Optimizing first-mile ridesharing services to intercity transit hubs

He, Ping; Jin, Jian Gang; Schulte, Frederik; Trépanier, Martin

DOI
10.1016/j.trc.2023.104082

Publication date
2023

Document Version
Final published version

Published in
Transportation Research Part C: Emerging Technologies

Citation (APA)
He, P., Jin, J. G., Schulte, F., & Trépanier, M. (2023). Optimizing first-mile ridesharing services to intercity transit hubs. *Transportation Research Part C: Emerging Technologies*, *150*, Article 104082. https://doi.org/10.1016/j.trc.2023.104082

Transportation Research Part C 150 (2023) 104082

Contents lists available at ScienceDirect

Transportation Research Part C

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

Optimizing first-mile ridesharing services to intercity transit hubs ☆

Ping He [a], Jian Gang Jin [a,*], Frederik Schulte [b], Martin Trépanier [c,d]

[a] School of Naval Architecture, Ocean & Civil Engineering, and State Key Laboratory of Ocean Engineering, Shanghai Jiao Tong University, Shanghai, China

[b] Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, Delft, The Netherlands

[c] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada

[d] Department of Mathematics and Industrial Engineering, Polytechnique Montreal, Montreal, Canada

ARTICLE INFO

Keywords:
Public transportation
First-mile ridesharing
Intercity transportation hubs
Routing with large luggage
Travel time uncertainty

ABSTRACT

Travel to intercity transportation hubs, such as railway stations and airports, can be the most troublesome and inefficient part of the entire air/railway travel journey, as travelers often carry large luggage and have stringent arrival time requirements. Taking public transportation, such as metro and bus services, is inconvenient to carry luggage and less reliable in arrival time while taking taxi services could be less economical. As a result, providing reliable and convenient yet economical on-demand first-mile services for travelers to intercity transportation hubs is essential. This paper proposes a ridesharing approach for the first-mile transport system for travelers heading towards the intercity transportation hub and develops a mixed-integer linear programming (MILP) model with the objective of minimizing the total operating costs for ridesharing service operators. The MILP model considers (1) large luggage that may occupy seats when the car trunk is not large enough to place them; (2) passengers' requirements on arrival time and ride time; and (3) travel time uncertainty ensuring that riders' arrival time and ride time can be satisfied. A tailored adaptive large neighborhood search algorithm with an acceleration strategy is developed for obtaining robust near-optimal solutions within a reasonable time. To assess the solution quality, the MILP model is reformulated as a set-partitioning model, and the column generation algorithm is leveraged to determine a tight lower bound; a greedy algorithm is introduced to obtain an upper bound. Computational experiments on Shanghai South Railway Station demonstrate that ridesharing is an effective strategy for reducing overall travel costs while meeting the first-mile travel demand. In addition, it is essential to consider luggage and travel time uncertainty for determining ridesharing schemes.

1. Introduction

High-speed rail and air transportation have seen a sharp increase in passenger volume over the past 20 years as a result of their crucial role in intercity travel (Chen and Lin, 2016; Wang et al., 2020). However, travel to a high-speed rail station or airport (also known as the intercity transportation hub) from the home, office, or hotel often has problems of inconvenience, poor economy, and low reliability, which can be defined as the first-mile problem heading to intercity transportation hubs. Passengers usually take the metro or bus to intercity transportation hubs (Zhou et al., 2022), which is inconvenient for riders with large luggage (Fang et al., 2015). Some passengers take taxis or online ride-hailing services to the transportation hub, but this is less economical, the arrival

☆ This article belongs to the Virtual Special Issue on "IG005586: VSI: On-Demand Transportation".

* Corresponding author.
E-mail address: jiangang.jin@sjtu.edu.cn (J.G. Jin).

https://doi.org/10.1016/j.trc.2023.104082
Received 3 April 2022; Received in revised form 20 November 2022; Accepted 24 February 2023
Available online 10 March 2023

time is unreliable, and it is susceptible to factors like traffic congestion (Zhou et al., 2022), which could result in travelers missing their next trip. As a result, it is crucial to provide passengers heading toward an intercity transportation hub with convenient, affordable, and stable on-demand first-mile (FM) transportation services.

Ridesharing by passenger cars has been considered an effective way to provide FM/LM (last-mile) door-to-door services (Shaheen and Chan, 2016; Bian and Liu, 2019b; Bian et al., 2020; Chen et al., 2020), as it can offer riders high-quality and low-cost transportation services. Nonetheless, most of the existing studies focus on FM ridesharing to metro hubs while only a few studies (Bian and Liu, 2017) focus on the intercity transit hub. It is more challenging to design first-mile ridesharing services to the intercity transportation hub than to the metro station, which necessitates taking the following factors into account.

First, passengers heading for an intercity transportation hub often need to transfer to the train or airplane to catch up with the next trip, so they want to arrive at the hub earlier than the scheduled departure time of the next trip. Second, riders may have different tolerances on the maximum ride time. For instance, riders closer to the transportation hub typically request shorter ride times. Third, when planning ridesharing routes, the travel time uncertainty must be considered to ensure that passengers can smoothly transfer to the next trip and that riders' ride times do not exceed their tolerances. Although some literature on FM ridesharing to metro hubs considers maximum ride times (Bian and Liu, 2019a; Bian et al., 2020; Chen et al., 2020) and arrival deadlines of riders (Bian and Liu, 2019a; Bian et al., 2020), they neglect to consider the impact of travel time uncertainty. Fourth, the target passengers often carry large luggage. If permitted by local laws and regulations, large luggage may be placed on seats when the number of pieces of large luggage exceeds the car trunk's carrying capacity.

This study focuses on the first-mile ridesharing to the intercity transportation hub (FMRITH), where passenger cars are used to provide FM ridesharing services to passengers within a specified service radius (for instance, 15 kilometers) of an intercity transportation hub. Since passengers know their exact departure time for the next trip, they can book FM services with service providers in advance, for instance, 2 h, according to their transportation requirements. The service provider can utilize the rolling horizon method to dynamically group requests according to their latest arrival time, then assign passengers to cars, plan routes for cars, and notify requests accordingly. To render on-demand door-to-door services, the service provider needs to make decisions based on the objective of minimizing the total operation cost as well as constraints such as large luggage occupying seats, riders' latest arrival times and maximum ride times, and travel time uncertainty. This cost-effective, stable, and flexible mode can tackle the problem of FM travel to intercity transportation hubs to some extent since it not only considers some special factors, such as passengers' larger luggage and the arrival time requirement, but also avoids the expensive operating costs for public transportation operators, high travel costs for passengers, and the inconvenience of fixed pickup times and places for public transportation.

Our contributions consist of: (a) proposing a ridesharing approach for the first-mile transport system for travelers heading towards an intercity transportation hub; (b) developing a MILP model that minimizes the total operating cost for the ridesharing service provider and that considers key factors including large luggage, arrival time and ride time requirements, and travel time uncertainty; (c) presenting a tailored adaptive large neighborhood search algorithm (ALNS) with an acceleration strategy to solve real-world instances and introducing the column generation algorithm and a greedy algorithm to evaluate the solution quality of the ALNS algorithm; (d) demonstrating through computational experiments that the large luggage and travel time uncertainty must be considered in the FM problem to an intercity transportation hub and that the proposed models and algorithms can effectively solve the FMRITH problem.

The remainder of the paper is organized as follows. Section 2 reviews the relevant literature and positions our work. Section 3 elaborates on the FMRITH problem and introduces a MILP model. The customized ALNS algorithm is developed in Section 4. In Section 5, we introduced a set-partitioning model derived from the MILP model and the column generation algorithm as well as a greedy algorithm. Numerical experiments for several cases are conducted in Section 6. Conclusions and future work are summarized in Section 7.

2. Literature review

The existing travel modes for the FM/LM problem include walking, shared bicycles, feeder buses, private cars, taxis, etc (Liu et al., 2012; Huang et al., 2021). Walking is the primary way for FM travel problems, but walking distances to metro/bus stations are often long bringing great disutility to passengers (El-Geneidy et al., 2014). Compared to walking, the bicycle can travel faster and easily reach longer distances (Hochmair, 2015; Lee et al., 2016). Bicycles can effectively render door-to-door service (Zuo et al., 2018) and improve transit accessibility (Boarnet et al., 2017; Zuo et al., 2020). However, traveling by bicycle can be affected by weather conditions (El-Assi et al., 2017; Kim, 2018). In northeast China, for instance, the weather conditions are not suitable for bicycles for half a year. In addition, there are no bicycle lanes on the roads in many cities (Winters et al., 2011; Zuo et al., 2018). As a result, riding bicycles to transit stations is not available in some cities (Campbell and Brakewood, 2017). The feeder bus is safer and less affected by the weather than the bicycle. Demand-responsive transportation (DRT) (Dessouky et al., 2003; Quadrifoglio et al., 2008; Masson et al., 2014; Chen and Nie, 2017; Mahéo et al., 2019; Xiong et al., 2015) is the most common operation mode for feeder buses since it is more economical than conventional bus operating modes. However, most DRTs do not provide door-to-door transportation, so they cannot completely tackle the passenger's FM problem. In addition, riders may await the DRT service for a long time due to limited service capabilities, especially during peak hours. The taxi service is convenient and flexible (Mahéo et al., 2019) and can render door-to-door transportation, but passengers' travel costs are relatively high (Ma et al., 2019).

Based on the above-mentioned situations, Shen et al. (2018), Stiglic et al. (2018), Bian and Liu (2019a,b), Bian et al. (2020), and Chen et al. (2020) put forward utilizing the ridesharing mode of passenger cars to offer affordable and flexible door-to-door first-mile transportation services for riders. For instance, Shen et al. (2018) simulated a public transit system integrating shared

Table 1
The comparison of some first-mile access modes.

First-mile access modes	Characteristics	Application conditions
Walking	Low speed, flexible	Short walking distance (<500 m)
Bicycle	Faster than walking, flexible	Good weather, bicycle lanes available
Shuttle bus	Fixed pick-up stations	High population density
Taxi	Flexible but expensive	Any conditions are applicable
Ridesharing	Flexible and affordable	Any conditions are applicable

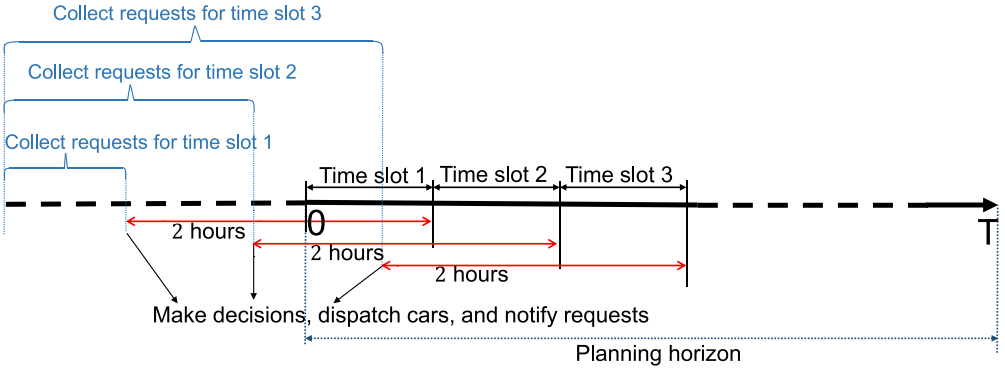


Fig. 1. An illustrative example of applying the rolling horizon approach to the first-mile ridesharing problem.

autonomous vehicles to provide first-mile travel services during morning peak hours. Stiglic et al. (2018) investigated a seamless integration of ridesharing and public transit, which can effectively solve the first- and last-mile problems and increase the use of public transport. Table 1 summarizes the characteristics and application scenarios for FM travel modes based on the aforementioned studies. In comparison to the other modes, ridesharing appears as a versatile and affordable approach for first-mile travel problems.

In the literature on FM ridesharing, most studies (Shen et al., 2018; Bian and Liu, 2019a,b; Chen et al., 2020; Jiang et al., 2020; Ning et al., 2021; Kumar and Khani, 2021; Huang et al., 2022) focus on FM ridesharing to metro/bus transit hubs, while there is comparatively scarce research on the FM ridesharing to intercity transit hubs, such as airports and railway stations. For instance, Bian and Liu (2019a,b) designed a novel mechanism for FM ridesharing services to metro stations and a heuristic algorithm to solve this problem. Chen et al. (2020) explored leveraging AVs to provide FM ridesharing services to metro stations and designed a cluster-based solution method for it. To the best of our knowledge, there is only one paper that focuses on the first-mile ridesharing to the intercity transit hub: Bian and Liu (2017) devised a simulated-based algorithm to solve the first-mile ridesharing problem for a railway station, which considers passengers' latest arrival time and the connection with the train schedule. Therefore, we find that the existing literature has not considered the characteristics of FM to intercity transportation hubs, such as passengers often carrying large luggage and riders' requirements on arrival time and maximum ride time.

Motivated by the above, this study focuses on the FM ridesharing problem of intercity transportation hubs with consideration of minimizing total operational costs for ridesharing service operators, riders' large luggage and requirements on the latest arrival time as well as the maximum ride time, and the uncertainty of travel time. To address this new and challenging problem efficiently, we propose a MILP model and design three algorithms.

3. First-mile ridesharing to the intercity transportation hub

3.1. Problem description

Consider an airport as the transportation hub, a group of passengers who need to travel from their locations to the airport, and a ridesharing service operator who runs a fleet of homogeneous passenger cars and provides on-demand FM transportation services to passengers. Since the next flight's departure time is already known, passengers who need the on-demand FM services can book a service with the service provider in advance (for instance, 2 h) based on their travel information: the geographical locations, the number of riders, the number of pieces of large luggage, the latest arrival time, the maximum ride time, and the earliest pickup time. The latest arrival time is indispensable for each rider since they need to transfer to their flight. Detour tolerance varies among riders, and as a result, different riders may have different maximum ride time requirements. Riders typically decide the earliest pickup time based on their distance to the airport and the scheduled departure time of their flights. If the car arrives earlier than the rider's earliest service time, it needs to wait until the earliest service start time to pick up riders. Riders' latest pickup time can be determined by subtracting the direct ride time to the transportation hub from the latest arrival time or by adding a certain amount of time, such as 30 min, to the earliest pickup time.

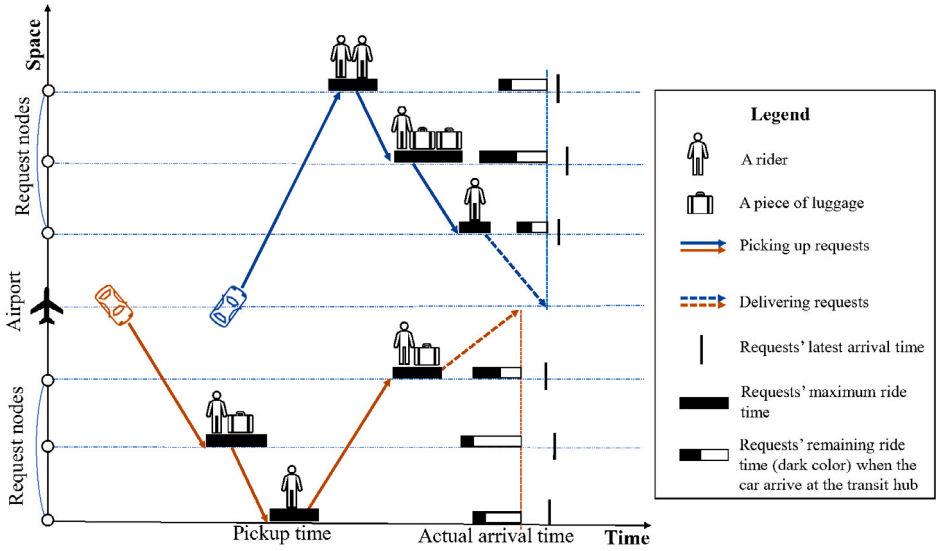


Fig. 2. An illustrative example of first-mile ridesharing to an intercity transportation hub.

Based on the received reservation information, the ridesharing service operator decides the ridesharing route for each rider. To better meet riders' dynamic demands and reduce the dimension of decision-making, the common practice for designing first-mile ridesharing schemes is utilizing the rolling horizon method to group requests according to their preferred pickup time (Chen et al., 2020). The entire planning period [0, T] is typically divided into a series of time slots, and a ridesharing decision is made before the start of each time slot for those travel demands whose earliest pickup times fall within this time slot. For passengers going to an intercity transportation hub, their first-mile travel distances are often long (generally greater than 10 km), and passengers with close earliest pickup times are often less likely to be visited by the same route, so the conventional way of dividing time slots according to passengers' earliest pickup times would not be applicable. The operator can divide time slots according to passengers' latest arrival times and design ridesharing routes a certain amount of time (δ) in advance, such as two hours, before the end of each time slot. Fig. 1 shows three time slots, and riders whose latest arrival times fall within each time slot should book the service at least δ hours in advance, as the service provider needs time to schedule cars to pick them up and take them to the transit hub. We assume that the operator can schedule enough cars to service requests of each time slot. After completing the ridesharing decision, the operator dispatches cars to pick up riders and notifies them of their ridesharing information. Fig. 2 shows an illustrative example of a first-mile ridesharing scheme for riders of a time slot, which is a time-space figure with the horizontal axis representing continuous time and the vertical axis representing discrete spatial locations. Two homogeneous cars are scheduled to serve six requests (each request consists of riders and pieces of large luggage), and the blue car departs later than the orange car.

The goal of the problem is to determine the lowest cost ridesharing scheme which needs to satisfy the following constraints: (1) Cars depart from and eventually return to the intercity transportation hub; (2) The number of passengers and pieces of large luggage carried by each car cannot exceed its carrying capacity; (3) Each request must be serviced; (4) Requests' pickup time must be met; (5) Even in the case of uncertain travel time, the cars should arrive at the transportation hub no later than the latest arrival time of requests it serves, and the riders' ride time cannot exceed their maximum ride time.

3.2. Mathematical model

We represent the problem on a directed graph $G(N, A)$, N and N_1 correspond to the set of all nodes and request nodes respectively. 0 and $n+1$ are used to denote the intercity transportation hub. A represents the set of arcs between all nodes. The general nomenclature is shown in Table 2.

3.2.1. Travel time uncertainty

For each arc $(i, j) \in A$, we assume that the travel time takes values in the range $[t_{ij}, t_{ij} + \hat{t_{ij}}]$, where t_{ij} is the nominal value of travel time and $\hat{t_{ij}}$ denotes the maximum deviation. Note that while the range would be defined as $[t_{ij} - \hat{t_{ij}}, t_{ij} + \hat{t_{ij}}]$ in some of the robust optimization problems, the worst-case scenario only occurs at the right-hand side of the range for the travel time (Munari et al., 2019), since arriving early at request nodes or destination does not affect passengers' travels. Therefore, the one-sided range $[t_{ij}, t_{ij} + \hat{t_{ij}}]$ is adequate to portray the travel time uncertainty. In this study, we employ the uncertain parameter Γ, also known as "budgets" proposed by Bertsimas and Sim (2004), to control the uncertainty level of travel time. $\Gamma = 0$ means the uncertainty is not

Table 2
Parameters and decision variables.

Notations	Descriptions
Set	
N	Set of all nodes, $N = N_1 \cup \{0, n+1\}$
N_1	Set of request nodes, $N_1 = \{1, 2, 3, \dots, n\}$
A	Set of arcs, $A = \{(i,j) \mid i, j \in N, i \neq j\}$
Parameters	
M	A sufficiently large positive constant
Γ	Budget of the travel time uncertainty
γ	Travel time uncertainty level, $\gamma \in \{0, 1, 2, \dots, \Gamma\}$
p_i	Number of passengers of request i, $i \in N_1$
ρ_i	Number of pieces of large luggage of request i, $i \in N_1$
e_i	Earliest service start time for request i, $i \in N_1$
l_i	Latest service start time for request i, $i \in N_1$
Q	Number of seats in each car
B	Number of pieces of luggage that can be placed on the trunk of each car
L	Number of pieces of luggage that can be placed on a seat
d_{ij}	Travel distance between requests i and j (km), $i, j \in N$
c	Transportation cost per kilometer $(\$)$, $i, j \in N$
d_i	Latest arrival time at the transportation hub of request i, $i \in N_1$
r_i	Maximum ride time for request i, $i \in N_1$
t_{ij}	Travel time between requests i and j, $i, j \in N$
$\hat{t_{ij}}$	Maximum deviation of travel time t_{ij} $i, j \in N$
Decision variables	
x_{ij}	$\in \{0, 1\}$. 1 if arc (i,j) is traversed; otherwise, 0
ξ_i	$\in \{\mathbb{Z}^+ \cup \{0\}\}$. Number of riders carried after leaving request i
ζ_i	$\in \{\mathbb{Z}^+ \cup \{0\}\}$. Number of pieces of luggage carried after leaving request i
$\varpi_{i\gamma}$	≥ 0. Service start time for request i under uncertainty level γ
$\tau^i_{n+1,\gamma}$	≥ 0. Arrival time for request i under uncertainty level γ

taken into account, which assumes that travel times are deterministic. Then, we can use $\mathcal{U}$, a polyhedral uncertain set, to describe all the travel time uncertainty for each route.

$$\mathcal{U}_k = \left\{ \tilde{t} \in \mathbb{R}_+^{|A|} \middle| \tilde{t_{ij}} = t_{ij} + \xi_{ij}\hat{t_{ij}}, \sum_{(i,j)\in A} \xi_{ij} \leq \Gamma, 0 \leq \xi_{ij} \leq 1, (i,j) \in A \right\} \tag{1}$$

Restricted by the service time window of each request, the pickup time at each request cannot be directly modeled as the departure time of the vehicle plus the travel time of arcs traversed by the vehicle ($t_j \geq t_0 + \sum_m \sum_n t_{mn} x_{mn}$, t_{mn} denotes the travel time on arc (m,n), and $x_{mn} \in \{0,1\}$ denotes whether the arc (m,n) is traversed), and it can only be modeled with the recursive approach ($t_j \geq t_i + s_i + t_{ij}$, s_i is the service duration at request node i). The classical robust optimization method based on the dual theory is only applicable to deal with the uncertainty of t_{mn} in $\sum_m \sum_n t_{mn} x_{mn}$, and it is not applicable to deal with the travel time uncertainty in this recursive inequality ($t_j \geq t_i + s_i + t_{ij}$). However, the travel time uncertainty can be portrayed accurately via the recursive Eq. (2) based on the uncertainty set defined in (1), just like Munari et al. (2019) and Zhang et al. (2022) did. Let $\mathcal{T}_{i,\gamma}$ be the earliest visiting time at the node N_i when up to γ ($\leq \Gamma$) route segments' travel times reaching the worst-case values. $\mathcal{T}_{i,\gamma}$ can be computed according to the following recursion equations:

$$\mathcal{T}_{i,\gamma} = \begin{cases} max\{e_i, \mathcal{T}_{i-1,\gamma} + t_{i-1,i}\}, & \text{if } \gamma = 0 \\ max\{e_i, \mathcal{T}_{i-1,\gamma} + t_{i-1,i}, \mathcal{T}_{i-1,\gamma-1} + t_{i-1,i} + \hat{t_{ij}}\}, & \text{if } \gamma > 0 \end{cases} \tag{2}$$

To ensure the route feasibility, each route must satisfy $\mathcal{T}_{i,\gamma} \leq l_i$ for all $\gamma \in \{0, 1, \dots, \Gamma\}$ and each request i visited by the route. Since the deviations in travel time may be offset by the waiting time at the request node, the value of Γ usually does not take a too large value. The detailed analysis can be found in Munari et al. (2019) and it is consistent with the numerical experimental results in Section 6.4. This recursive equation efficiently handles the travel time uncertainty of route segments.

3.2.2. Mathematical model

Based on the notations defined in Table 2 and the idea of modeling travel time uncertainty, we developed the following mathematical formulation for the first-mile ridesharing to the intercity transit hub:

$$min \sum_{i\in N} \sum_{j\in N} c d_{ij} x_{ij} \tag{3}$$

subject to:

$$\sum_{j\in N \cap j\neq\{i\cup 0\}} x_{ij} = 1, \forall i \in N_1 \tag{4}$$

$$\sum_{j\in N\cap j\neq\{i\cup 0\}} x_{ij} = \sum_{j\in N\cap j\neq i} x_{ji}, \forall i \in N_1 \tag{5}$$

$$\xi_j \geq \xi_i + p_i - M(1 - x_{ij}), \forall i, j \in N_1 \tag{6}$$

$$\zeta_j \geq \zeta_i + \rho_i - M(1 - x_{ij}), \forall i, j \in N_1 \tag{7}$$

$$\xi_i + max\left\{0, \left\lceil \frac{\zeta_i - B}{L} \right\rceil \right\} \leq Q, \forall i \in N_1 \tag{8}$$

$$\varpi_{j\gamma} \geq \varpi_{i\gamma} + t_{i,j} - M(1 - x_{ij}), \forall i, j \in N_1, \forall \gamma \in \{0, \ldots, \Gamma\} \tag{9}$$

$$\varpi_{j\gamma} \geq \varpi_{i\gamma-1} + t_{i,j} + \widehat{t_{ij}} - M(1 - x_{ij}), \forall i, j \in N_1, \forall \gamma \in \{1, \ldots, \Gamma\} \tag{10}$$

$$e_i \leq \varpi_{i\gamma} \leq l_i, \forall i \in \ N_1, \forall \gamma \in \{0, \ldots, \Gamma\} \tag{11}$$

$$\tau^j_{n+1,\gamma} \geq \tau^i_{n+1,\gamma} - M(1 - x_{ij}), \forall i, j \in N_1, \forall \gamma \in \{0, \ldots, \Gamma\} \tag{12}$$

$$\tau^j_{n+1,\gamma} \leq \tau^i_{n+1,\gamma} + M(1 - x_{ij}), \forall i, j \in N_1, \forall \gamma \in \{0, \ldots, \Gamma\} \tag{13}$$

$$\tau^j_{n+1,\gamma} \geq \varpi_{j\gamma} + t_{j,n+1} - M(1 - x_{j,n+1}), \forall j \in N_1, \forall \gamma \in \{0, \ldots, \Gamma\} \tag{14}$$

$$\tau^j_{n+1,\gamma} \geq \varpi_{j,\gamma-1} + t_{j,n+1} + \widehat{t_{j,n+1}} - M(1 - x_{j,n+1}), \forall j \in N_1, \forall \gamma \in \{1, \ldots, \Gamma\} \tag{15}$$

$$\tau^j_{n+1,\gamma} \leq d_j, \forall j \in N_1, \forall \gamma \in \{0, \ldots, \Gamma\} \tag{16}$$

$$\tau^j_{n+1,\Gamma} - \varpi_{j0} \leq r_j, \forall j \in N_1 \tag{17}$$

$$x_{ij} \in \{0, 1\}, \forall i, j \in N \tag{18}$$

$$\xi_i \in \{\mathbb{Z}^+ \cup \{0\}\}, \forall i \in N_1. \tag{19}$$

$$\zeta_i \in \{\mathbb{Z}^+ \cup \{0\}\}, \forall i \in N_1. \tag{20}$$

$$\varpi_{i\gamma} \geq 0, \forall i \in N_1, \forall \gamma \in \{0, \ldots, \Gamma\} \tag{21}$$

$$\tau^i_{n+1,\gamma} \geq 0, \forall i \in N_1, \forall \gamma \in \{0, \ldots, \Gamma\} \tag{22}$$

The objective function (3) minimizes the overall transportation cost. Constraint (4) guarantees that each request must be serviced. Constraint (5) ensures flow conservation. Constraints (6) and (7) respectively denote the carried number of riders and pieces of large luggage. Constraint (8) ensures that the total number of riders and pieces of luggage at each request node cannot exceed the carrying capacity of cars. Constraints (9)–(11) confine the pickup time at each request, which is based on the recursive Eq. (2). Constraints (12) and (13) impose that two requests sharing a trip reach the hub at the same time. Constraints (14) and (15) denote the latest arrival time for each request. Constraint (16) ensures that the arrival time at the intercity transportation hub must meet passengers' requirements. Constraint (17) ensures that riders' ride times must be less than their maximum ride times. Constraints (18)–(22) define the domain of decision variables. Constraint (8) is a nonlinear constraint, which can be linearized into constraints (23) and (24).

$$L\xi_i - \zeta_i - B \leq LQ, \forall i \in N_1 \tag{23}$$

$$\xi_i \leq Q, \forall i \in N_1 \tag{24}$$

4. Solution methodology

The FMRITH problem can be solved by off-the-shelf solvers (for instance, CPLEX and GUROBI) for small-scale cases, while it is time-consuming to solve large-scale instances since the MILP model contains a large set of integer variables and big-M constraints. To solve the large-scale FMRITH problem more efficiently, we design an algorithm based on the framework of Adaptive Large Neighborhood Search (Pisinger and Ropke, 2007).

4.1. Framework of the ALNS algorithm

Algorithm 1 lays out the main steps of the ALNS algorithm, including initial solution generation and iteratively neighborhood search by removal and repair operators, which are selected by roulette-wheel procedure. To adaptively adjust the search scope, the number of removed and inserted nodes of requests is dynamically adjusted by the parameter Π, which increases gradually with the iteration numbers of non-improved solution (NIIts). $\zeta \in [0,1]$ is the ratio of removing the requests. To avoid local optimal solutions in the search process, we adopt the Metropolis rule (Metropolis et al., 1953) to accept new solutions. Parameter T is the temperature for the Metropolis rule, and the linear cooling rate $\mathscr{R}$ is employed to lower the temperature. The ALNS is terminated when a given number of iterations are finished. Please see Algorithm 1 for the detailed adaptive large neighborhood search procedure.

Algorithm 1: A robust adaptive large neighborhood search

Input: instance data, removal operators, repair operators, and other algorithm parameters
Output: the best solution (S_{best})
construct an initial solution S_{new} by **insert algorithm**;
set $S_{best} \leftarrow S_{new}$, $S_{current} \leftarrow S_{new}$, $NIIts \leftarrow 0$;
while *the termination conditions of ALNS are not met* **do**
    $\Pi \leftarrow min\left\{\zeta_{min} + \zeta_{rate} \times NIIts\ , \zeta_{max}\right\} \times$total number of requests;
    select removal operator ($R0$) and destroy $S_{current}$ by $R0$;
    select repair operator ($R1$) and repair $S_{current}$ by $R1$;
    if S_{new} *is better than* S_{best} **then**
        $S_{best} \leftarrow S_{new}$, $S_{current} \leftarrow S_{new}$, NIIts $\leftarrow$0;
    else if S_{new} *is better than* $S_{current}$ **then**
        $S_{current} \leftarrow S_{new}$, NIIts $\leftarrow$ 0;
    else if S_{new} *is accepted by the Metropolis criterion* **then**
        $S_{current} \leftarrow S_{new}$, NIIts= NIIts+1;
    else
        NIIts= NIIts+1;
    update Π, T, scores, and weights of removal and repair operators;
return S_{best};

4.2. Initial solution

We use the randomly insert algorithm to construct an initial solution. The basic idea is that we first construct an empty route and then insert unvisited requests to this route one by one. If no more requests can be inserted into this route, we generate another empty route and continue to insert requests into it until all the requests are inserted into routes. Algorithm 2 defines the detailed procedure of generating initial solution.

Algorithm 2: Insert algorithm

Input: requests, cars
Output: set of routes (R)
uninserted requests $\leftarrow$ all requests;
while |*uninserted requests*| > 0 **do**
    generate a new empty route r;
    for *each uninserted request* **do**
        for *all positions in the route* **do**
            new route $\leftarrow$ insert the request to this position;
            check the feasibility of the new route by **Algorithm 3**;
            if *the new route is feasible* **then**
                delete this uninserted request, $r \leftarrow$ new route, and **break**;
    add r to R;
return R

4.3. Feasibility check

During the process of initial solution generation and neighborhood search, the feasibility check for routes is a crucial step. Regarding this problem, the carrying capacity, the pickup time window, and the maximum ride time as well as the latest arrival time

considering travel time uncertainty need to be checked, which is more complicated than traditional routing problems. Consequently, we design a tailored algorithm (Algorithm 3) to verify the route feasibility for this problem.

To begin, we check whether the carried number of passengers and pieces of large luggage meet the car's capacity constraint. Then, the actual pickup time window for each request and the minimum cumulative route duration from the depot to this node can be determined by "moving" the departure time window to each node and calculating the intersection of the time window of the node (see Algorithm 4). More details can be found in Algorithm 4, in which the idea of dynamic programming has been leveraged to compute corresponding time information. Based on these computational results, the feasibility of each passenger's latest arrival time at the intercity transportation hub can be checked trivially. Last, we can check whether the worst minimum duration of the sub-route starting from the incumbent node, with an initial time window of $[es_k^0, l_k]$ and the new feasible latest arrival time, is less than the rider's maximum ride time.

Algorithm 3: Route feasibility check

Input: a route
Output: feasibility of the route
set initial parameters: es_k^γ, $ls_k^\gamma \leftarrow 0$ (the earliest and latest service start time for request k under uncertainty level of γ), $dr_k^\gamma \leftarrow 0$ (route duration from depot to k);
//check the capacity feasibility for this route;
if *the capacity constraint is infeasible* **then**
    return false;
//calculate the total route duration and service time windows;
for *each request k in this route* **do**
    for *each uncertainty level of* γ ($\gamma \le \Gamma$) **do**
        calculate es_k^γ, ls_k^γ, *and* dr_k^γ by **Algorithm 4**;
get the worst minimum duration of this route $(dr^\Gamma) \leftarrow dr_k^\Gamma$ of the last request in the route;
//check the latest arrival time for each request;
for *each request k in this route* **do**
    if *the* es_{n+1}^Γ *exceeds the latest arrival time of request k* **then**
        return false;
//check the maximum ride time for each request;
for *each request k in this route* **do**
    generate a sub-route starting from k of this route and set the initial time window for k being $[es_k^0, l_k]$;
    for *each request j in the sub route* **do**
        for *each uncertainty level of* γ ($\gamma \le \Gamma$) **do**
            calculate $\overline{es_j^\gamma}$, $\overline{ls_j^\gamma}$, $\overline{dr_j^\gamma}$ by **Algorithm 4**;
    get the worst minimum route duration of the sub-route $(\overline{dr^\Gamma})$;
    if $\overline{dr^\Gamma} >$ *the maximum ride time of request k* **then**
        return false;
return true;

4.4. Neighborhood operators

Four removal operators and four repair operators are designed to obtain a neighborhood solution.

4.4.1. Removal operators

Random Removal. Requests can be randomly chosen and removed from the routes.

Worst Removal. The route cost may be quite high if requests are placed in undesirable positions. Therefore, it is necessary to identify and remove these requests. To begin with, we calculate the cost reduction for each request after it is removed. Then, the request with the biggest cost reduction is removed from the current solution. Finally, the previous two steps are repeated until the number of removed requests meets the requirement.

Route Removal. To begin with, a route is chosen randomly and removed. Then, if the number of requests on the route is less than the number that needs to be removed, the worst removal is used to remove the remaining number of requests.

Shaw Removal. For the FMRITH, we define the closeness of two requests i and j as $CL(i,j) = t_{ij} + \vartheta \times |e_i - e_j| + \sigma \times |l_i - l_j| + \mu \times |d_i - d_j|$, where t_{ij} is the travel time from request i to request j, ϑ, σ, and μ are the penalty parameters of deviation for the earliest service start time, the latest service start time, and the latest arrival time at the intercity transportation hub. We set the value of 0.5 to these penalty parameters.

Algorithm 4: Robust service start time window computation

Input: current request, predecessor request, es^{γ}_{k-1}, ls^{γ}_{k-1}, dr^{γ}_{k-1}, $es^{\gamma-1}_{k-1}$, $ls^{\gamma-1}_{k-1}$, $dr^{\gamma-1}_{k-1}$ of predecessor request, time window, travel time(TT) and uncertainty travel time ($\widetilde{TT}$) between two requests

Output: the earliest and latest service start time for request k under γ ($es^{\gamma}_k, ls^{\gamma}_k$), and route duration from depot to current request under γ (dr^{γ}_k)

//assume that the travel time from the predecessor to k is deterministic;

calculate $es^{\gamma}_k,\ ls^{\gamma}_k,\ dr^{\gamma}_k : es^{\gamma}_k \leftarrow es^{\gamma}_{k-1} + TT,\ ls^{\gamma}_k \leftarrow ls^{\gamma}_{k-1} + TT,\ dr^{\gamma}_k \leftarrow dr^{\gamma}_{k-1} + TT$;

if $es^{\gamma}_k > l_k$ **then**

  return false **//violate the service time window of k**;

else if $ls^{\gamma}_k < e_k$ **then**

  $dr^{\gamma}_k \leftarrow dr^{\gamma}_k + (e_k - ls^{\gamma}_k),\ es^{\gamma}_k, ls^{\gamma}_k \leftarrow e_k$ **//wait for picking up the request**;

else

  $es^{\gamma}_k \leftarrow max\{es^{\gamma}_k, e_k\},\ ls^{\gamma}_k \leftarrow min\{ls^{\gamma}_k, l_k\}$;

//assume that the travel time from the predecessor to k is uncertain;

calculate $\widetilde{es^{\gamma}_k},\ \widetilde{ls^{\gamma}_k},\ \widetilde{dr^{\gamma}_k} : \widetilde{es^{\gamma}_k} \leftarrow es^{\gamma-1}_{k-1} + \widetilde{TT},\ \widetilde{ls^{\gamma}_k} \leftarrow ls^{\gamma-1}_{k-1} + \widetilde{TT},\ \widetilde{dr^{\gamma}_k} \leftarrow dr^{\gamma-1}_{k-1} + \widetilde{TT}$;

if $\widetilde{es^{\gamma}_k} > l_k$ **then**

  return false **//violate the service time window of k**;

else if $\widetilde{ls^{\gamma}_k} < e_k$ **then**

  $\widetilde{dr^{\gamma}_k} \leftarrow \widetilde{dr^{\gamma}_k} + (e_k - \widetilde{ls^{\gamma}_k}),\ \widetilde{es^{\gamma}_k}, \widetilde{ls^{\gamma}_k} \leftarrow e_k$ **//wait for picking up the request**;

else

  $\widetilde{es^{\gamma}_k} \leftarrow max\{\widetilde{es^{\gamma}_k},\ e_k\},\ \widetilde{ls^{\gamma}_k} \leftarrow min\{\widetilde{ls^{\gamma}_k},\ l_k\}$;

return $es^{\gamma}_k \leftarrow max\left\{es^{\gamma}_k,\ \widetilde{es^{\gamma}_k}\right\},\ ls^{\gamma}_k \leftarrow min\left\{ls^{\gamma}_k, \widetilde{ls^{\gamma}_k}\right\},\ and\ dr^{\gamma}_k \leftarrow max\left\{dr^{\gamma}_k, \widetilde{dr^{\gamma}_k}\right\}$;

4.4.2. Repair operators

Random Repair. Each request is randomly inserted into a feasible insertion position, and if the request cannot be inserted into any positions, an empty route will be generated to insert this request.

Best Repair. For each request, it will be inserted into the lowest-cost position if feasible insertion positions exist in the available routes. Otherwise, a new empty route will be generated for the insertion.

Greedy Repair. Regarding this operator, the insertion costs per request inserted in each available position are calculated. Then, the lowest-cost request-position pair is selected, which means this request will be inserted into this position. If the lowest insertion cost equals M (an extremely large number; note that if a route is infeasible, we set the route cost to M.), which means that there is no feasible position to insert this request, and a new empty route will be generated to insert this request.

Regret Repair. The *regret repair* operator (Pisinger and Ropke, 2007) is a variant of the *greedy repair* operator, which uses the look-ahead information to choose the request and position for insertion. We use ΔC^r_i to denote the minimum insert cost of inserting the request i into route r. Let ΔC^r_i be M, if route r will be infeasible after inserting i into it. Furtherly, we use $r(k)$ to denote the route which has the kth lowest cost for inserting request i. The regret value (RV) of request i can be evaluated by $RV(i) = \Delta C^{r(2)}_i - \Delta C^{r(1)}_i$, and request i with the maximum regret value should be inserted in this iteration, $i := arg\ max_i(\Delta C^{r(2)}_i - \Delta C^{r(1)}_i)$. The *regret repair* method can also be extended to regret-k repair operators, $i := arg\ max_i \sum_k (\Delta C^{r(k)}_i - \Delta C^{r(1)}_i)$. In our research, the value of k is set to 2 and 3, which can be randomly selected in each iteration.

4.5. Adaptive mechanism

The roulette-wheel procedure is employed to determine which destroy and repair operators are selected in each iteration of the ALNS. The selection is based on the weight of each operator. Each operator's weight is assigned based on its past performance (also known as "score") in previous iterations. Let w_i, which is initialized to 1, denote the weight of the destroy (or repair) operator i. Given k destroy (or repair) operators, the probability of selecting operator i is $w_i / \sum_{i=1}^{k} w_i$. The operators' weights are adjusted via Eq. (25) after ϕ iterations (also known as "segment", see Pisinger and Ropke, 2007). During the iteration process of each segment, the scores of each operator are recorded. At the beginning of each segment, all scores are set to 0. Then, scores are increased by σ_1, σ_2, σ_3 depending on the new solution's quality (see Pisinger and Ropke, 2007 for the definition of σ_1, σ_2, σ_3).

$$w_i = \begin{cases} (1-R)w_i + R\ \frac{\Pi_i}{\Xi_i}, & \text{if } \Xi_i > 0 \\ w_i, & \text{if } \Xi_i = 0 \end{cases} \tag{25}$$

Where Π_i is the score of operator i recorded in the last segment. $R \in [0,1]$ is the reaction factor controlling the speed of the weight adjustment. Ξ_i records the number of times operator i is called during the last segment.

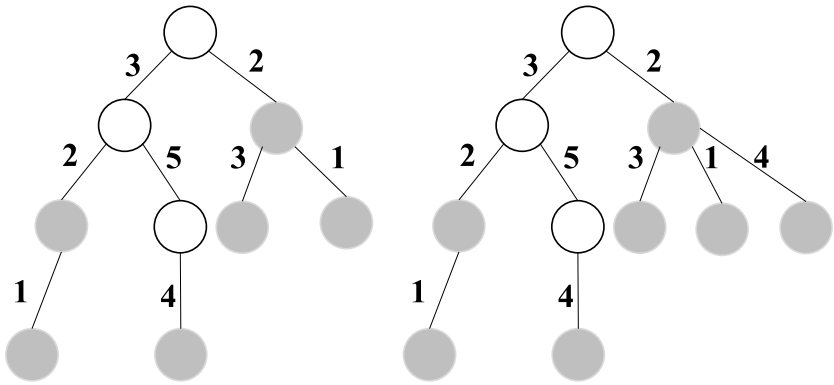


Fig. 3. An illustrative example of routes storing in trie.

4.6. Acceleration strategy

There are many identical routes during the process of the neighborhood search. Accordingly, the routes which have been checked can be preserved, and the information on these routes can be directly used if they reappear, which can efficiently speed up the algorithm's computation time since it avoids some repeated computations. The route information can be preserved by the *trie*, an ordered tree data structure also known as "digital tree" proposed by De La Briandais (1959), which has been widely applied in string retrievals, predictive text, spell-checking, and term indexing (Park et al., 2007). *Trie* has been utilized in time-consuming feasibility checks or calculations by Wei et al. (2015) and Zhang et al. (2021).

According to the characteristic of this problem, we utilize the *trie* to record requests' ID and total costs of each route. For any new route generated in the process of the neighborhood search, we can try to retrieve the route from the *trie* before we check its feasibility and calculate its cost. To reduce the memory space occupied by the *trie*, we can set the cost of an infeasible route to a sufficiently large number, so we can retrieve the feasibility of the route directly from its cost value. If the route can be retrieved from the *trie*, we can immediately get the cost and feasibility of the route. Otherwise, we need to check the feasibility, compute the cost of the route, and store this information in the *trie*. Fig. 3 illustrates the storage and retrieval of route information. The current *trie* preserves 6 routes, including 0-3-2-0, 0-3-2-1-0, 0-2-0, 0-2-3-0, 0-2-1-0 and 0-3-5-4-0. If we have two routes now, 0-3-2-0 and 0-2-4-0, respectively, the first route's cost and feasibility can be directly retrieved, but the second route does not exist in the *trie*, which needs to compute related information and store it into the *trie*.

Note that the trie can effectively reduce the overall running time of the algorithm only when the path length is limited by the capacity of the car, for instance, 4–7 requests per car. If the path is too long, it may decrease the read and storage speeds of the trie and thus cannot ensure that the algorithm's runtime is reduced.

5. Lower-bound and upper-bound solutions for the FMRITH problem

To evaluate the solution quality of the tailored ALNS algorithm, information about the lower-bound and upper-bound solutions for the FMRITH problem can be utilized. In this section, we introduce the column-generation algorithm and a greedy algorithm to get a tight lower-bound and an upper-bound solution for this problem, respectively.

5.1. A column-generation algorithm for the lower-bound solution

We first reformulate the MILP model as a set-partitioning model via the Danzig–Wolfe decomposition technique since the linear relaxation of the set-partitioning model can provide a tighter bound than the original MILP model (see, for instance, Li and Jia, 2019, Liu et al., 2021). Then, the column generation algorithm can be leveraged to solve the set-partitioning model and obtain the lower-bound information about the original problem. In this model, each decision variable (column) denotes a ridesharing scheme consisting of the served requests, the serving sequence, and the latest arrival time.

Let Θ represent the set of ridesharing schemes. The original problem becomes selecting the best combination of a subset from Θ. The decision variable is defined as δ_θ such that $\delta_\theta = 1$ if ridesharing scheme $\theta \in \Theta$ is employed as part of the solution, and $\delta_\theta = 0$ otherwise. Moreover, we define binary parameter χ_i^θ such that $\chi_i^\theta = 1$ if request i is served by the ridesharing scheme θ, and 0 otherwise. The cost of each ridesharing scheme can be denoted as Ψ_θ. With these notations, the first-mile ridesharing to intercity transportation hubs problem can be reformulated as the following set-partitioning model:

$$[\mathbf{MP}] \sum_{\theta \in \Theta} \Psi_\theta \delta_\theta \tag{26}$$

subject to:

$$\sum_{\theta\in\Theta} \chi_i^\theta \delta_\theta = 1, \forall i \in N_1 \tag{27}$$

$$\delta_\theta \in \{0,1\}, \forall \theta \in \Theta \tag{28}$$

The objective function (26) minimizes the total operating cost for rendering ridesharing services. Constraint (27) ensures that each request is served exactly by one ridesharing scheme. Constraint (28) defines the range of variable δ_θ. Note that the number of complete columns in MP grows exponentially with the scale of the problem. Enumerating all columns is computationally intractable in most cases, and the common practice is repeatedly solving the restricted master problem (RMP) by focusing on a subset $\overline{\Theta} \in \Theta$ (Song et al., 2017). In addition, decision variables δ_θ are relaxed to be continuous variables ($\delta_\theta \geq 0, \forall \theta \in \Theta$). To ensure RMP is always feasible, we introduce an artificial decision variable $\mu_i \geq 0$ for each request.

$$[\mathbf{RMP}] \sum_{\theta\in\overline{\Theta}} \Psi_\theta \delta_\theta + \sum_{i\in N_1} M\mu_i \tag{29}$$

subject to:

$$\sum_{\theta\in\overline{\Theta}} \chi_i^\theta \delta_\theta + \mu_i = 1, \forall i \in N_1 \tag{30}$$

$$\delta_\theta \geq 0, \forall \theta \in \overline{\Theta} \tag{31}$$

Let π_i denote dual variable associated with each request i of constraint (30) in RMP. Then the reduced cost corresponding to each ridesharing scheme θ is:

$$\overline{\Psi_\theta} = \Psi_\theta - \sum_{i\in N_1} \pi_i \chi_i^\theta = \sum_{i\in N}\sum_{j\in N} cd_{ij}x_{ij} - \sum_{i\in N_1} \pi_i \chi_i^\theta \tag{32}$$

Then, the pricing sub-problem (PSP) can be formulated as:

$$[\mathbf{PSP}]\ min\ \overline{\Psi_\theta} = \sum_{i\in N}\sum_{j\in N} cd_{ij}x_{ij} - \sum_{i\in N_1} \pi_i \chi_i^\theta \tag{33}$$

subject to:

$$\sum_{j\in N\cap j\neq\{i\cup 0\}} x_{ij} \leq 1, \forall i \in N_1 \tag{34}$$

$$(6)-(7), (9)-(24) \tag{35}$$

The objective function (33) aims to identify a ridesharing scheme with the lowest reduced cost. Constraint (34) denotes that each request is served at most once by the ridesharing scheme.

The sub-problem is a variant of the elementary shortest path problem with resource constraints (ESPPRC). Employing the labeling algorithm to obtain the optimal solution for the ESPPRC and its variants is a common practice (see for example Feillet et al., 2004, Irnich and Desaulniers, 2005). The process of the CG algorithm is repeatedly solving the RMP and passing values of dual variables to the PSP. Then, solving the PSP to obtain columns with negative reduced costs. The CG algorithm halts when the PSP cannot find a column with a negative objective value, which implies that we have identified the optimal solution for the RMP and a tight lower bound for the MILP model.

5.2. A greedy algorithm for an upper-bound solution

We introduce a greedy algorithm based on the solution's characteristic of the FMRITH problem, in which cars typically visit the farthest requests first since their maximum ride time is longer than that of other requests that are closer to the intercity transit hub. The detailed procedure for the greedy algorithm is summarized as follows, and Fig. 4 shows an illustrative example.

Step 1: Select the k farthest requests as the first request node of k routes;

Step 2: Pick out the n closest nodes for each farthest request, and obtain k × n new routes by extending these nodes to the ends of the k routes, respectively;

Step 3: For each new feasible route, select n new requests closest to the last node of the route and extend them to the route, which can get n new partial routes. Repeat this process until no more feasible extensions can be found for each route;

Step 4: Obtain the set of feasible routes starting from the k farthest requests. Then, we choose the route with the most riders from the set as an "optimal" route and mark requests traversed by the "optimal" route as serviced;

Step 5: Repeat steps 1–4 for the remaining requests until each request is serviced, and utilize these "optimal" routes as a solution;

Step 6: Output the "optimal" route set.

Note that we can run different combinations of k and n and select the solution with the lowest cost as the upper-bound solution for the FMRITH problem. In this paper, we set the maximum values of k and n to be $max\{5, \lceil \frac{|requests|}{2} \rceil\}$ and 3, respectively.

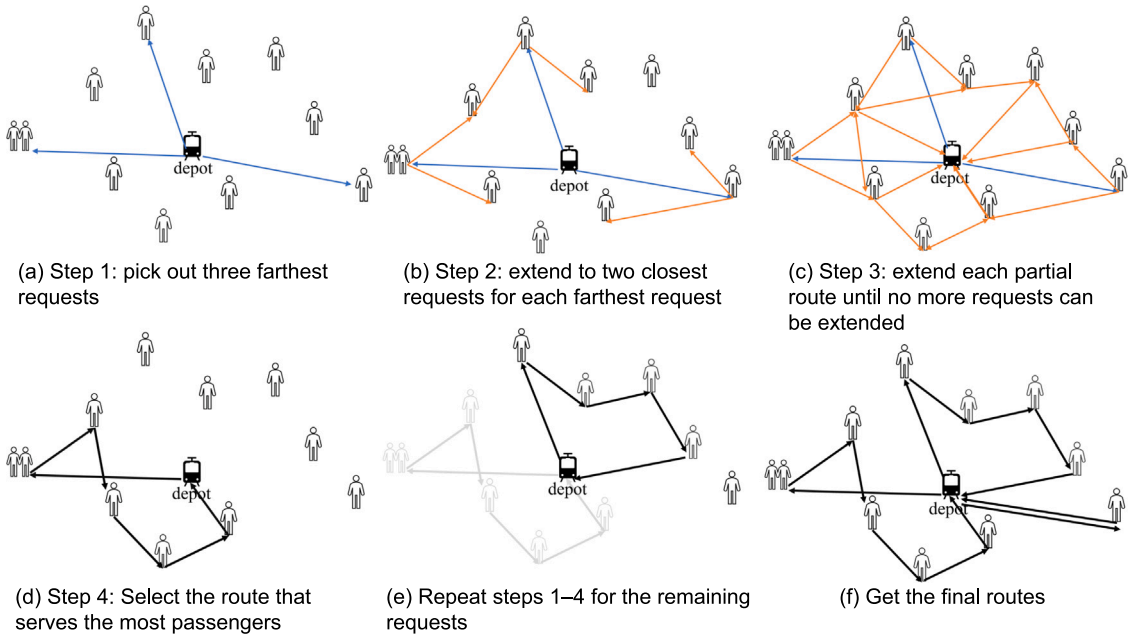


Fig. 4. An illustrative example for the greedy algorithm with k = 3 and n = 2.

Fig. 5. Locations of some sample requests heading for the SSRS.

6. Case study

In this section, we conduct computational experiments to validate the performance of the proposed model and algorithms. The Shanghai South Railway Station (SSRS) is selected as the intercity transport hub, and passenger demand is randomly generated within 15 km of the hub station (for instance, see Fig. 5, which consists of 1 transportation hub and 25 request nodes). The algorithms are coded in C++, and CPLEX 12.7 is used to solve the MILP, MP, and RMP models. All computational experiments are conducted on a PC with 8 GB of RAM and a 2.3 GHz CPU.

Table 3
Values of some basic parameters for the case study.

Parameters	Value
Transportation cost per kilometer	0.2
Number of passenger seats in the car	7
Number of pieces of luggage that can be placed on one seat	2
Number of pieces of luggage that can be placed in the trunk	2
Budget of travel time uncertainty (Γ)	3
Deviation coefficients of travel time (α)	0.15/0.3/0.5
Limit of computation time (seconds) by CPLEX	7200
Maximum number of iterations for the ALNS	10000
Number of iterations for each segment (ϕ)	50
Maximum number of iterations for the NIIts	500
Ratio of requests removed according to the NIIts (ζ_{rate})	0.0002
Minimum ratio of requests removed (ζ_{min})	0.1
Maximum ratio of requests removed (ζ_{max})	0.3
Initial temperature (T)	100
linear cooling rate ($\mathscr{R}$)	0.01

Table 4
Values of several parameters for the ALNS algorithm.

Parameters	Range	Best value
R	[0,1; 0.2; 0.5]	0.1
σ_1	[10; 20; 30]	30
σ_2	[10; 20; 30]	10
σ_3	[5; 10; 20; 30]	5

6.1. Parameters setting and instances generation

The values of some basic parameters are shown in Table 3. The proposed ALNS algorithm has several parameters (R, σ_1, σ_2, σ_3) to be tuned, and we conduct a sensitivity analysis to determine the best combination for these parameters. To avoid the exponential explosion of these combinations, we first give each parameter a set of values according to experiences in the literature (for instance, Luo et al., 2016) and some preliminary experiments. Then, we calibrate the values of parameters R, σ_1, σ_2, σ_3 by running 108 ($3 \times 3 \times 3 \times 4$) different combinations for the instance with 50 requests. The best combination of parameters is determined based on the objective value, and the best values for the weight-adjustment parameters are shown in Table 4.

Regarding the instance generation, we first randomly generate coordinates, the number of riders, and the number of pieces of large luggage for each request. We assume that the number of riders is 1 or 2, with probabilities of 80% and 20%, respectively, and the number of pieces of large luggage is 0, 1, or 2, with probabilities of 50%, 40%, and 10% (Li et al., 2021). The distance between all nodes (consisting of request nodes and the depot at the SSRS) is extracted via the Python package OSMnx (Boeing, 2017). Then, we randomly generate the latest arrival time (within the same time slot, for instance, 8:00 to 8:15), the maximum travel time, which is between 1.5 and 2 times the direct ride time to the hub station, and the earliest pickup time for each request. The latest pickup time is derived by adding 30 min to the earliest pickup time. Data can be found on this webpage.

6.2. Computational performance of the ALNS algorithm

To assess the computational performance of the ALNS, we compare the results with three methods: directly solving the proposed model by the CPLEX solver, the CG algorithm, and the greedy algorithm. As mentioned in Section 5, the CG algorithm and the greedy algorithm can provide a tighter lower bound and an upper bound, respectively. In addition, the CG algorithm can identify a near-optimal solution by solving the MP model when the RMP model finds all the negative columns, which is also known as the CG heuristic (CGH) algorithm. Table 5 reports the computational results for instances with the request number ranging from 5 to 80 and the uncertainty level of 0, 1, and 3. The deviation coefficient of travel time is 0.5. Note that the ALNS runs 10 times for each instance.

As can be seen, CPLEX can only solve small-scale instances with less than 20 requests to optimality. The ALNS algorithm outperforms the CPLEX as it finds either the same or better solutions. The CG algorithm finds optimal solutions for small-scale instances with less than 15 requests and better solutions than CPLEX for large-scale instances with more than 30 requests. Although the greedy algorithm has the shortest computation time, its solutions are unstable. The greedy algorithm cannot find the optimal solutions for some small-scale instances, such as instance "R10Γ0". The maximum gap (Gap2) between the greedy solution and the lower bound solution is up to 16.93%, while for the ALNS algorithm, the gap (Gap3) is less than 8.14%. In addition, the standard deviation of objective values for the ALNS algorithm is less than 1.76. Although the CG algorithm finds better solutions for some larger-scale instances than the ALNS and the greedy algorithm, it takes a much longer time to determine the solution, and the ALNS shows higher performance on the computation efficiency as instances with less than 50 requests can be solved within one minute, and most of the large-scale instances can be solved within three minutes. Moreover, we find that the uncertainty level has

Table 5
Computational performance of the ALNS.

Instances	CPLEX solver			CG algorithm				Greedy algorithm			ALNS algorithm				
	Obj ($)	Time (s)	Gap (%)	Obj^{CG} ($)	Obj^{CGH} ($)	Time (s)	Gap1 (%)	Obj^{GRD} ($)	Time (s)	Gap2 (%)	Obj^{BEST} ($)	Obj^{AVR} ($)	STD ($)	Time (s)	Gap3 (%)
R5Γ0	13.43	0.1	0.00	13.43	13.43	0.2	0.00	13.43	0.1	0.00	13.43	13.43	0.00	5.0	0.00
R5Γ1	14.99	0.1	0.00	14.99	14.99	0.2	0.00	14.99	0.1	0.00	14.99	14.99	0.00	6.2	0.00
R5Γ3	16.90	0.1	0.00	16.90	16.90	0.2	0.00	16.90	0.1	0.00	16.90	16.90	0.00	6.9	0.00
R10Γ0	29.86	0.1	0.00	28.80	29.86	0.6	3.68	30.02	0.1	4.24	29.86	29.86	0.00	6.5	3.68
R10Γ1	30.07	0.1	0.00	30.07	30.07	0.5	0.00	30.07	0.1	0.00	30.07	30.07	0.00	8.6	0.00
R10Γ3	33.27	0.1	0.00	33.27	33.27	0.2	0.00	33.27	0.1	0.00	33.27	33.27	0.00	10.1	0.00
R15Γ0	32.61	3.2	0.00	32.61	32.61	1.4	0.00	33.75	0.1	3.50	32.62	32.62	0.00	10.2	0.03
R15Γ1	37.90	0.4	0.00	37.90	37.90	0.7	0.00	39.66	0.1	4.64	37.90	37.90	0.00	11.1	0.00
R15Γ3	41.51	0.2	0.00	41.51	41.51	0.5	0.00	41.52	0.2	0.02	41.51	41.51	0.00	15.8	0.00
R20Γ0	37.75	950.9	0.00	37.00	38.27	13.7	3.43	38.78	0.5	4.81	37.75	37.96	0.42	13.2	2.59
R20Γ1	41.76	90.6	0.00	41.09	42.12	4.5	2.51	42.47	0.3	3.36	41.76	41.76	0.00	15.5	1.63
R20Γ3	46.41	16.9	0.00	46.41	46.41	2.8	0.00	46.42	0.3	0.02	46.41	46.41	0.00	22.5	0.00
R30Γ0	55.77	7200.0	21.35	48.64	50.49	120.4	3.80	53.26	1.5	9.50	50.07	50.93	0.54	21.6	4.71
R30Γ1	59.00	7200.0	12.77	52.25	52.72	31.2	0.90	60.12	1.0	15.06	52.72	52.72	0.00	27.4	0.90
R30Γ3	66.91	7200.0	16.33	62.16	66.01	85.1	6.19	66.21	0.8	6.52	65.06	65.13	0.10	36.5	4.78
R40Γ0	67.88	7200.0	25.57	61.84	64.14	356.3	3.72	68.80	4.6	11.25	63.46	64.48	0.71	39.9	4.27
R40Γ1	70.58	7200.0	25.61	64.27	69.30	94.2	7.83	69.78	3.9	8.57	66.98	67.69	0.57	43.8	5.32
R40Γ3	80.75	7200.0	33.07	76.95	79.87	65.0	3.79	84.02	2.5	9.19	78.67	79.45	0.28	59.2	3.25
R50Γ0	85.13	7200.0	34.61	74.60	76.35	1779.1	2.35	85.35	14.1	14.41	76.40	77.69	1.22	55.9	4.14
R50Γ1	87.80	7200.0	32.81	80.62	84.90	582.1	5.31	90.63	11.4	12.42	82.24	83.86	1.02	62.9	4.02
R50Γ3	99.08	7200.0	39.75	91.77	95.45	186.5	4.01	103.46	6.1	12.74	94.88	95.40	0.85	103.2	3.96
R60Γ0	104.43	7200.0	44.93	85.71	89.17	5926.7	4.04	94.26	29.5	9.98	88.54	91.34	1.62	93.2	6.57
R60Γ1	118.72	7200.0	48.73	90.89	96.12	1137.7	5.75	105.26	16.8	15.81	93.11	93.80	0.57	90.7	3.20
R60Γ3	120.83	7200.0	48.08	104.92	108.49	496.6	3.40	111.96	11.8	6.71	107.74	107.85	0.22	124.7	2.79
R70Γ0	129.07	7200.0	53.81	94.65	97.89	5605.0	3.42	110.67	66.2	16.93	97.37	100.05	1.61	113.8	5.71
R70Γ1	132.52	7200.0	53.96	99.57	101.43	1214.5	1.87	114.75	28.7	15.25	100.90	102.25	1.76	125.1	2.69
R70Γ3	144.26	7200.0	57.20	121.61	127.99	692.2	5.25	128.51	21.4	5.67	123.81	125.63	1.14	199.8	3.31
R80Γ0	147.15	7200.0	58.34	109.19	115.47	20450.9	5.75	125.91	105.7	15.31	115.49	118.08	1.48	160.2	8.14
R80Γ1	152.94	7200.0	54.32	113.85	121.37	4022.6	6.61	131.75	46.6	15.72	118.28	120.17	1.47	179.8	5.55
R80Γ3	163.58	7200.0	58.98	135.11	137.60	2040.7	1.84	144.14	33.8	6.68	136.42	137.15	0.52	249.2	1.51

Note: "Obj", "Obj^{CG}", "Obj^{CGH}", and "Obj^{GRD}" denote the objective values identified by the CPLEX solver, the CG algorithm, the CG heuristic algorithm, and the greedy algorithm. "Obj^{BEST}" and "Obj^{AVR}" are the best and average objective values determined by running the ALNS algorithm 10 times, respectively. Gap refers to the difference between the upper and lower bounds determined by the CPLEX solver; Gap1 = (Obj^{CGH} - Obj^{CG})/ Obj^{CG} ×100%; Gap2 = (Obj^{GRD} - Obj^{CG})/ Obj^{CG} ×100%; Gap3 = (Obj^{AVR} - Obj^{CG})/ Obj^{CG} × 100%. "STD" shows the standard deviation between the 10 objective values (found by the ALNS algorithm) of each instance.

a significant impact on the computation time of the ALNS, and the higher the uncertainty level, the longer the computation time. As the uncertainty level increases, it causes an increase in the computation of time-related information (for instance, the ride time) for each route, which leads to an increase in the overall running time. For CG and greedy algorithms, the lower the uncertainty level, the larger the search space for solutions, and the longer the computation time.

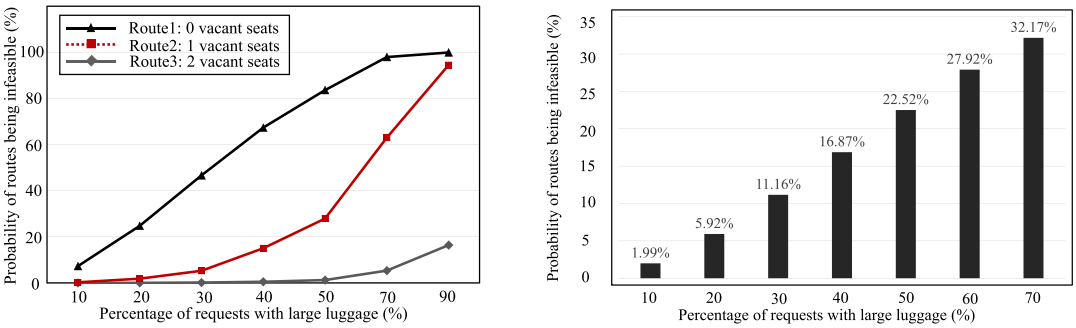
6.3. Analysis of the large luggage impacts on route feasibility

To demonstrate the significance of considering large luggage in route planning, we use the Monte Carlo simulation method to evaluate the feasibility of routes without considering large luggage. Two simulation cases are analyzed. For the first case, three routes with 0, 1, and 2 seats unused are generated, and we assume that requests served by the route only have one rider. For the second case, we obtain 118 routes from 5 instances, each with 100 requests. To check the feasibility of these routes considering luggage constraints, we randomly generate the number of pieces of luggage for requests according to the parameters setting of Section 6.1. Then, 10,000 simulations are performed for each route to identify the feasible probability. Detailed results are shown in Fig. 6.

It can be seen that, when the proportion of requests with large luggage is small, such as less than 10%, the impact of large luggage on the route's feasibility is insignificant, and the large luggage can be ignored in route planning. As a result, large luggage can be ignored in the ridesharing scheme design for the FM ridesharing to bus or metro stations. The proportion of passengers carrying large luggage is relatively high at intercity railway stations and airports, so the impact of luggage on route feasibility must be considered. As shown in Fig. 6(b), the probability of route infeasibility increases as the proportion of requests with large luggage increases. In particular, when the proportion reaches 50 or more, the probability of infeasibility is greater than 22.52%.

6.4. Analysis of routes' robustness

This section leverages the Monte Carlo simulation method to evaluate the robustness of the solutions. To begin with, three request density cases (low, medium, and high) are generated, which correspond to 25, 50, and 100 requests, respectively. Then, for each



(a) route feasibility with different vacant seats (b) average feasibility of all routes in the solution

Fig. 6. Probability of infeasible routes without considering large luggage.

Table 6
Risk and PoR for cases with 25, 50, and 100 requests.

Requests	Γ	15% deviation		30% deviation		50% deviation	
		Risk (%)	PoR (%)	Risk (%)	PoR (%)	Risk (%)	PoR (%)
25	0	36.29	0.00	48.43	0.00	49.89	0.00
	1	5.13	8.44	14.94	9.49	28.67	10.45
	2	0.00	9.49	0.56	9.83	1.33	31.21
	3	0.00	9.49	0.00	15.11	0.00	35.20
50	0	24.97	0.00	35.11	0.00	51.25	0.00
	1	0.00	0.76	12.83	2.88	25.11	6.59
	2	0.00	0.76	0.59	5.15	5.33	10.05
	3	0.00	1.98	0.00	9.43	0.00	19.62
100	0	33.64	0.00	74.95	0.00	92.14	0.00
	1	0.27	1.66	6.79	3.07	10.97	6.56
	2	0.27	2.36	2.14	3.50	3.72	10.08
	3	0.00	3.53	0.00	3.93	0.00	13.93

case, the near-optimal solution is determined under different uncertainty levels: 0, 1, 2, and 3, and different deviation coefficients of travel time: 15%, 30%, and 50%, respectively. We randomly generate the travel times for each route segment in the half-interval $[t_{ij}, t_{ij}+\widehat{t_{ij}}]$, to evaluate whether the route is feasible. Furthermore, the simulation process is performed 10,000 times for each route to estimate the probability of the route being infeasible (i.e., Risk). Finally, we evaluate the price of robustness (PoR, the increase in the route cost) under uncertain travel time in contrast to deterministic travel time.

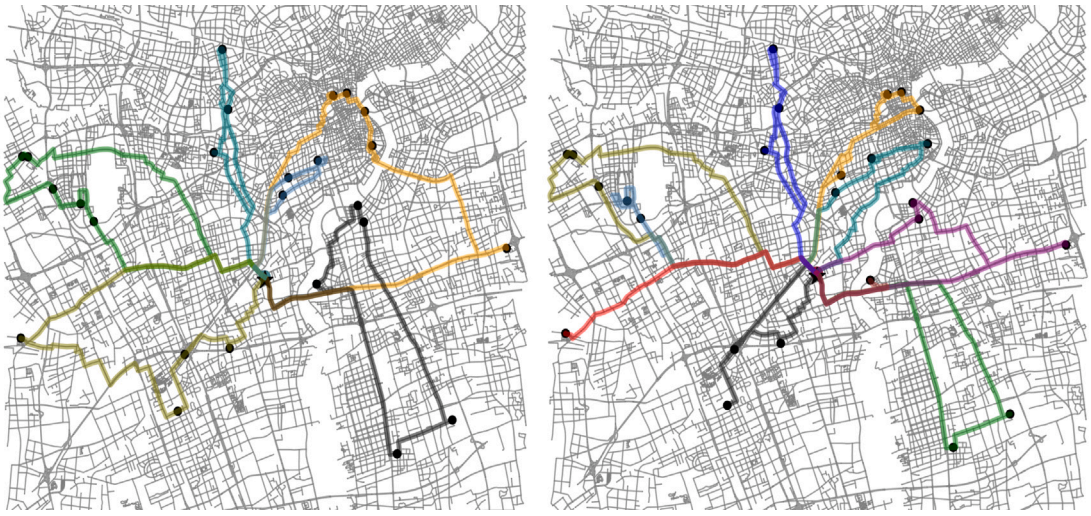
Regarding the route feasibility, the results in Table 6 show that the solution's robustness is poor if the uncertainty of travel time is not considered, which is the case of $\Gamma = 0$. In particular, the probability of routes being infeasible exceeds 90% for the case with 100 requests when the deviation coefficient of travel time is 50%. The robustness of the route increases as the level of uncertainty rises. 2 budgets of travel time uncertainty make the robustness of routes excellent, and the infeasible probability of the route is less than 10%. To ensure that the solution can well withstand the uncertainty of travel time, 3 budgets are indispensable.

Regarding the route cost, it varies significantly for the case of low request density with the increase in uncertain budgets, especially in instances with fewer requests. The increase in the uncertain budgets with fewer requests can make the route vary considerably, thus increasing the cost of the solution. This conclusion is illustrated in Fig. 7, which shows the ridesharing solution under 25 requests, with (a) representing the solution without considering uncertainty and (b) showing the solution under 3 budgets of travel time uncertainty. The average number of requests served per car is more for (a), and the solution in (a) used only 6 cars, while the solution in (b) used 10 cars, which means more no-load mileages (occurring before picking up the first request), so the cost difference between the two solutions is significant.

However, for cases with a medium and high density of requests, the budgets of travel time uncertainty have a smaller impact on the cost, and the impact becomes smaller as the density of requests increases. Since the requests in these two cases are more numerous and more evenly distributed, the variation of the routes after considering the uncertainty of travel times is not significant, thus making the cost variation small.

6.5. Average service cost of each request

This subsection conducts experiments to identify the impact of service radius and the number of requests on the average service cost for each request. The service radius ranges from 10 to 15 km, and the number of requests ranges from 20 to 100. We randomly generate 5 instances for each combination of radius-request and take their average objective value as the service cost for each



(a) without considering uncertainty (b) considering uncertainty

Fig. 7. Routes of whether to consider the uncertainty of travel time.

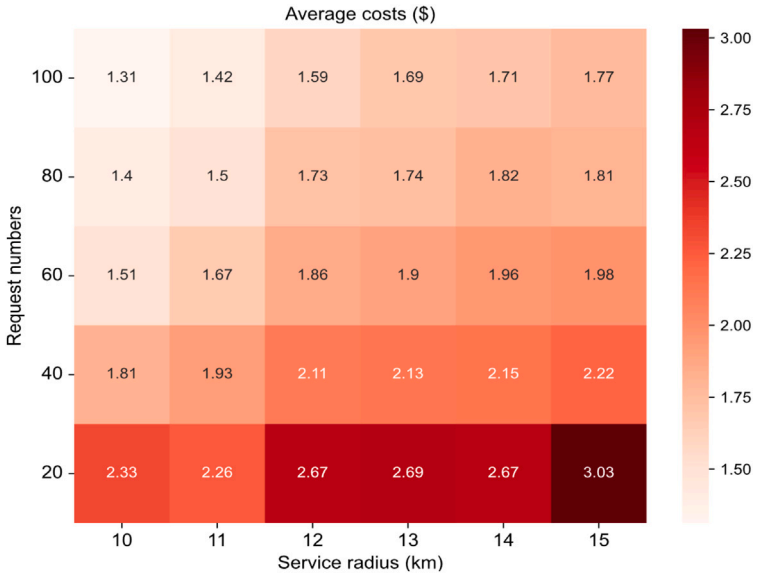


Fig. 8. Average serving cost of each request with different combinations of service radius and number of requests.

combination. The average service cost of each combination is reported in Fig. 8. It can be seen that, in a given service area with a fixed service radius, more requests generate lower average service costs, while fewer requests incur higher costs. For instance, as the number of requests increased from 20 to 100 within a 10 km service radius, the average service cost decreased by 40%. Similarly, with the same number of requests, the larger area they distribute over, the more cost it generates. The average cost for serving each request with different radius-request combinations is helpful for service providers to determine the best service radius and fare level.

6.6. Impact of car types on total travel distances

Two commonly used car types, 4-seat and 7-seat(number of seats for passengers) cars, are selected to analyze the impact of the car type on total travel distances, which may be useful for estimating total operating costs. We directly use the instances in Section 6.2 and set the number of requests from 10 to 80 and the level of travel time uncertainty to 3. Fig. 9 shows the total travel

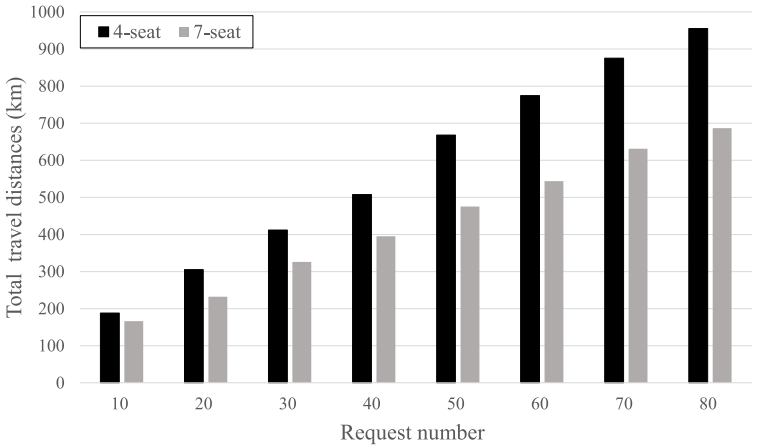


Fig. 9. Total travel distances for different car types with different request numbers.

distances for 4-seat and 7-seat cars with the different numbers of requests. Compared to 4-seat passenger cars, total travel distances are lower for 7-seat cars, and the distances saved increase as requests increase. Therefore, when the transportation cost per kilometer is similar for 7-seat and 4-seat passenger cars, it is recommended that the service provider should leverage 7-seat cars to render first-mile ridesharing services, especially when the request number is high.

7. Conclusion

In the quest to provide economical, reliable, and flexible first-mile travel services for passengers heading to intercity transportation hubs, this study proposes a ridesharing approach to this problem. A mixed-integer linear programming model is developed with the objective of minimizing the cost of the ridesharing service operator while considering riders' requirements on the latest arrival times and maximum ride times, large luggage, and travel time uncertainty. A robust adaptive large neighborhood search algorithm with an acceleration strategy, a greedy algorithm, and a column generation algorithm are designed to solve real-world problems.

The case study shows that (1) the method proposed in this paper can effectively solve the first-mile ridesharing problem. (2) Ridesharing service operators must consider large luggage when dispatching cars for the first-mile problem to intercity transportation hubs to ensure the capacity limits of cars are respected. (3) Considering the travel time uncertainty effectively improves the robustness of the routes and does not make the cost increase significantly for the scenarios with a medium and high density of requests. (4) The average cost of providing first-mile ridesharing services for each request is strongly associated with the request density, which should be considered by ridesharing service providers to determine the best service radius and the fare level. (5) When the transportation cost per kilometer is similar for 7- and 4-seat passenger cars, it is recommended that the service provider should leverage 7-seat cars to render first-mile ridesharing services, especially when the request number is high.

This study assumes that passenger cars depart from the depot in the intercity transportation hub and are homogeneous, which may contribute to a lot of no-load mileages before picking up the first passenger and low-load mileages on some routes. In future research, we will therefore investigate the problem of first-mile ridesharing to intercity hubs with heterogeneous passenger cars, which may enhance the flexibility of the services and reduce the overall operating cost for service providers. The integration of first- and last-mile services also deserves further research, which could reduce the empty mileages occurring in first-mile services.

CRediT authorship contribution statement

Ping He: Conceptualization, Methodology, Software, Validation, Writing – original draft. **Jian Gang Jin:** Conceptualization, Methodology, Writing – review & editing, Supervision, Funding acquisition. **Frederik Schulte:** Conceptualization, Writing – review & editing. **Martin Trépanier:** Conceptualization, Writing – review & editing.

Acknowledgments

This work is supported by the National Natural Science Foundation of China [Grant 72061127003, 72122014, 71771149, 71831008, 52088102], and National Key Research and Development Program of China [Grant 2020AAA0107600]. The first author also appreciates the support of the China Scholarship Council [Grant 202106230253]. The authors would like to thank the three anonymous referees for their valuable comments.

References

Bertsimas, D., Sim, M., 2004. The price of robustness. Ope. Res. 52 (1), 35–53.

Bian, Z., Liu, X., 2017. Planning the ridesharing route for the first-mile service linking to railway passenger transportation. In: 2017 Joint Rail Conference. American Society of Mechanical Engineers Digital Collection, pp. 1–11.

Bian, Z., Liu, X., 2019a. Mechanism design for first-mile ridesharing based on personalized requirements Part I: Theoretical analysis in generalized scenarios. Transp. Res. B 120, 147–171.

Bian, Z., Liu, X., 2019b. Mechanism design for first-mile ridesharing based on personalized requirements Part II: Solution algorithm for large-scale problems. Transp. Res. B 120, 172–192.

Bian, Z., Liu, X., Bai, Y., 2020. Mechanism design for on-demand first-mile ridesharing. Transp. Res. B 138, 77–117.

Boarnet, M.G., Giuliano, G., Hou, Y., Shin, E.J., 2017. First/last mile transit access as an equity planning issue. Transp. Res. A 103, 296–310.

Boeing, G., 2017. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. Comput. Environ. Urban Syst. 65, 126–139.

Campbell, K.B., Brakewood, C., 2017. Sharing riders: How bikesharing impacts bus ridership in New York City. Transp. Res. A 100, 264–282.

Chen, X., Lin, L., 2016. The integration of air and rail technologies: Shanghai's Hongqiao integrated transport hub. J. Urban Tech. 2, 23–46.

Chen, P.W., Nie, Y.M., 2017. Connecting E-hailing to mass transit platform: Analysis of relative spatial position. Transp. Res. C 77, 444–461.

Chen, S., Wang, H., Meng, Q., 2020. Solving the first-mile ridesharing problem using autonomous vehicles. Comput.-Aided Civ. Infrastruct. Eng. 35 (1), 45–60.

De La Briandais, R., 1959. File searching using variable length keys. In: Western Joint Computer Conference. ACM, pp. 295–298.

Dessouky, M., Rahimi, M., Weidner, M., 2003. Jointly optimizing cost, service, and environmental performance in demand-responsive transit scheduling. Transp. Res. D 8 (6), 433–465.

El-Assi, W., Salah Mahmoud, M., Nurul Habib, K., 2017. Effects of built environment and weather on bike sharing demand: A station level analysis of commercial bike sharing in Toronto. Transportation 44 (3), 589–613.

El-Geneidy, A., Grimsrud, M., Wasfi, R., Tétreault, P., Surprenant-Legault, J., 2014. New evidence on walking distances to transit stops: Identifying redundancies and gaps using variable service areas. Transportation 41 (1), 193–210.

Fang, Z.M., Lv, W., Jiang, L., Xu, Q., Song, W.G., 2015. Observation, simulation and optimization of the movement of passengers with baggage in railway station. Internat. J. Modern Phys. C 26, 1550124.

Feillet, D., Dejax, P., Gendreau, M., Gueguen, C., 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks 44 (3), 216–229.

Hochmair, H.H., 2015. Assessment of bicycle service areas around transit stations. Int. J. Sustain. Transp. 9 (1), 15–29.

Huang, Y., Kockelman, K.M., Garikapati, V., 2022. Shared automated vehicle fleet operations for First-Mile Last-Mile transit connections with dynamic pooling. Comput. Environ. Urban Syst. 92 (2), 101730.

Huang, Y., Kockelman, K.M., Garikapati, V., Zhu, L., Young, S., 2021. Use of shared automated vehicles for first-mile last-mile service: Micro-simulation of rail-transit connections in Austin, Texas. Transp. Res. Record 2675 (2), 135–149.

Irnich, S., Desaulniers, G., 2005. Shortest path problems with resource constraints. Column Generation. (Springer, New York) 33–65.

Jiang, G., Lam, S.K., Ning, F., He, P., Xie, J., 2020. Peak-hour vehicle routing for first-mile transportation: Problem formulation and algorithms. IEEE Trans. Intell. Transp. Syst. 21 (8), 3308–3321.

Kim, K., 2018. Investigation on the effects of weather and calendar events on bike-sharing according to the trip patterns of bike rentals of stations. J. Transp. Geogr. 66, 309–320.

Kumar, P., Khani, A., 2021. An algorithm for integrating peer-to-peer ridesharing and schedule-based transit system for first mile/last mile access. Transp. Res. C 122, 102891.

Lee, J., Choi, K., Leem, Y., 2016. Bicycle-based transit-oriented development as an alternative to overcome the criticisms of the conventional transit-oriented development. Int. J. Sustain. Transp. 10 (10), 975–984.

Li, S., Jia, S., 2019. The seaport traffic scheduling problem: Formulations and a column-row generation algorithm. Transp. Res. Part B 128, 158–184.

Li, C., Yang, R., Chen, L., Tang, T., 2021. A boarding model for heterogeneous passengers on the platform of high-speed railway station. Simu. Model. Prac. Theo. 106, 102188.

Liu, Z., Jia, X., Cheng, W., 2012. Solving the last mile problem: Ensure the success of public bicycle system in Beijing. Procedia - Soc. Behav. Sci. 43, 73–78.

Liu, B., Li, Z., Sheng, D., Wang, Y., 2021. Integrated planning of berth allocation and vessel sequencing in a seaport with one-way navigation channel. Transp. Res. Part B 143, 23–47.

Luo, Z., Qin, H., Zhang, D., Lim, A., 2016. Adaptive large neighborhood search heuristics for the vehicle routing problem with stochastic demands and weight-related cost. Transp. Res. E 85, 69–89.

Ma, T.Y., Rasulkhani, S., Chow, J.Y.J., Klein, S., 2019. A dynamic ridesharing dispatch and idle vehicle repositioning strategy with integrated transit transfers. Transp. Res. E 128, 417–442.

Mahéo, A., Kilby, P., Van Hentenryck, P., 2019. Benders decomposition for the design of a hub and shuttle public transit system. Transp. Sci. 53 (1), 77–88.

Masson, R., Ropke, S., Lehuédé, F., Péton, O., 2014. A branch-and-cut-and-price approach for the pickup and delivery problem with shuttle routes. European J. Oper. Res. 236 (3), 849–862.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., 1953. Equation of state calculations by fast computing machines. J. Chem. Phys. 21 (6), 1087–1092.

Munari, P., Moreno, A., De La Vega, J., Alem, D., Gondzio, J., Morabito, R., 2019. The robust vehicle routing problem with time windows: Compact formulation and branch-price-and-cut method. Transp. Sci. 53, 1043–1066.

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. Inform. Sci. 570, 241–261.

Park, G., Hwang, H.-k., Nicodeme, P., Szpankowski, W., 2007. Profiles of tries. SIAM J. Comput. 4957 (5), 1–11.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Comput. Oper. Res. 34 (8), 2403–2435.

Quadrifoglio, L., Dessouky, M.M., Ordóñez, F., 2008. A simulation study of demand responsive transit system design. Transp. Res. A 42 (4), 718–737.

Shaheen, S., Chan, N., 2016. Mobility and the sharing economy: Potential to facilitate the first- and last-mile public transit connections. Built Environ. 42 (4), 573–588.

Shen, Y., Zhang, H., Zhao, J., 2018. Integrating shared autonomous vehicle in public transportation system: A supply-side simulation of the first-mile service in Singapore. Transp. Res. A 113, 125–136.

Song, Y., Zhang, J., Liang, Z., C., Y., 2017. An exact algorithm for the container drayage problem under a separation mode. Transp. Res. E 106, 231–254.

Stiglic, M., Agatz, N., Savelsbergh, M., Gradisar, M., 2018. Enhancing urban mobility: Integrating ride-sharing and public transit. Comput. Oper. Res. 90, 12–21.

Wang, J., Huang, J., Jing, Y., 2020. Competition between high-speed trains and air travel in China: From a spatial to spatiotemporal perspective. Transp. Res. A 133 (1), 62–78.

Wei, L., Zhang, Z., Zhang, D., Lim, A., 2015. A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints. European J. Oper. Res. 243 (3), 798–814.

Winters, M., Davidson, G., Kao, D., Teschke, K., 2011. Motivators and deterrents of bicycling: Comparing influences on decisions to ride. Transportation 38 (1), 153–168.
Xiong, J., He, Z., Guan, W., Ran, B., 2015. Optimal timetable development for community shuttle network with metro stations. Transp. Res. C 60, 540–565.
Zhang, L., Liu, Z., Lan, Y., Ke, F., Yao, B., Yu, B., 2022. Routing optimization of shared autonomous electric vehicles under uncertain travel time and uncertain service time. Transp. Res. E 157, 102548.
Zhang, Y., Zhang, Z., Lim, A., Sim, M., 2021. Robust data-driven vehicle routing with time windows. Oper. Res. 69 (2), 469–485.
Zhou, Z., Yang, M., Cheng, L., Yuan, Y., 2022. Do passengers feel convenient when they transfer at the transportation hub? Trav. Behav. and Soci. 29, 65–77.
Zuo, T., Wei, H., Chen, N., Zhang, C., 2020. First-and-last mile solution via bicycling to improving transit accessibility and advancing transportation equity. Cities 99, 102614.
Zuo, T., Wei, H., Rohne, A., 2018. Determining transit service coverage by non-motorized accessibility to transit: Case study of applying GPS data in Cincinnati metropolitan area. J. Transp. Geogr. 67, 1–11.