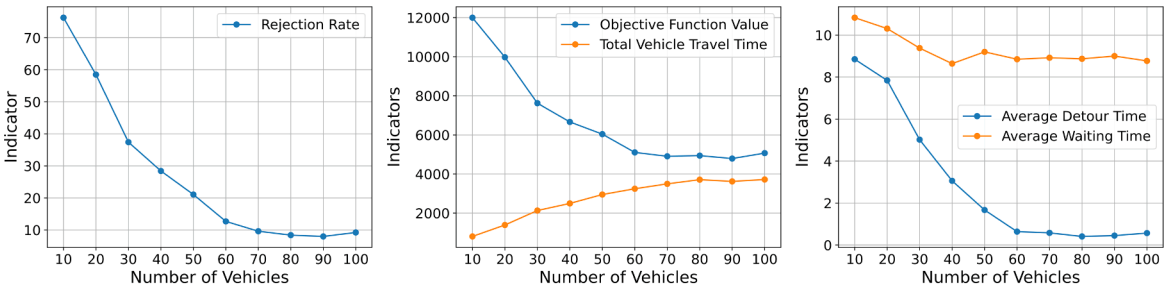

Fig. 14. Flexible bus system performances under different fleet sizes.

to decline as the passenger rejection rate declined but remained stable when there were more than 70 vehicles. The total vehicle travel time increased because more passengers were served. In conclusion, a fleet size of approximately 60–70 is the optimal choice for the system.

In Fig. 15, the number of vehicles was set to 50 and the vehicle capacity was increased from 4 seats to 15 seats. Similarly, when the vehicle capacity increased, the total number of seats increased. Hence, the rejection rate and generalised costs, considering the rejection penalty, decreased substantially. However, the average passenger-waiting-time, average passenger-detour-time, and total vehicle travel time did not significantly change. This indicates that using a large-capacity vehicle does not significantly improve the passenger service level.

In Fig. 16, the total number of seats was set to 500, but the vehicle capacity and fleet size were changed simultaneously to identify the most suitable vehicle capacity for the flexible bus system. The X-axis represents the vehicle capacity. The fleet size decreased when the vehicle capacity increased. As shown in Fig. 16(a), the request rejection rate increased substantially when a smaller number of large-capacity vehicles were used. The average passenger-detour-time also increased. The average passenger-waiting-time decreased slightly when 15-seat vehicles were used. This is probability because over 30 % requests were rejected, and they did not contribute to the passenger waiting time.

In summary, increasing the fleet size is more effective in reducing the request rejection rate than increasing the vehicle capacity. In this case, the flexible bus system performs best using 7-seat buses.

4.4.2. Effect of stochastic future demand

We tested the effects of future demand scenarios versus deterministic cases with no future requests. In this study, five future demand scenarios were considered. As observed from Table 5, the stochastic model reduced the objective function value by 1.4 %, total vehicle travel time by 1 %, request rejection rate by 1.7 %, average passenger-detour-time by 26.0 %, and average passenger-waiting-time by 2.2 %. The average passenger-detour-time decreased significantly, possibly owing to the reduction of unnecessary detours by considering future vehicle travel paths in the current time step.

4.4.3. Effect of the travel time windows of passengers

The boarding time windows of the passengers are closely related to the performance of the flexible bus system. This duration of the time window also indicates the maximum waiting time for a passenger to board a bus after the request has been issued. The duration of the passenger boarding time window was set to 5 min, 10 min, 15 min, 20 min and 25 min, and analysed the performance of the flexible bus system.

In Fig. 17, the X-axis represents the time window duration and the Y-axis represents the different performance measurement indexes. With an increase in time window duration, the request rejection rate decreased significantly from 46 % to less than 14 %. When the time window size was greater than 10 min, the request rejection rate was approximately 20 %. However, increasing the time window size had a substantially lower impact on reducing the rejection rate. With an increase in the time window size, the average passenger-waiting-time increased from 6.7 min to 8.9 min, and the average passenger-detour-time fluctuated slightly, stabilizing at approximately 1 min. The value of the objective function consisting of the passenger rejection penalty and vehicle travel cost declined quickly first and remained stable when the time window duration was greater than 15 min. This is probably because the longer the time window duration, the fewer the passengers rejected. The penalty cost of rejecting passengers and the total cost were reduced. The cost of driving a vehicle increased because it served more passengers, and the total travel time increased. However, this increase was less than the decrease in the cost of penalties for refusing passengers' requests. As shown in Fig. 17(a) and (b), with an increase in time window duration, the rejection cost decreased by 300 % and the driving cost only increased by 37 %. This indicates that a longer time window duration can arrange vehicle operation and serve passengers more effectively. In the flexible bus system, it is more appropriate to choose a passenger time window width of approximately 10–15 min, considering the balance between the passenger waiting time and the request rejection rate.

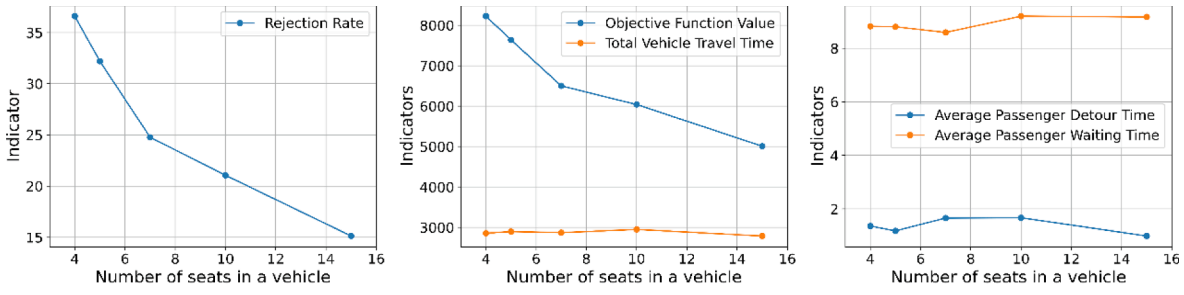

Fig. 15. Flexible bus system performances under different vehicle capacities.

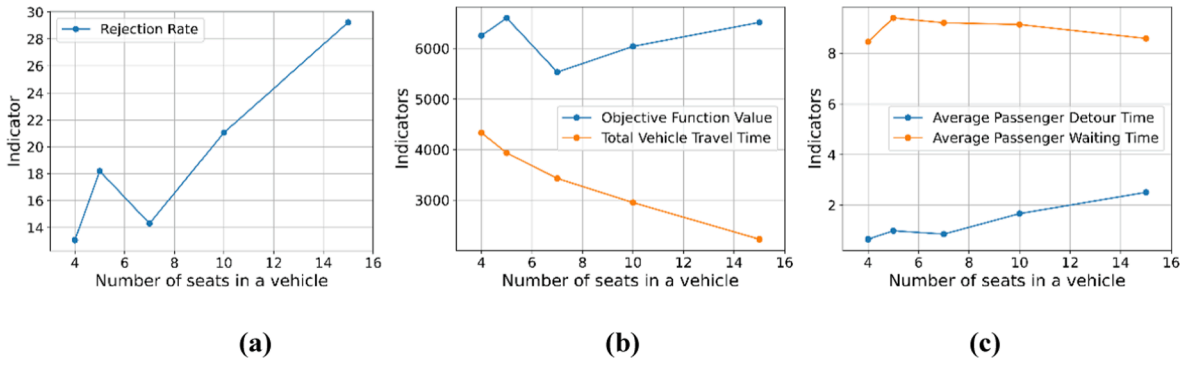

Fig. 16. Variations in each index with the total number of seats in a vehicle.

Table 5
Performance comparison under different scenarios.

Future Request	Objective Function Value	Rejection Rate	Total Vehicle Travel Time	Average Passenger-Detour-Time	Average Passenger-Waiting-Time	Calculation Time
Deterministic	5834	20.1 %	2889	1.7	9.2	2543
Stochastic	5755	19.7 %	2861	1.3	8.9	3045
Gap	−1.4 %	−1.7 %	−1%	−26.0 %	−2.2 %	19.8 %

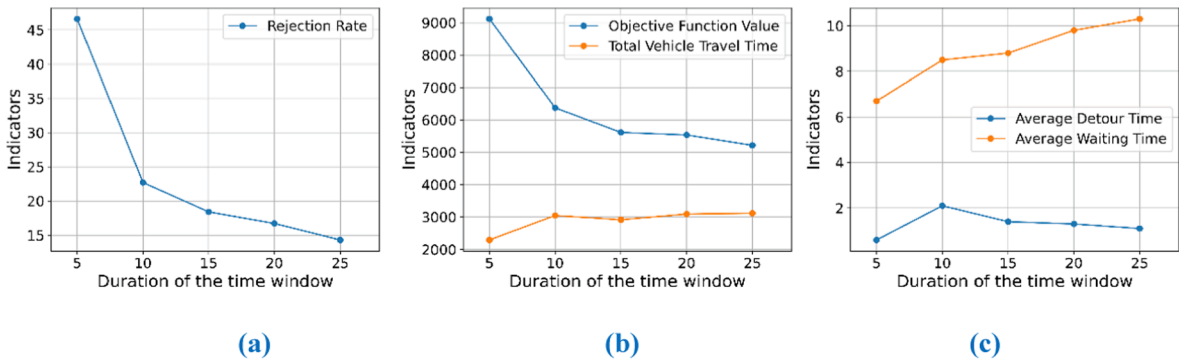

Fig. 17. Flexible bus system performance under different boarding time window sizes.

4.4.4. Effect of the time interval of optimisation horizon

The time length of the optimisation horizon is an important parameter that affects the performance of the flexible bus system. On the one hand, a longer length indicates that the system has more time to calculate and alleviate the short-sighted problem of the algorithm to obtain a better solution. On the other hand, a longer length reduces the service level to passengers. We tested four different time lengths, that is, 1 min, 3 min, 5 min, and 10 min, to examine the impacts. The maximum waiting time for the passengers was set to 15 min.

The results are shown in Fig. 18. With an increase in horizon length, the objective function value and request rejection rate decreased slightly when the time length was 1–5 min. When the time length was longer, the sub-problems of each horizon had more chances of obtaining better solutions. As a side effect, the average passenger-waiting-time increased with increasing time length. However, when the time length was 10 min, the objective function value and the request rejection rate increased rapidly. This is because a longer time was required for a long time interval when the request was input into the solution algorithm. For example, when the interval was 15 min, the request issued at 8:05 could only be calculated at 8:15. Therefore, the available time window for passengers became 8:15 to 8:20. Thus, the probability of rejecting requests increased. This may be the main reason for the increase in the objective function and the request rejection rate. In this case, we should increase the waiting time windows for the passengers. Simultaneously, a short time interval requires higher computational power.

5. Conclusion

This study formulated a two-stage stochastic model in a rolling horizon framework for the dynamic vehicle-routing problem of on-demand flexible buses. The effects of stochastic future requests were considered while planning the vehicle routes at the current time

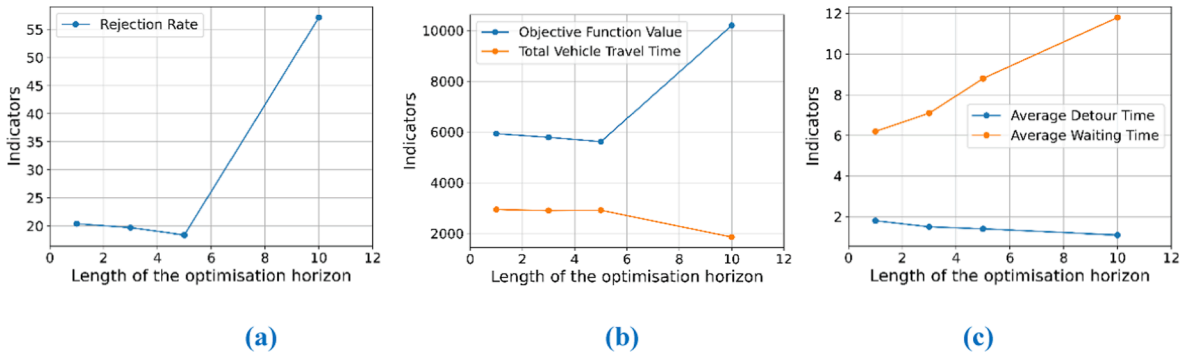

Fig. 18. Flexible bus system performance under different optimisation-horizon time intervals.

step. The aim was to minimise the sum of the vehicle operating costs and penalty for request rejection. A VSC-ALNS heuristic algorithm was proposed to solve this problem. The vector-similarity-based algorithm clusters requests and vehicles into groups first, and the ALNS plans the vehicle routes in each group. Parallel computing was used to further increase solution efficiency. The test results for four real cases in Shanghai show that the proposed method outperformed the Gurobi and ALNS. Compared to other travel modes, the flexible bus performed better under medium demand and had strong robustness against extreme demand. The conventional bus is the most cost-efficient way to service high and stable travel demands owing to the economics of scale. Taxis can cover low travel demands with a higher efficiency than shared mobility. The flexible bus can reduce the total vehicle travel time by 59 % compared to ride sharing in the case of high demand case with 3639 requests, and can reduce average-passenger-detour-time by up to 69 % compared to the conventional fixed route buses in the case of the low-demand case with 1396 requests. Future extensions of this study include the following: (1) unexpected situations such as order cancellation in actual operation will be considered in the proposed model and algorithm, and (2) accurate short-term future demand forecasting methods will be developed to improve the performance of the flexible bus system.

CRediT authorship contribution statement

Wanjing Ma: Formal analysis, Funding acquisition, Supervision. **Lin Zeng:** Methodology, Data curation, Writing – original draft, Writing – review & editing. **Kun An:** Conceptualization, Methodology, Formal analysis, Writing – review & editing, Supervision.

Declaration of Competing Interest

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

Acknowledgements

This research is supported by the National Natural Science Foundation of China (No. 72101186), the Fundamental Research Funds for the Central Universities (16002150100, 2022-5-YB-05), Shanghai Science and Technology Innovation Action Plan Project (20DZ1202805). The support is gratefully acknowledged.

References

Agatz, N.A.H., Erera, A.L., Savelsbergh, M.W.P., Wang, X., 2011. Dynamic ride-sharing: A simulation study in metro Atlanta. Transportation Research Part B-Methodological 45 (9), 1450–1464.

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., Rus, D., 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. Proceedings of the National Academy of Sciences of the United States of America 114(3), 462-467.

An, K., 2020. Battery electric bus infrastructure planning under demand uncertainty. Transport. Res. Part C-Emerg. Technol. 111, 572–587.

An, K., Lo, H.K., 2014. Service reliability–based transit network design with stochastic demand. Transportation Research Record: Journal of the Transportation Research Board 2467 (1), 101–109.

Atasoy, B., Ikeda, T., Song, X., Ben-Akiva, M.E., 2015. The concept and impact analysis of a flexible mobility on demand system. Transportation Research Part C-Emerging Technologies 56, 373–392.

Becker, H., Becker, F., Abe, R., Bekhor, S., Belgiawan, P.F., Compostella, J., Frazzoli, E., Fulton, L.M., Bicudo, D.G., Gurumurthy, K.M., Hensher, D.A., Joubert, J.W., Kockelman, K.M., Kroeger, L., Le Vine, S., Malik, J., Marczuk, K., Nasution, R.A., Rich, J., Carrone, A.P., Shen, D., Shiftan, Y., Tirachini, A., Wong, Y.Z., Zhang, M., Bosch, P.M., Axhausen, K.W., 2020. Impact of vehicle automation and electric propulsion on production costs for mobility services worldwide. Transportation Research Part A-Policy and Practice 138, 105–126.

Ben-Dor, G., Ben-Elia, E., Benenson, I., 2019. Determining an optimal fleet size for a reliable shared automated vehicle ride-sharing service. Procedia Computer Science 151, 878–883.

Bongiovanni, C., Kaspi, M., Geroliminis, N., 2019. The electric autonomous dial-a-ride problem. Transportation Research Part B-Methodological 122, 436–456.

Carotenuto, P., Martis, F., 2017. A double dynamic fast algorithm to solve multi-vehicle dial a ride problem. Transp. Res.Procedia 27, 632–639.

Chen, X.Q., Zheng, H.Y., Ke, J.T., Yang, H., 2020. Dynamic optimization strategies for on-demand ride services platform: Surge pricing, commission rate, and incentives. Transportation Research Part B-Methodological 138, 23–45.

Daganzo, C.F., 1984. Checkpoint dial-a-ride systems. Transportation Research Part B-Methodological 18 (4–5), 315–327.
Daganzo, C.F., Ouyang, Y., Yang, H., 2020. Analysis of ride-sharing with service time and detour guarantees. Transp. Res. B Methodol. 140, 130–150.
Duan, L., Wei, Y., Zhang, J., Xia, Y., 2020. Centralized and decentralized autonomous dispatching strategy for dynamic autonomous taxi operation in hybrid request mode. Transportation Research Part C: Emerging Technologies 111, 397–420.
Engels, D., Ambrosino, G., Boero, M., 2004. Demand responsive transport services: Towards the flexible mobility agency. J. Transp. Geogr. 55–73.
Flusberg, M., 1976. Innovative public transportation system for a small city: The Merrill, Wisconsin, case study. Transp. Res. Rec. 606, 54–59.
Guo, X., Caros, N.S., Zhao, J., 2021. Robust matching-integrated vehicle rebalancing in ride-hailing system with uncertain demand. Transp. Res. B Methodol. 150, 161–189.
Hasan, M.H., Van Hentenryck, P., Legrain, A., 2020. The commute trip-sharing problem. Transp. Sci. 54 (6), 1640–1675.
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 Methodol. 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. Transportation Research Part C: Emerging Technologies 111, 1–21.
Jin, S.T., Kong, H., Wu, R., Sui, D.Z., 2018. Ridesourcing, the sharing economy, and the future of cities. Cities 76, 96–104.
Ke, J.T., Qin, X.R., Yang, H., Zheng, Z.F., Zhu, Z., Ye, J.P., 2021. Predicting origin-destination ride-sourcing demand with a spatio-temporal encoder-decoder residual multi-graph convolutional network. Transportation Research Part C-Emerging Technologies 122.
Ke, J.T., Xiao, F., Yang, H., Ye, J.P., 2022. Learning to delay in ride-sourcing systems: A multi-agent deep reinforcement learning framework. IEEE Trans. Knowl. Data Eng. 34 (5), 2280–2292.
Kirchler, D., Calvo, R.W., 2013. A granular tabu search algorithm for the dial-a-ride problem. Transportation Research Part B-Methodological 56, 120–135.
Lee, E., Cen, X., Lo, H.K., 2021. Zonal-based flexible bus service under elastic stochastic demand. Transportation Research Part E: Logistics and Transportation Review 152.
Li, B., Krushinsky, D., Van Woensel, T., Reijers, H.A., 2016. The Share-a-Ride problem with stochastic travel times and stochastic delivery locations. Transportation Research Part C: Emerging Technologies 67, 95–108.
Li, X., Quadrifoglio, L., 2009. Optimal zone design for feeder transit services. Transp. Res. Rec. 2111 (2111).
Liang, X., Correia, G.H.d.A., van Arem, B., 2016. Optimizing the service area and trip selection of an electric automated taxi system used for the last mile of train trips. Transportation Research Part E: Logistics and Transportation Review 152 93, 115-129.
Liang, X., Correia, G.H.d.A., van Arem, B., 2018. Applying a Model for Trip Assignment and Dynamic Routing of Automated Taxis with Congestion: System Performance in the City of Delft, The Netherlands. Transportation Research Record 2672(8), 588-598.
Liang, X., Correia, G.H.d.A., An, K., van Arem, B., 2020. Automated taxis' dial-a-ride problem with ride-sharing considering congestion-based dynamic travel times. Transportation Research Part C: Emerging Technologies 112, 260-281.
Liu, T., Ceder, A., 2015. Analysis of a new public-transport-service concept: Customized bus in China. Transp. Policy 39, 63–76.
Liu, J., Mirchandani, P., Zhou, X., 2020. Integrated vehicle assignment and routing for system-optimal shared mobility planning with endogenous road congestion. Transportation Research Part C: Emerging Technologies 117.
Lv, C., Zhang, C., Lian, K., Ren, Y., Meng, L., 2020. A hybrid algorithm for the static bike-sharing re-positioning problem based on an effective clustering strategy. Transp. Res. B Methodol. 140, 1–21.
Lyu, Y., Chow, C.-Y., Lee, V.C.S., Ng, J.K.Y., Li, Y., Zeng, J., 2019. CB-Planner: A bus line planning framework for customized bus systems. Transportation Research Part C: Emerging Technologies 101, 233–253.
Ning, F., Jiang, G., Lam, S.-K., Ou, C., He, P., Sun, Y., 2021. Passenger-centric vehicle routing for first-mile transportation considering request uncertainty. Inf. Sci. 570, 241–261.
Qiu, Z., Liu, L., Li, G., Wang, Q., Xiao, N., Lin, L., IEEE, 2019. Taxi Origin-Destination demand prediction with contextualized spatial-temporal network, 2019 IEEE International Conference on Multimedia and Expo (ICME), PP. 760-765.
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.
Schilde, M., Doerner, K.F., Hartl, R.F., 2011. Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. Comput. Oper. Res. 38 (12), 1719–1730.
Shaheen, S., Cohen, A., 2019. Shared ride services in North America: Definitions, impacts, and the future of pooling. Transp. Rev. 39 (4), 427–442.
Soeffker, N., Ulmer, M.W., Mattfeld, D.C., 2022. Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. Eur. J. Oper. Res. 298 (3), 801–820.
SteadieSeifi, M., Dellaert, N., Van Woensel, T., 2021. Multi-modal transport of perishable products with demand uncertainty and empty repositioning: A scenario-based rolling horizon framework. EURO Journal on Transportation and Logistics 10.
Syed, A.A., Kaltenhaeuser, B., Gaponova, I., Bogenberger, K., IEEE, 2019. Asynchronous adaptive large neighborhood search algorithm for dynamic matching problem in ride hailing services, 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 3006-3012.
Tafreshian, A., Masoud, N., 2020. Trip-based graph partitioning in dynamic ridesharing. Transportation Research Part C: Emerging Technologies 114, 532–553.
Ulmer, M.W., 2019. Anticipation versus reactive reoptimization for dynamic vehicle routing with stochastic requests. Networks 73 (3), 277–291.
Vodopivec, N., Miller-Hooks, E., 2017. An optimal stopping approach to managing travel-time uncertainty for time-sensitive customer pickup. Transp. Res. B Methodol. 102, 22–37.
Winter, S., Nittel, S., 2006. Ad hoc shared-ride trip planning by mobile geosensor networks. Int. J. Geogr. Inf. Sci. 20 (8), 899–916.
Xiao, L.L., Huang, H.J., Liu, R.H., 2015. Congestion behavior and tolls in a bottleneck model with stochastic capacity. Transp. Sci. 49 (1), 46–65.
Xu, H.Y., Pang, J.S., Ordonez, F., Dessouky, M., 2015. Complementarity models for traffic equilibrium with ridesharing. Transp. Res. B Methodol. 81, 161–182.
Zhan, X., Szeto, W.Y., Shui, C.S., Chen, X., 2021. A modified artificial bee colony algorithm for the dynamic ride-hailing sharing problem. Transportation Research Part E: Logistics and Transportation Review 150.
Zhao, Y., Guo, X.L., Liu, H.X., 2021. The impact of autonomous vehicles on commute ridesharing with uncertain work end time. Transp. Res. B Methodol. 143, 221–248.
Zhou, F., He, Y., Zhou, L., 2019. Last mile delivery with stochastic travel times considering dual services. IEEE Access 7, 159013–159021.