Online Scheduling and Route Planning for Shared Buses in Urban Traffic Networks

Zhaolong Ning, *Senior Member, IEEE*, Shouming Sun, MengChu Zhou, *Fellow, IEEE*, Xiping Hu, *Member, IEEE*, Xiaojie Wang, Lei Guo, Bin Hu, *Senior Member, IEEE*, and Ricky Y. K. Kwok, *Fellow, IEEE*

***Abstract*— It is critical to reduce the operating cost of shared buses for bus companies and improve the user experience of passengers. However, existing studies focus on either bus scheduling or route planning, which cannot accomplish the above mentioned goals concurrently. In this paper, we construct a joint bus scheduling and route planning framework to maximize the number of passengers, minimize the total length of routes and the number of required buses, as well as guarantee good user experience of passengers. First, we establish a system model based on a real-world scenario and formulate a multi-objective combinational optimization problem. Then, based on the extracted traffic topology of urban traffic networks and the generated candidate line set, we propose an offline algorithm to cope with the similar passenger flow distributions, e.g., morning or evening peak of every day. In order to cope with dynamic real-time passenger flows, an online algorithm is designed. Experiments are carried out based on real-word scenarios. The results show that the proposed algorithms can greatly reduce the operating cost of bus companies and guarantee good user experience based on real-world scheduling data in comparison with several existing methods.**

***Index Terms*— Shared bus, last mile, bus scheduling, route planning, multi-objective optimization.**

Manuscript received February 18, 2020; revised July 1, 2020, August 28, 2020, and October 27, 2020; accepted November 2, 2020. Date of publication March 24, 2021; date of current version March 29, 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFE0206800, in part by the National Natural Science Foundation of China under Grant 61971084 and Grant 62001073, in part by the Fundamental Research Funds for the Central Universities under Grant DUT19JC18, in part by the National Natural Science Foundation of Chongqing under Grant cstc2019jcyj-msxmX0208, in part by the Open Research Fund of National Mobile Communications Research Laboratory, Southeast University, under Grant 2020D05, and in part by the Open Research Project of the State Key Laboratory of Industrial Control Technology, Zhejiang University, China, under Grant ICT20070. The Associate Editor for this article was A. Che. *(Corresponding authors: Xiping Hu; Xiaojie Wang.)*

Zhaolong Ning is with the School of Software, Dalian University of Technology, Dalian 116620, China, also with the National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China, also with the School of Communications and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China, and also with the State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310007, China (e-mail: zhaolongning@dlut.edu.cn).

Shouming Sun is with the School of Software, Dalian University of Technology, Dalian 116620, China (e-mail: shoumingsun@outlook.com).

MengChu Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07103 USA (e-mail: zhou@njit.edu).

Xiping Hu and Bin Hu are with the School of Information Science and Engineering, Lanzhou University, Lanzhou 730000, China (e-mail: huxp@lzu.edu.cn; bh@lzu.edu.cn).

Xiaojie Wang and Lei Guo are with the School of Communications and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: xiaojie.kara.wang@ieee.org; guolei@cqupt.edu).

Ricky Y. K. Kwok is with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong (e-mail: ricky.kwok@hku.hk).

Digital Object Identifier 10.1109/TITS.2020.3036396

I. Introduction

ENERGY shortage is becoming a serious problem for countries all over the world in recent years. According to statistics, totally about 142.86 billion gallons of motor gasoline were consumed in the United States in 2018 [1]. Fossil fuels are mainly consumed by vehicles, among which buses consume more energy than private cars and taxies. As ride-sharing receives increasing attention recently [2], shared bus emerges as a novel kind of transportation means. It integrates the online car-hailing services and traditional public transportation to provide an inexpensive door-to-door travel experience, and can provide customized bus services for passengers with the same fragmented trips by dynamically scheduling buses and planning routes to allocate transportation resources.

Compared with the traditional mode of taxi, shared buses have the following characteristics:

- Inexpensiveness: The ticket prices of shared buses are much lower than that of taxies, which can significantly improve the travel experience of passengers.
- High road resource utilization: Shared buses can effectively save road resources compared with taxies, since shared buses have larger seating capacities than the latter, which can relieve the issues caused by the shortage of road resources and traffic congestion.
- Environment-friendliness: Since shared buses are generally powered by electric or natural gas, they can reduce the emissions of greenhouse gas enormously compared with taxies to protect the environment.

Compared with traditional public transportation [3], [4], shared buses have the following characteristics:

- Various application scenarios: The operating scenarios of shared buses contain the urban transport hub scene (e.g., railway stations and airports), commuting scene, and the "last mile" scene. It is worth noting that the size of shared bus application scenario is smaller than that of public transportation.

- Customized services: Shared buses emerge due to the use of Internet technology and public/personal transportation needs. Passengers can book tickets online, so that passenger demands can be obtained in a timely and accurate way. By dynamic bus scheduling and route planning, shared buses can provide customized services for passengers to meet their travel demands and improve their experience.
- Flexibility: Shared buses can generate dynamic schedules based on specific passenger demands to reduce operating cost and improve user experience other than a fixed schedule in traditional public transportation.
- High vehicle resource utilization: Shared buses can meet the travel demands of passengers better by providing customized services than conventional public transportation. Therefore, they have higher load factor and can effectively improve the utilization of vehicle resources.

Bus companies prefer to use fewer shared buses with pre-designed routes to transport as many passengers as possible for the sake of profits. Passengers intend to spend less time on their waiting and traveling. However, these needs are in contradiction, and it is urgent to make a good trade-off between them. Although many researchers have proposed either bus scheduling or route planning strategies [5]–[8], the unilateral research cannot resolve the above issue. The challenges of a Bus Scheduling and Route Planning (BSRP) problem can be summarized as follows:

- Multi-objective: BSRP is a multi-objective combinational optimization problem, where both the operating cost of bus companies and user experience of passengers require to be considered. The former generally includes the number of required buses and the length of routes. The latter mainly contains the waiting and traveling time of passengers as well as the congestion degree in buses.
- Temporal-spatial: BSRP has the temporal-spatial characteristic, including the temporal information of arrival, waiting and traveling time of passengers, departure, running and arrival time of buses, as well as the spatial location information of passengers, buses and stations.
- High coupling: BSRP is rather complicated due to the high coupling characters among buses, e.g., the high correlation degree, especially when buses are dispatched on multiple lines. Furthermore, the high coupling among buses is time-dependent.
- Random arrival: In most real-world scenarios, passenger arrival time is random, and all passenger information cannot be obtained beforehand. Therefore, feasible solutions are necessary to cope with the dynamic real-time passenger flows.

In this paper, we construct a *Joint bUs Scheduling and rouTe planning framework* (named JUST), with the purposes of maximizing the number of passengers picked up by the shared bus in each trip, minimizing the total route length of and number of required buses, as well as guaranteeing good user experience of passengers. We first model BSRP as a multi-objective combinational optimization problem. Then, we extract a traffic topology existing in big cities. After that, both offline and online schemes are put forward to solve the formulated problem. The main contributions of this paper are:

1) We formulate a multi-objective combinational optimization model for BSRP by considering the operating cost of shared bus companies and user experience of passengers. Specifically, we guarantee the waiting time of each passenger is no more than a defined threshold, which is different from previous work that merely guarantees the average waiting time of passengers.
2) We design a two-phase method to extract a traffic topology suitable for the running of shared buses by considering passenger flows, passenger waiting time, road length and cyclization. We demonstrate that such a traffic topology exists in metropolis through real-world data analysis. Based on the extracted traffic topology, an improved local search algorithm is presented to solve the relaxed BSRP problem to generate a candidate line set.
3) We propose an offline solution, i.e., *TrIp Generation and Assignment algoRithm* (TIGAR) algorithm, to effectively dispatch shared buses to cope with passenger flows with similar distributions, and decompose BSRP into two subproblems, i.e., trip generation and trip assignment. We further design an online algorithm, i.e., *Arrival Data-based Passenger assignmenT algorithm* (ADPT), to schedule shared buses in real time to cope with dynamic and random passenger flows. In addition, carry out extensive experiments based on the real-world data set of shared buses in Shanghai (China) to demonstrate the effectiveness of our algorithms. The results show that both TIGAR and ADPT can guarantee personal good user experience, and have advantages over several existing methods.

Section II reviews the related work. Section III illustrates BSRP and its optimization model. Offline and online solutions are specified in Section IV. Section V shows performance evaluation results, followed by the conclusion in Section VI.

II. Related Work

A. Bus Scheduling

Recent efforts have been made to investigate the bus scheduling problem. Zuo *et al.* [9] leveraged an improved multi-objective genetic algorithm to optimize the number of vehicles and drivers for a vehicle scheduling problem. Boyer *et al.* [10] proposed a variable neighborhood search scheme to rapidly provide solutions for vehicle and crew scheduling problems. A human-machine interactive bus transit scheduling system was developed in [11] to reduce vehicle fleet size and the number of duties, by applying the deficit function theory. The multiple depot vehicle scheduling problem was solved in [12] by a local search algorithm based on pruning and deepening techniques to reduce the cost of vehicles. A two-phase heuristic was proposed to reduce the operating cost [13]. The first phase schedules vehicles based on a column-generation heuristic, and the possible timetable can be found based on a mixed integer program in the second phase. The above studies schedule buses based on a fixed

schedule. However, they are difficult to reduce the operating cost of companies and guarantee the user experience of passengers simultaneously.

In order to provide a comfortable user experience by bus scheduling, Zhang *et al.* proposed a two-step model of coarse prediction and calibration by leveraging an extended kalman filter to predict the passenger flow in real time [14]. Wang *et al.* [5] developed a data-driven scheme for real-time bus scheduling optimization. First, time-dependent traffic and customer demands are inferred from the travel data of passengers, and then a model is formulated to minimize the average waiting time of passengers by scheduling the departure time of each bus. A graphical human-machine interactive technique was leveraged to solve the bus scheduling problem [15]. Specifically, multiple strategies are used to adjust trips to minimize the passenger travel time changes and the number of required buses to reduce the operating cost. Kumar *et al.* [16] dynamically dispatched buses by leveraging a headway-based approach to minimize the number of trips to maximize the benefit of operators, by considering the capacity of buses and the waiting time of passengers.

Different from previous studies, our solutions can effectively reduce the operating cost of shared bus companies, meet the travel demands and guarantee the user experience of passengers and achieve the goal of low-carbon, environmental protection as well as high utilization of bus resources under the background of shared economy. The operating cost is reduced by minimizing the number of required buses and total route length as well as maximizing the number of passengers picked up in each trip. The user experience is guaranteed by the constraints of load factor and the waiting time of each passenger. In addition, existing studies generally focus on the average waiting time of passengers, but we focus on the personal user experience (i.e., waiting time of each passenger).

B. Route Planning

The route planning problem on a point-of-interest network is solved by leveraging a two-module framework [17]. One module plans a preliminary route with pruning and caching strategies, and the other refines the route by both Dijkstra algorithm and post-processing mechanism. Wang and Lu [18] proposed a memetic algorithm with competition to solve a vehicle routing problem by reducing the total distance. Chen *et al.* developed a bidirectional probability-based spreading algorithm, in which the next station could be selected with a certain probability by considering passenger flow, to plan routes for night buses [19]. A probabilistic route planning algorithm was proposed in [20] by extending an Ant Colony Optimization (ACO) algorithm to mobile crowdsensing environment to search the shortest route. Zhu *et al.* in [21] proposed a route recommendation scheme by considering personal preferences, proper visiting time and transition time, and leveraged the High Order Singular Value Decomposition (HOSVD) to decompose a three tensor, i.e., user, location and time to infer personal preferences. The ride-sharing routing problem is solved by leveraging a dynamic-programming based algorithm [22]. Reinforcement learning based selection method is presented in [23] to generate routes, aiming to avoid potential crime risk as well as obtain short route distance. In [24], a double rewarded value iteration network is proposed to learn an experienced driver's routing decisions for route planning, where a long-short-term memory network is trained to model the knowledge of traffic trends.

Fig. 1. The "last mile" scenario.

Different from previous studies, we not only jointly consider bus scheduling and route planning problems, but also focus on three objectives, i.e., optimizing the number of required buses, route length and the number of passengers picked up in each trip, to reduce the operating cost for bus providers and guarantee good user experience. Although [25] and [26] have some similarity with our work, several significant differences exist: First, we consider dynamic passenger flows with temporal-spatial correlation; Second, personal waiting time is guaranteed instead of optimizing the average waiting time of passengers; Third, we design an offline algorithm to decompose BSRP into two subproblems and solve them in polynomial time. Finally, we present an online algorithm to deal with random arrival passengers. Note that iterative heuristic algorithms in [25], [26] are inefficient to solve BSRP.

III. System Model and Problem Formulation

We consider a real-world "last mile" scenario of shared buses, in which they pick up passengers from residential areas to a subway station, and there is no passenger getting off the shared bus in intermediate stations. Specifically, as shown in Fig. 1, s_1 and s_5 are two starting stations and s_9 is the subway station. Shared buses start from s_1 and s_5 to meet the demands of all passengers waiting at stations $s_1 - s_8$ during the operating time. We intend to minimize the operating cost of bus providers and guarantee the personal user experience of passengers.

A complete directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is used to represent the traffic network, where $\mathcal{V}$ is a set of shared bus stations, expressed by $\mathcal{V} = \{1, 2, \ldots, V\}$, and $\mathcal{E}$ is a set of edges, denoted by $\mathcal{E} = \{e\langle u, v\rangle, u, v \in \mathcal{V} \wedge u \neq v\}$. To meet the travel demands of all passengers, a set of shared bus trips is represented by $\mathcal{S} = \{1, \ldots, S\}$. The route of trip $s \in \mathcal{S}$ is represented by $\Phi(s) = \langle \mathfrak{s}, \ldots, J\rangle$, where $\mathfrak{s}$ is a starting station

and J is the terminal. The departure time of trip s is $\Phi_{\mathfrak{d}}(s)$. There are multiple starting stations in the scenario. The formal definitions of $\Phi_D(s)$, $\Phi_P(s)$ and $\Phi_A(s)$ are given as follows:

Definition 1: $\Phi_D(s)$ is the length of route $\Phi(s)$, equaling to the sum of the distances among adjacent stations, i.e.,

$$\Phi_D(s) = \sum_{u=1}^{V}\sum_{v=1}^{V} e\left\langle u, v\right\rangle_d \delta_{uv}(s), \tag{1}$$

where u and v are stations u and v, and $e\left\langle u, v\right\rangle_d$ is the distance between stations u and v. Symbol $\delta_{uv}(s)$ is a binary variable, equaling to 1 if station v is the next station of station u in route $\Phi(s)$. Otherwise, it is 0.

Definition 2: $\Phi_P(s)$ is the set of passengers picked up by the shared bus on route $\Phi(s)$, equaling to the whole passengers picked up in all stations on route $\Phi(s)$, i.e.,

$$\Phi_P(s) = \bigcup_{v\in\Phi(s)} \Phi_P(s, v), \tag{2}$$

where $\Phi_P(s, v)$ is the set of passengers picked up by the shared bus in station v of route $\Phi(s)$.

Definition 3: $\Phi_A(s)$ is the arrival time of the bus to terminal J in route $\Phi(s)$, i.e.,

$$\Phi_A(s) = \Phi_{\mathfrak{d}}(s) + \sum_{u=1}^{V}\sum_{v=1}^{V} e\left\langle u, v\right\rangle_{\mathfrak{r}}(t)\delta_{uv}(s), \tag{3}$$

where $\Phi_{\mathfrak{d}}(s)$ is the departure time from the starting point of route $\Phi(s)$, and $e\left\langle u, v\right\rangle_{\mathfrak{r}}(t)$ is the running time from stations u to v at t^{th} time slot. It is worth noting that for different time, $e\left\langle u, v\right\rangle_{\mathfrak{r}}(t)$ differs. Similar to [27], we count the running time between two stations per ξ minutes from the start of shared bus operating time based on the passenger data set.

The formulated problem has three objectives. To reduce the operating cost, we need to minimize the number of required buses to meet the travel demands of all passengers, i.e.,

$$\min\ N_B, \tag{4a}$$
$$\text{s.t. } N_B \geq 1, N_B \in \mathbb{N}, \tag{4b}$$

where N_B denotes the number of required buses. (4b) indicates that there is at least one bus to cover all stations.

Our second objective is to minimize the total route length of all involved trips to reduce the operating cost, i.e.,

$$\min\ \Phi_D = \sum_{s=1}^{S}\Phi_D(s), \tag{5a}$$
$$\text{s.t. } \delta_{uv}(s) \in \{0, 1\}, \forall s \in \mathcal{S}, \tag{5b}$$
$$\sum_{v=1}^{V}\delta_{uv}(s) = 1, \forall s \in \mathcal{S}, u \in \Phi(s) \wedge u \neq J, \tag{5c}$$
$$\sum_{u=1}^{V}\delta_{uv}(s) = 1, \forall s \in \mathcal{S}, v \in \Phi(s) \wedge v \neq \mathfrak{s}, \tag{5d}$$
$$\Phi_D(s) \leq D^{\wedge}, \forall s \in \mathcal{S}, \Phi_D(s) \in \mathbb{R}, \tag{5e}$$
$$z_u - z_v + z_J\delta_{uv}(s) \leq z_J - 1, \quad \forall s \in \mathcal{S}, \forall u, v \in \Phi(s), u \neq v, \tag{5f}$$

where (5b) indicates that $\delta_{uv}(s)$ is a binary variable. Since the route of trip $\Phi(s)$ is selected from the candidate line set, for each trip, $\delta_{uv}(s)$ is known if the candidate line set is determined. Constraints (5c) and (5d) are leveraged to restrict that each station in route $\Phi(s)$ has merely one outgoing link and one ingoing link, respectively. Herein, $u \in \Phi(s)$ and $v \in \Phi(s)$ can guarantee that stations u and v are in route $\Phi(s)$ of trip s, respectively. Since we do not consider the route from terminals to starting stations, there is no outgoing link in terminal (i.e., $u \neq J$) and there is no ingoing link in starting stations (i.e., $v \neq \mathfrak{s}$), where J and $\mathfrak{s}$ are terminals and starting stations, respectively. These two constraints can be found in [28]. In (5f), z_u is position of station u in the route $\Phi(s)$ and $z_{\mathfrak{s}} = 1$, and it can eliminate the sub-tour in all trips [29], [30], where $\forall u, v \in \Phi(s)$ can guarantee stations u and v are in route $\Phi(s)$ of trip s.

In addition, the number of passengers picked up by the bus on each route is related to the operating cost. Without loss of generality, we assume that the set of passengers during the operating time of shared buses is $\mathcal{P}$, and $P = |\mathcal{P}|$, which can be calculated by $P = \sum_t M_t$. Herein, t is the t^{th} time slot, and M_t is the passenger demands in the t^{th} time slot. Increasing the number of passengers picked up by the shared bus on each route $\Phi(s)$ can reduce the operating cost by decreasing the number of required trips as well as shared buses. Hence, we intend to maximize the number of passengers picked up in each trip, i.e.,

$$\max\ |\Phi_P(s)| = \sum_{v\in\Phi(s)} |\Phi_P(s, v)|, \tag{6a}$$
$$\text{s.t. } L_f(s) \leq L_f^{\wedge}, \quad \forall s \in \mathcal{S}, L_f(s) \in [0, +\infty), \tag{6b}$$

where $|\Phi_P(s, v)|$ is the number of passengers that are picked up at station v in trip s, which is related to the arrival time of passengers and the bus. $L_f(s)$ is the load factor of the shared bus in trip s, defined as the ratio of the number of passengers picked up by the shared bus to the number of seats, i.e., $L_f(s) = \frac{|\Phi_P(s)|}{\lambda}$, where λ is the capacity of shared buses. This value should be as large as possible to improve the resource utilization of buses, but it cannot exceed threshold $L_f^{\wedge}$ to avoid congestion in the shared buses.

User experience of passengers can be impacted by three factors, i.e., traveling time, waiting time, and congestion degree in a bus. Since passenger traveling time is positively correlated with route length. Our second objective (5a) can guarantee the traveling time of passengers, while the congestion degree can be satisfied by (6b). Since the waiting time is significant for user experience, the below constraint is needed to guarantee the waiting time of each passenger:

$$b_{\mathfrak{a}}(s, v) - p_{\mathfrak{a}}(s, v) \leq T_w^{\wedge}, \quad \forall s \in \mathcal{S}, v \in \Phi(s), p \in \Phi_P(s, v). \tag{7}$$

Herein, $b_{\mathfrak{a}}(s, v)$ and $p_{\mathfrak{a}}(s, v)$ are the arrival time of shared bus b and passenger p to station v of trip s, respectively. $T_w^{\wedge}$ is the upper bound of waiting time for each passenger, denoting the tolerance of passengers for waiting time. The waiting time of each passenger is no more than the defined threshold to

provide customized services. The defined threshold can be adjusted for different user experience (e.g., a smaller one can bring better user experience for passengers), and it can reduce the transport resources by increasing the departure interval of shared buses. In here, we comprehensively consider all passengers and manage their waiting time accurately (i.e., the time interval between the arrival time of shard buses and that of passengers to the station). This constraint brings a higher complexity to the problem, since there is no limitation for $p_{\mathfrak{a}}(s,v)$ and the arrival time distribution of passengers, which means the arrival time of passengers is random, i.e., the uncertainty of passengers' arrival.

For trip s, we aggregate the variables of $\Phi_D(s)$ and $|\Phi_P(s)|$ based on the split and integration of the optimization objectives, i.e.,

$$\Phi_{DP}(s) = \frac{\Phi_D(s)}{|\Phi_P(s)|} = \frac{\sum_{u=1}^{V}\sum_{v=1}^{V} e\langle u,v\rangle_d\, \delta_{uv}(s)}{\sum_{v\in\Phi(s)}|\Phi_P(s,v)|}, \quad (8)$$

where $\Phi_D(s)$ and $|\Phi_P(s)|$ have the same decision variable (i.e., trip s) and definition domain, and $\Phi_{DP}(s)$ is defined as the average route length of each passenger in route $\Phi(s)$. It is obvious that $\Phi_{DP}(s)$ reaches the minimum value only if $\Phi_D(s)$ and $|\Phi_P(s)|$ reach the minimum and maximum values under the corresponding constraints, respectively. From the aspect of a trip, each one has an optimization objective, i.e., $\Phi_{DP}(s)$. Since all trips have the same optimization objective, and each trip is optimized independently (i.e., one trip is determined in each round by selecting routes from candidate lines and its start time is determined based on our optimization objective $\Phi_{DP}(s)$ as well as corresponding constraints), the optimization of each trip is equivalent to the optimization of all trips, i.e., $\Phi_{DP}(s), \forall s \in \mathcal{S}$. Finally, we model BSRP as a bi-objective combinational optimization problem, i.e.,

$$\begin{aligned} &\min\ (N_B, \Phi_{DP}(s)),\\ &\text{s.t. Constraints } (4b)(5b)(5c)(5d)(5e)(5f)(6b)(7). \end{aligned} \quad (9)$$

These two objectives cannot be combined into one by conventional linear weighted combination method [31], [32], since they have different units and it is difficult to determine the optimal weights for these two objectives for different companies with various requirements. The decision variable is the union of trips and their corresponding shared buses, i.e., (i,s), where i is the ID of the shared bus assigned to trip s, and trip s includes two attributes, i.e., its route $\Phi(s)$ and departure time $\Phi_{\mathfrak{d}}(s)$. For clarity, the main notations are summarized in Table I.

Proposition 1: *The formulated multi-objective BSRP problem is NP-hard.*

Proof: We simplify the formulated BSRP problem, i.e., only considering the optimization objective of minimizing the route length of one trip, while ignoring the time dimension, and regarding only one bus and one departure station exist in the traffic network. The formulated BSRP problem can be simplified to the problem that a bus starts from the departure station and visits each station only once to pick up passengers to the terminal with the minimal traveling distance, i.e., the optimization problem in (5a). Obviously, it is a traditional Traveling Salesman Problem (TSP) [33], which is a special case of our formulated BSRP problem. Since TSP has been proven to be NP-hard [34], our problem is also NP-hard. □

TABLE I
MAIN NOTATIONS

Notations	Description
$\mathcal{P}$	The set of passengers during the operating time of shared buses
$\mathcal{S}$	The set of trips for meeting the travel demands of all passengers during the operating time of shared buses
L_f	The load factor of shared buses
$\mathcal{G}(\mathcal{V},\mathcal{E})$	The complete directed graph representing the traffic network, whose sets of stations and edges are $\mathcal{V}$ and $\mathcal{E}$, respectively
$\Phi(s)$	The route of trip s
$\Phi_D(s), \Phi_A(s)$	The route length and arrival time of shared bus to the terminal station of route $\Phi(s)$, respectively
$\Phi_P(s)$	The number of passengers picked up by shared bus on route $\Phi(s)$
N_B	The number of required buses to meet the travel demands of all passengers
Φ_D	The total route length of all trips
$\Phi_{DP}(s)$	The average route length of each passenger in route $\Phi(s)$
$e\langle u,v\rangle_d$	The distance between stations u and v
$e\langle u,v\rangle_{\mathfrak{r}}(t)$	The running time of shared buses from station u to station v at t^{th} time slot
$b_a(s,v), p_a(s,v)$	The arrival time of shared bus b and passenger p to station v of trip s, respectively
T_f, P_f	The time and passenger network flow, respectively
$\mathcal{L}_{\varkappa}, \mathcal{L}_{\varepsilon}, \mathcal{L}_{\varsigma}$	The running, available and existing shared bus sets, respectively

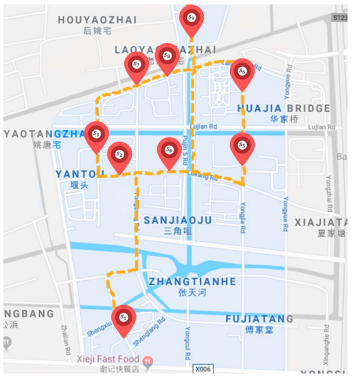


Fig. 2. The extracted traffic topology $\mathcal{G}(\mathcal{V},\mathcal{E}'')$.

IV. PROPOSED METHODS FOR BSRP

We first extract a traffic topology $\mathcal{G}(\mathcal{V},\mathcal{E}'')$, as shown in Fig. 2 from a traffic network $\mathcal{G}(\mathcal{V},\mathcal{E})$, where station s_1 and station s_5 are two starting points, and station s_9 is the subway station. Then, we design an improved local search algorithm to generate the candidate line set. Based on the candidate lines, both offline and online schemes are proposed to solve the formulated BSRP problem. It is worth noting that from real-world data analysis, we can observe that the extracted topology commonly exists in other cities, which demonstrates that our schemes are not limited to specific scenarios. The global algorithm for our JUST framework is shown in Algorithm 1.

Algorithm 1 Global Algorithm for JUST Framework

Input: $\mathcal{G}(\mathcal{V}, \mathcal{E})$, $\mathcal{H}$
Output: $\mathcal{D}$
Extract traffic topology $\mathcal{G}(\mathcal{V}, \mathcal{E}'')$ with Algorithm 2;
Generate candidate line set $\mathcal{S}$ with Algorithm 3;
if *the scenario has similar passenger flows* **then**
    TIGAR algorithm, containing Algorithm 4 and Algorithm 5, is preferred to solve the BSRP problem;
else
    Solve the BSRP problem with ADPT algorithm based on Algorithm 6;

Algorithm 2 Traffic Topology Extraction Algorithm

Input: $\mathcal{G}(\mathcal{V}, \mathcal{E})$, $\mathcal{H}$
Output: $\mathcal{G}(\mathcal{V}, \mathcal{E}'')$
Calculate T_f and P_f in $\mathcal{G}(\mathcal{V}, \mathcal{E})$ based on $\mathcal{H}$;
$T_f^{\wedge} \leftarrow$ Ford-Fulkerson(T_f, o, k);
$P_f^{\wedge} \leftarrow$ Ford-Fulkerson(P_f, o, k);
Determine $\check{T_f}$ and $\check{P_f}$;
for *each* $e \in \mathcal{G}(\mathcal{V}, \mathcal{E})$ **do**
    if $T_f < \check{T_f}$ *and* $P_f < \check{P_f}$ **then**
        $\mathcal{G}(\mathcal{V}, \mathcal{E}) \leftarrow \mathcal{G}(\mathcal{V}, \mathcal{E}) - e$;
Generate $\mathcal{G}(\mathcal{V}, \mathcal{E}')$;
$\mathcal{L}_{\varrho} \leftarrow$ Improved DFS($\mathcal{G}(\mathcal{V}, \mathcal{E}')$);
for $l \in \mathcal{L}_{\varrho}$ **do**
    Remove an edge from $\mathcal{G}(\mathcal{V}, \mathcal{E}')$ with Assumption 1;
Generate $\mathcal{G}(\mathcal{V}, \mathcal{E}'')$

A. Traffic Topology Extraction

Since some sections between two stations in the real traffic network are unsuitable for the running of shared buses, a two-phase algorithm, named *Traffic Topology Extraction algorithm*, is developed to extract a new traffic topology by considering passenger flows, passenger waiting time, route length and cyclization. Given the real traffic network $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and order data set $\mathcal{H}$, the corresponding pseudo-code is shown in Algorithm 2.

For the first stage, the real traffic network is represented by a complete directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. From the perspectives of shared bus companies and passengers, we need to consider the road length between two stations, passenger flows and passenger waiting time when extracting a traffic topology. In order to evaluate the quality of a section based on the above three factors, we define the time network flow T_f and passenger network flow P_f in $\mathcal{G}(\mathcal{V}, \mathcal{E})$. For edge $e\langle u, v\rangle$, the value of time network flow can be calculated by:

$$T_f = \frac{\gamma \overline{\mathfrak{T}_u} + \zeta \mathfrak{T}_u^{\wedge}}{e\langle u, v\rangle_d}, \tag{10}$$

where $\overline{\mathfrak{T}_u}$ and $\mathfrak{T}_u^{\wedge}$ are the average and maximum waiting time of passengers boarding at station u each day in order data set $\mathcal{H}$. Variables γ and ζ represent the importance of the average and maximum waiting time, respectively, determined by the Entropy method [35] based on $\mathcal{H}$. We can see that the larger the value of T_f is, the section tends to have a shorter length and the passengers in station u tend to have a longer waiting time, indicating edge $e\langle u, v\rangle$ is more essential for the traveling experience of passengers. For edge $e\langle u, v\rangle$, the value of passenger network flow can be calculated by:

$$P_f = \frac{\overline{\mathfrak{N}_u}}{e\langle u, v\rangle_d}, \tag{11}$$

where $\overline{\mathfrak{N}_u}$ is the average number of passengers boarding at station u. It is obvious that a larger P_f can reduce more operating cost for shared bus companies, and edge $e\langle u, v\rangle$ becomes more significant. We calculate the time network flow and passenger network flow of each edge in $\mathcal{G}(\mathcal{V}, \mathcal{E})$, and delete the edges whose T_f and P_f are smaller than $\check{T_f}$ and $\check{P_f}$, respectively. $\check{T_f}$ and $\check{P_f}$ are determined based on the maximum time network flow $T_f^{\wedge}$ and maximum passenger network flow $P_f^{\wedge}$ calculated by leveraging the Ford-Fulkerson algorithm [36], where the parameters o and k in the Ford-Fulkerson algorithm denote the source point and sink point, respectively. After the first stage, topology $\mathcal{G}(\mathcal{V}, \mathcal{E}')$ can be generated.

For the second stage, it is vital to search the loops in the traffic topology, so that they can be broken by deleting some relatively unimportant edges. This is because circular lines lead to the increase of the shared bus operating cost for companies, the waste of shared bus resource and the inferior traveling experience for passengers. An improved Deep-First Search (DFS) algorithm is designed to find all the loops in topology $\mathcal{G}(\mathcal{V}, \mathcal{E}')$. The details of the improved DFS algorithm are described as follows:

Based on the DFS algorithm, we save the unvisited edges and visit the subsequent nodes of the last visited node to find the obvious loops. Then, we check whether new loops exist or not by inserting the unvisited edges into the loops that have been found. Finally, all loops can be searched and are stored in $\mathcal{L}_{\varrho}$. In order to break the loops in $\mathcal{G}(\mathcal{V}, \mathcal{E}')$, the assumption is made, i.e.,

Assumption 1: *We assume that for a node in a graph, if it has a larger out-degree, it is more central. If the node has a larger in-degree, it is more authoritative. Removing the edge containing the node with the largest degree in a loop has the minimal impact on the graph.*

Assumption 1 indicates that deleting the edges that contain the nodes with the maximum degree can bring a relatively little influence on traffic topology $\mathcal{G}(\mathcal{V}, \mathcal{E}')$. Without loss of generality, considering node v' has the maximum degree in a loop, if it has the maximum out-degree, the edge that starts from v' is removed. Otherwise, delete the edge that ends at v'. It is worth noting that we only break the loops that do not contain starting stations, since the real operation scenario has multiple starting stations and the lines cross each other, leading to the loops that include starting stations exist in the traffic topology. Finally, the extracted traffic topology $\mathcal{G}(\mathcal{V}, \mathcal{E}'')$ suitable for the running of shared buses can be obtained after the second stage.

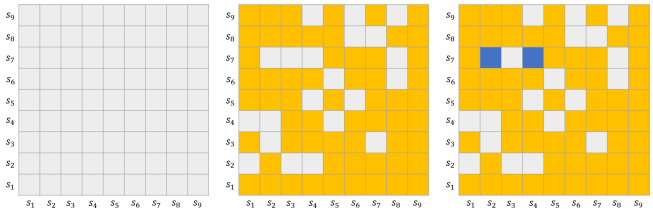


Fig. 3. Traffic topology extraction processes: (a) $\mathcal{G}(\mathcal{V},\mathcal{E})$, (b) $\mathcal{G}(\mathcal{V},\mathcal{E}')$, (c) $\mathcal{G}(\mathcal{V},\mathcal{E}'')$.

In order to make the above two stages clear, we give an example for the traffic topology extraction, as shown in the following example.

First, we use a complete directed graph in Fig. 3(a) to represent the traffic topology of the shared bus operational scenario, containing nine stations, i.e., $s_1 - s_9$. In $\mathcal{G}(\mathcal{V},\mathcal{E})$, the grey squares denote the edges between two nodes. In the first stage, we calculate the time network flow T_f and passenger network flow P_f of each edge, and then obtain $T_f^{\wedge} = 0.02862$ and $P_f^{\wedge} = 0.00083$ by leveraging the Ford-Fulkerson algorithm. Based on $T_f^{\wedge}$ and $P_f^{\wedge}$, we can determine $\check{T_f} = 0.02378$ and $\check{P_f} = 0.00031$, and delete the edge whose T_f and P_f is smaller than $\check{T_f}$ and $\check{P_f}$, respectively. After the first stage, the obtained traffic topology $\mathcal{G}(\mathcal{V},\mathcal{E}')$ is shown in Fig. 3(b), where the orange squares represent the deleted edges in the first state. In the second stage, we break two loops that do not contain the starting stations, i.e., $s_2 \rightarrow s_4 \rightarrow s_7 \rightarrow s_3 \rightarrow s_2$ and $s_2 \rightarrow s_7 \rightarrow s_3 \rightarrow s_2$. Based on Assumption 1, we delete edge $< s_4, s_7 >$ in the first loop and edge $< s_2, s_7 >$ in the second loop. Finally, we extract the traffic topology $\mathcal{G}(\mathcal{V},\mathcal{E}'')$ as shown in Fig. 3(c), where the blue squares denote the edges removed in the second stage. By analyzing the real-world traffic data, we find the traffic topology $\mathcal{G}(\mathcal{V},\mathcal{E}'')$ commonly exists in metropolis, such as Beijing and Shanghai, China, especially near the railway stations and airports, demonstrating the extendibility of our proposed traffic topology-based shared bus scheduling and route planning algorithms.

B. Candidate Line Generation

For simplicity, we relax constraint (7) to constraint (12), i.e.,

$$\left\{ \frac{1}{|\Phi_P(s,v)|} \sum_{p\in\Phi_P(s,v)} (b_{\mathfrak{a}}(s,v) - p_{\mathfrak{a}}(s,v)) \le T_w^{\wedge}, \quad \forall s \in \mathcal{S}, v \in \Phi(s) \right\}, \tag{12}$$

which means to guarantee the average waiting time of passengers in each station is no more than $T_w^{\wedge}$, rather than the waiting time of each passenger. We first present an improved local search algorithm to solve the relaxed BSRP problem, and the different routes of all shared buses in the solution are regarded as the candidate lines, which can be used for solving the BSRP problem in the following subsections.

In Algorithm 3, for global variables, $\mathcal{L}_\varsigma$ is the available shared bus set of the company, $\mathcal{L}_\varepsilon$ is the running shared bus set, $\mathcal{K}$ is the routing experience pool which stores the triads, i.e., $((current\_time, current\_station), next\_station)$, and $\check{\mathcal{T}_w}$ is the minimum value of the maximal average waiting time of passengers in the solutions obtained in all iterations. In each iteration, for local variables, $\mathcal{M}$ stores the routing experience, $\mathcal{C}$ is the line set of buses in the solution obtained, $\mathcal{T}_w$ is the maximum average waiting time of passengers, and $\mathfrak{D}^{\wedge}$ is the maximum route length. The iterative stopping criterion of the algorithm is the solution obtained in a certain iteration meets all the constraints of the relaxed BSRP problem, i.e., generating a feasible solution.

Algorithm 3 Candidate Line Generation Algorithm

Input: $\mathcal{P}$, $\mathcal{G}(\mathcal{V},\mathcal{E}'')$
Output: $\mathcal{C}$

Initialization $\mathcal{L}_\varsigma \leftarrow \varnothing$, $\mathcal{L}_\varepsilon \leftarrow \varnothing$, $\mathcal{K} \leftarrow \varnothing$, $\check{\mathcal{T}_w} \leftarrow \infty$;
while *the constraints of relaxed BSRP are not met* **do**
  $\mathcal{M} \leftarrow \varnothing$, $\mathcal{C} \leftarrow \varnothing$, $\mathcal{T}_w \leftarrow 0$, $\mathfrak{D}^{\wedge} \leftarrow 0$;
  while $\mathcal{P} \neq \varnothing$ **do**
    for *each bus* $\mathfrak{b} \in \mathcal{L}_\varepsilon$ **do**
      $b \leftarrow \mathfrak{b}$ with minimal arrival time $\mathfrak{b}_{\mathfrak{a}}(u)$;
    if *bus b randomly selects next station* **then**
      if $(b_{\mathfrak{a}}(u), u) \in \mathcal{K}$ **then**
        $v \leftarrow \mathcal{K}[(b_{\mathfrak{a}}(u), u)]$;
      else
        $v \leftarrow$ bus b randomly selects from $\mathfrak{V}_u$;
        $\mathcal{M} \leftarrow \mathcal{M} + ((u, b_{\mathfrak{a}}(u)), v)$;
    else
      for $\mathfrak{v} \in \mathfrak{V}_u$ **do**
        $v \leftarrow$ bus b selects $\mathfrak{v}$ with minimal $F_{\mathfrak{v}}$;
    Bus b goes to station v and $\mathcal{P}$ is updated;
    if $\frac{\sum_{p\in\Phi_P(v)}(b_{\mathfrak{a}}(v)-p_{\mathfrak{a}}(v))}{|\Phi_P(v)|} > \mathcal{T}_w$ **then**
      $\mathcal{T}_w \leftarrow \frac{\sum_{p\in\Phi_P(v)}(b_{\mathfrak{a}}(v)-p_{\mathfrak{a}}(v))}{|\Phi_P(v)|}$;
    if *bus b reaches the terminal* **then**
      $\mathcal{L}_\varsigma \leftarrow \mathcal{L}_\varsigma + b$, $\mathcal{L}_\varepsilon \leftarrow \mathcal{L}_\varepsilon - b$, $\mathcal{C} \leftarrow \mathcal{C} + \Phi$;
      if $\Phi_D > \mathfrak{D}^{\wedge}$ **then**
        $\mathfrak{D}^{\wedge} \leftarrow \Phi_D$;
    if *starting one bus* **then**
      if $\mathcal{L}_\varsigma \neq \varnothing$ **then**
        $b \leftarrow \mathcal{L}_\varsigma[1]$;
      else
        $b \leftarrow$ a new bus;
      $\mathcal{L}_\varepsilon \leftarrow \mathcal{L}_\varepsilon + b$;
    if $L_f = L_f^{\wedge}$ **then**
      Bus b cannot pick up passengers;
  if $\mathcal{T}_w \le \check{\mathcal{T}_w}$ *and* $\mathfrak{D}^{\wedge} \le D^{\wedge}$ **then**
    $\check{\mathcal{T}_w} \leftarrow \mathcal{T}_w$;
    Update $\mathcal{K}$ with $\mathcal{M}$;
Return $\mathcal{C}$

In each iteration, the algorithm schedules shared buses and plans routes to pick up all passengers to the terminal, i.e., $\mathcal{P} = \varnothing$. When planning routes for shared buses, the algorithm selects the next station for the shared bus that has the minimum arrival time. Without loss of generality, we consider bus b is the first one to arrive at the station, and its arrival station is u. Then, our algorithm can select the next station υ from three modes, i.e., selecting from the experience pool $\mathcal{K}$, randomly selecting from candidate station set $\mathfrak{S}_u$ and greedily selecting the optimal next station based on the priority of stations. The priority of stations is evaluated by $F_{\mathfrak{v}}$ in equation (13), i.e.,

$$F_{\mathfrak{v}} = \frac{e\,\langle u, \mathfrak{v}\rangle_d}{|\Phi_P(\mathfrak{v})|} 2^{(T_w^{\wedge} - \frac{\sum_{p\in\Phi_P(\mathfrak{v})}(b_{\mathfrak{a}}(\mathfrak{v}) - p_{\mathfrak{a}}(\mathfrak{v}))}{|\Phi_P(\mathfrak{v})|})/\tau}, \tag{13}$$

where τ is a constant to control the degree of constraint (12). A station $\mathfrak{v}$ with a smaller $F_{\mathfrak{v}}$ has a higher priority. In addition, the number of stations in the lines should also be considered to take full advantages of roads.

In order to determine whether scheduling the shared buses or not, the following three situations need to be checked: 1) When bus b reaches the terminal, it is removed from $\mathcal{L}_\varepsilon$ and appended into $\mathcal{L}_\varsigma$, and its route is stored in $\mathcal{C}$; 2) When it is time to start a shared bus by monitoring the average waiting time of passengers in each station, our algorithm prefers to select a shared bus from $\mathcal{L}_\varsigma$ if it is non-empty, otherwise it can assign a new shared bus; and 3) When the load factor of bus b reaches upper bound $L_f^{\wedge}$, it cannot pick up any passenger in the following stations. At the end of each iteration, if the maximum average waiting time of passengers $\mathcal{T}_w$ in the iteration is no more than its minimum value $\mathcal{T}_w^{\wedge}$ in all previous iterations, and the maximum route length $\mathfrak{D}^{\wedge}$ is no more than $D^{\wedge}$, referring to the current iteration generates the best solution so far, the algorithm can update the global experience pool $\mathcal{K}$ according to the routing experience $\mathcal{M}$ of the current iteration.

When the algorithm ends, it outputs the candidate line set $\mathcal{C}$. As illustrated in Fig. 4, there are four different candidate lines in $\mathcal{C}$ generated from the traffic topology $\mathcal{G}(\mathcal{V}, \mathcal{E}'')$, i.e., $s_1 \to s_2 \to s_3 \to s_7 \to s_8 \to s_9$, $s_1 \to s_4 \to s_5 \to s_6 \to s_9$, $s_5 \to s_6 \to s_8 \to s_7 \to s_3 \to s_2 \to s_4 \to s_9$, and $s_5 \to s_4 \to s_2 \to s_3 \to s_7 \to s_8 \to s_9$. Based on the candidate lines, we propose offline and online algorithms to solve the BSRP problem in the following subsections, respectively.

C. Offline: Trip Generation and Assignment

To effectively cope with the shared bus operational scenario with similar passenger flows, we propose a two-phase offline algorithm, named TIGAR. We decompose the BSRP problem into two subproblems, i.e., trip generation and trip assignment.

1) Trip Generation: The input are the candidate line set $\mathcal{C}$, passenger data $\mathcal{P}$ and traffic topology $\mathcal{G}(\mathcal{V}, \mathcal{E}'')$, and the output of *Trip Generation* is the set of trips $\mathcal{S}$, whose pseudo-code is illustrated in Algorithm 4. The objectives of this algorithm are to generate the minimum number of trips to meet the travel demands of all passengers, i.e., $\mathcal{P} = \varnothing$, and select the best route from the candidate line set for each trip. The

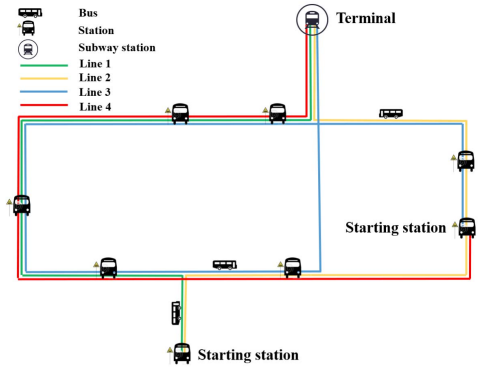


Fig. 4. A candidate line set.

Algorithm 4 Pseudo-Code of Trip Generation

Input: $\mathcal{C}$, $\mathcal{P}$, $\mathcal{G}(\mathcal{V}, \mathcal{E}'')$
Output: $\mathcal{S}$

Initialization $\mathcal{S} \leftarrow \varnothing$;
while $\mathcal{P} \neq \varnothing$ **do**
  $p \leftarrow$ the first arrival passenger in $\mathcal{P}$;
  $b_{\mathfrak{a}}(\upsilon) \leftarrow p_{\mathfrak{a}}(\upsilon) + T_w^{\wedge}$;
  $\mathcal{R} \leftarrow$ the set of lines containing station υ in $\mathcal{C}$;
  for *each route* $R \in \mathcal{R}$ **do**
    for *each passenger* $q \in R_P(u)$ *and* $u \in \upsilon_+$ **do**
      $q_w^{\wedge}(u) \leftarrow$ maximum waiting time of q;
    if $q_w^{\wedge}(u) > T_w^{\wedge}$ **then**
      $\hat{b}_{\mathfrak{a}}(\upsilon) \leftarrow b_{\mathfrak{a}}(\upsilon) - (q_w^{\wedge}(u) - T_w^{\wedge})$;
    for *each station* $u \in \upsilon\_$ **do**
      $\hat{b}_{\mathfrak{a}}(u) \leftarrow \hat{b}_{\mathfrak{a}}(u+1) - e\,\langle u, u+1\rangle_{\mathfrak{r}}(t)$;
    $R_{\mathfrak{d}} \leftarrow \hat{b}_{\mathfrak{a}}(\mathfrak{s})$;
    $\Phi \leftarrow R$ with minimal R_{DP};
  $s \leftarrow (\Phi, \Phi_{\mathfrak{d}})$;
  Update $\mathcal{P}$;
  $\mathcal{S} \leftarrow \mathcal{S} + s$;
Return $\mathcal{S}$

generation of each trip is based on the first passenger to the station in $\mathcal{P}$, the route Φ of each trip is selected from the reasonable candidate line set $\mathcal{R}$ and has the minimal Φ_{DP}, while the constraint for the waiting time of each passenger picked up in each trip and the departure time of each trip can be guaranteed and obtained by the *Forward-Backward process*, respectively.

For the former process, without loss of generality, we consider p is the first arrival passenger, and the arrival station is υ. For each candidate route containing station υ, symbol υ_+ represents the station set after station υ, and $\upsilon\_$ is the station set before station υ. *Forward process* only needs to check the waiting time of passengers picked up in υ_+ to guarantee the waiting time of each passenger picked up in the trip is no more than $T_w^{\wedge}$, since the waiting time of the passengers boarding at stations υ and $\upsilon\_$ can be guaranteed by setting the waiting time of passenger p to be $T_w^{\wedge}$. The corresponding proof can be seen in Theorem 1; The later process can deduce

Algorithm 5 Pseudo-Code of Trip Assignment

Input: $\mathcal{S}$
Output: $\mathcal{D}$
Initialization $\mathcal{L}_\varsigma \leftarrow \varnothing$, $\mathcal{L}_\varepsilon \leftarrow \varnothing$, $i \leftarrow 1$, $\mathcal{D} \leftarrow \varnothing$;
$s \leftarrow \mathcal{S}[1]$, $\mathcal{L}_\varepsilon \leftarrow \mathcal{L}_\varepsilon + \mathfrak{b}$, $\mathcal{D} \leftarrow \mathcal{D} + (i, s)$, $\mathcal{S} \leftarrow \mathcal{S} - s$;
while $\mathcal{S} \neq \varnothing$ **do**
  $b \leftarrow$ the first arrival bus in $\mathcal{L}_\varepsilon$;
  $b_\mathfrak{a}(v) \leftarrow b_\mathfrak{a}(u) + e\langle u, v\rangle_\mathfrak{r}(t)$;
  $s \leftarrow \mathcal{S}[1]$;
  while $\Phi_\mathfrak{d}(s) \leq b_\mathfrak{a}(v)$ **do**
    if $\mathcal{L}_\varsigma \neq \varnothing$ **then**
      $\mathcal{L}_\varepsilon \leftarrow \mathcal{L}_\varepsilon + \mathcal{L}_\varsigma[1]$, $\mathcal{L}_\varsigma \leftarrow \mathcal{L}_\varsigma - \mathcal{L}_\varsigma[1]$;
      $\mathcal{D} \leftarrow \mathcal{D} + (\mathcal{L}_\varsigma[1]_i, s)$;
    else
      $i \leftarrow i + 1$;
      $\mathcal{L}_\varepsilon \leftarrow \mathcal{L}_\varepsilon + \mathfrak{b}$, $\mathcal{D} \leftarrow \mathcal{D} + (i, s)$;
    $\mathcal{S} \leftarrow \mathcal{S} - s$, $s \leftarrow \mathcal{S}[1]$;
  Bus b goes to station v;
  if *bus b reaches the terminal* **then**
    $\mathcal{L}_\varepsilon \leftarrow \mathcal{L}_\varepsilon - b$, $\mathcal{L}_\varsigma \leftarrow \mathcal{L}_\varsigma + b$;
Return $\mathcal{D}$

the departure time $\Phi_\mathfrak{d}$ of trip s based on the traveling time $e\langle u-1, u\rangle_\mathfrak{r}(t)$ between two stations. Finally, all trips can be generated when $\mathcal{P}$ is empty.

2) Trip Assignment: The input is the candidate line set $\mathcal{S}$ obtained by the trip generation, *Trip Assignment* can output the result $\mathcal{D}$ of the BSRP problem, i.e., the union set of trips and their assigned shared buses. The idea of the *Trip Assignment* is to assign shared buses to the generated trips by comparing the departure time of the trips with the minimum arrival time of running shared buses. In the initialization, $\mathcal{L}_\varsigma$ is the set of available shared buses, $\mathcal{L}_\varepsilon$ is the set of the running shared buses, and i is the ID of shared buses. It is worth noting that the trips in $\mathcal{S}$ are ordered in an increasing order according to their departure time, and the first trip $\mathcal{S}[1]$ is assigned a new shared bus $\mathfrak{b}$ whose ID is 1. Without loss of generality, we consider bus b is the the first arrival bus in $\mathcal{L}_\varepsilon$, and its current station is u and next station is v. The algorithm can assign a shared bus to the first trip s by checking whether its departure time $\Phi_\mathfrak{d}(s)$ is no larger than the earliest arrival time $b_\mathfrak{a}(v)$ or not. In order to minimize the number of required buses, the shared bus $\mathcal{L}_\varsigma[1]$ in the set of available buses has priority to be assigned to the trip, where $\mathcal{L}_\varsigma[1]_i$ is its ID. Finally, all the trips are assigned shared buses, i.e., $\mathcal{S}$ is empty, and the solution $\mathcal{D}$ can be obtained. The details of the trip assignment process can be seen in Algorithm 5.

Theorem 1: The number of trips generated by the presented trip generation algorithm can be minimized, and the waiting time of each passenger is no more than $T_w^\wedge$.

Proof: We define the set of all passengers as $\mathcal{P}$, containing P passengers. Passenger p is the first one to arrive at the station, and the arrival station as well as arrival time are v and $p_\mathfrak{a}(v)$, respectively. The trip generated based on passenger p is s, and its route and departure time are $\Phi(s)$ and $\Phi_\mathfrak{d}(s)$, respectively. The set of passengers picked up in stations $v\_ = \{\mathfrak{s}, \ldots, v-1\}$ is $\Phi_P(s, v\_) = \bigcup_{u \in v\_} \Phi_P(s, u)$, and the counterpart in stations $v_+ = \{v+1, \ldots, J\}$ is $\Phi_P(s, v_+) = \bigcup_{u \in v_+} \Phi_P(s, u)$, where $\Phi_P(s, u)$ is the set of passengers picked up in station u of route $\Phi(s)$. The waiting time of passenger p is $p_w(v)$, satisfying $b_\mathfrak{a}(v) = p_\mathfrak{a}(v) + p_w(v)$, where $b_\mathfrak{a}(v)$ is the arrival time of bus b to station v. Then, we have $b_\mathfrak{a}(u) = p_\mathfrak{a}(v) + p_w(v) - e\langle u, v\rangle_\mathfrak{r}(t)$, $\forall u \in v\_$, where $e\langle u, v\rangle_\mathfrak{r}(t)$ is the running time of bus b from station u to station v at t^{th} time slot. Similarly, we have $b_\mathfrak{a}(u) = p_\mathfrak{a}(v) + p_w(v) + e\langle v, u\rangle_\mathfrak{r}(t)$, $\forall u \in v_+$. The number of passengers picked up by bus b in trip s is:

$$|\Phi_P(s)| = \sum_{u \in \Phi(s)} |\Phi_P(s, u)|. \tag{14}$$

Herein,

$$\begin{aligned}\Phi_P(s, u) &= \{q | q_\mathfrak{a}(u) \leq b_\mathfrak{a}(u), q \in \mathcal{P}\} \\ &= \{q | q_\mathfrak{a}(u) \leq p_\mathfrak{a}(v) + p_w(v) - e\langle u, v\rangle_\mathfrak{r}(t), \\ &\qquad q \in \mathcal{P} \wedge u \in v\_\},\end{aligned} \tag{15}$$

and vice versa for $u \in v_+$. It is obvious that the larger $p_w(v)$ is, the more passengers exist in $\Phi_P(s, u)$, and the more passengers picked up by shared bus in route $\Phi(s)$, i.e., $|\Phi_P(s)|$. Since the number of all passengers is finite, i.e., P, the maximum waiting time $p_w(v)$ can make the trip generation algorithm obtain the minimum number of trips. Therefore, the minimum number of trips can be generated in trip generation by setting $p_w(v)$ to $T_w^\wedge$. Finally, we prove the waiting time of each passenger picked up in trip s is no more than $T_w^\wedge$ from the following three situations:

(a) For $\forall q \in \Phi_P(s, v\_)$, considering passenger q arrives at station u at time $q_\mathfrak{a}(u)$, we have $q_\mathfrak{a}(u) > p_\mathfrak{a}(v)$, $u \in v\_$, since p is the first arrival passenger, and $b_\mathfrak{a}(u) < b_\mathfrak{a}(v)$. Then we can obtain $b_\mathfrak{a}(u) - q_\mathfrak{a}(u) < b_\mathfrak{a}(v) - p_\mathfrak{a}(v)$, i.e., $q_w(u) < p_w(v)$. Therefore, the waiting time of passengers picked up in stations $v\_$ is lower than that of passenger p, i.e., $T_w^\wedge$.

(b) For $\forall q \in \Phi_P(s, v) \wedge q \neq p$, we have $q_\mathfrak{a}(v) > p_\mathfrak{a}(v)$. The waiting time of passenger q is $q_w(v) = b_\mathfrak{a}(v) - q_\mathfrak{a}(v)$, and the waiting time of passenger p is $p_w(v) = b_\mathfrak{a}(v) - p_\mathfrak{a}(v)$. It is obvious that $q_w(v) < p_w(v)$, i.e., $q_w(v) < T_w^\wedge$.

(c) For $\forall q \in \Phi_P(s, v_+)$, we have $q_\mathfrak{a}(u) > p_\mathfrak{a}(v) \wedge b_\mathfrak{a}(u) > b_\mathfrak{a}(v), u \in v_+$. If the maximum waiting time of passenger q is larger than $T_w^\wedge$, i.e., $q_w^\wedge(u) = \max\{b_\mathfrak{a}(u) - q_\mathfrak{a}(u), \forall q \in \Phi_P(s, v_+)\}$ and $q_w^\wedge(u) > T_w^\wedge$, the arrival time of bus b to station v can be optimized to guarantee the user experience of passenger q, i.e., $\hat{b}_\mathfrak{a}(v) = b_\mathfrak{a}(v) - (q_w^\wedge(u) - T_w^\wedge)$, and then the arrival time of bus b to station u is changed into $\hat{b}_\mathfrak{a}(u) = b_\mathfrak{a}(u) - (q_w^\wedge(u) - T_w^\wedge)$. The waiting time of passenger p is:

$$\begin{aligned}p_w(v) &= \hat{b}_\mathfrak{a}(v) - p_\mathfrak{a}(v) \\ &= b_\mathfrak{a}(v) - p_\mathfrak{a}(v) - q_w^\wedge(u) + T_w^\wedge \\ &= 2T_w^\wedge - q_w^\wedge(u) \\ &< T_w^\wedge,\end{aligned} \tag{16}$$

Algorithm 6 Pseudo-Code of Passenger Assignment

Input: $\mathcal{C}$, $\mathcal{G}(\mathcal{V}, \mathcal{E}'')$, $\mathfrak{P}$
Output: $\mathcal{D}$
if $\mathfrak{P} \neq \varnothing$ **then**
  for *each passenger* $p \in \mathfrak{P}$ **do**
    for *each bus* $b \in \mathcal{L}_\varkappa$ **do**
      if $L_f < L_f^\wedge$ **then**
        $\check{b}_\mathfrak{a}(\upsilon) \leftarrow$ the earliest arrival time of bus b to station υ;
    if $\check{b}_\mathfrak{a}(\upsilon) - T_w^\wedge \leq p_\mathfrak{a}(\upsilon) \leq \check{b}_\mathfrak{a}(\upsilon)$ **then**
      Assign passenger p the bus b ;
    else if $\mathcal{L}_\varsigma \neq \varnothing$ **then**
      Obtain trip s similar with Algorithm 4;
      Assign passenger p the $\mathcal{L}_\varsigma[1]$;
      $\mathcal{L}_\varkappa \leftarrow \mathcal{L}_\varkappa + \mathcal{L}_\varsigma[1]$, $\mathcal{L}_\varsigma \leftarrow \mathcal{L}_\varsigma - \mathcal{L}_\varsigma[1]$;
    else
      Obtain trip s similar with Algorithm 4;
      Assign passenger p a new bus $\mathfrak{b}$;
      $\mathcal{L}_\varkappa \leftarrow \mathcal{L}_\varkappa + \mathfrak{b}$;
for *each bus* $b \in \mathcal{L}_\varkappa$ **do**
  if *bus* b *starts* **then**
    $\mathcal{D} \leftarrow \mathcal{D} + (b_i, s)$, $\mathcal{L}_\varepsilon \leftarrow \mathcal{L}_\varepsilon + b$;
for *each bus* $b \in \mathcal{L}_\varepsilon$ **do**
  Update its running time;
  if *bus* b *reaches the terminal* **then**
    $\mathcal{L}_\varkappa \leftarrow \mathcal{L}_\varkappa - b$, $\mathcal{L}_\varepsilon \leftarrow \mathcal{L}_\varepsilon - b$, $\mathcal{L}_\varsigma \leftarrow \mathcal{L}_\varsigma + b$;
Return $\mathcal{D}$

since $q_w^\wedge(u) > T_w^\wedge$ holds. Since $q_w^\wedge(u) \geq b_\mathfrak{a}(u) - q_\mathfrak{a}(u)$ holds, the waiting time of passenger q is:

$$\begin{aligned} q_w(u) &= \hat{b}_\mathfrak{a}(u) - q_\mathfrak{a}(u) \\ &= b_\mathfrak{a}(u) - q_\mathfrak{a}(u) - q_w^\wedge(u) + T_w^\wedge \\ &\leq T_w^\wedge, \end{aligned} \tag{17}$$

If the maximum waiting time of passenger q is no more than $T_w^\wedge$, i.e., $q_w^\wedge(u) = \max\{b_\mathfrak{a}(u) - q_\mathfrak{a}(u), \forall q \in \Phi_P(s, \upsilon_+)\}$ and $q_w^\wedge(u) \leq T_w^\wedge$, we can get the waiting time of passenger p equals to $T_w^\wedge$, i.e., $p_w(\upsilon) = T_w^\wedge$, and the waiting time of passenger q is no more than $T_w^\wedge$, i.e., $q_w(u) \leq T_w^\wedge$. Therefore, the waiting time of each passenger picked up in trip s is no more than $T_w^\wedge$. □

D. Online: Arrival Data-Based Passenger Assignment

Although TIGAR can solve BSRP with the similar passenger flows in an offline way, an online algorithm is necessary to handle the dynamic and real-time passenger flows. Thus, an online algorithm, named ADPT, is proposed.

The idea of ADPT is to assign the arrival passengers in real-time by executing *Passenger Assignment algorithm*, whose pseudo-code is given in Algorithm 6. The input are the candidate line set $\mathcal{C}$, the extracted traffic topology $\mathcal{G}(\mathcal{V}, \mathcal{E}'')$ and the set of passengers $\mathfrak{P}$ arriving at stations in real-time, the algorithm can output the solution $\mathcal{D}$ of BSRP. In the passenger assignment process, $\mathcal{L}_\varsigma$ is the set of available shared buses, $\mathcal{L}_\varepsilon$ is the set of running buses, and $\mathcal{L}_\varkappa$ is the set of existing buses containing the running buses as well as the shared buses whose routes and departure time are determined, but have not started due to their late departure time.

The details of the passenger assignment are described as follows: Without loss of generality, we consider passenger p arrives at station υ in $\mathfrak{P}$. In order to assign passenger p to a shared bus, there are three modes: 1) When the first arrival bus b to station υ in $\mathcal{L}_\varkappa$ can guarantee the waiting time of p is no more than $T_w^\wedge$, passenger p is assigned to bus b to increase the load factor of buses and reduce the number of the required buses; 2) When passenger p cannot be assigned to a bus in $\mathcal{L}_\varkappa$ and $\mathcal{L}_\varsigma$ is non-empty, the algorithm first generates a trip based on passenger p similar with Algorithm 4, and then assigns passenger p to shared bus $\mathcal{L}_\varsigma[1]$ in the set of available buses to reduce the number of required buses; and 3) Otherwise, passenger p can be assigned to a new shared bus $\mathfrak{b}$. Note that different from Algorithm 4, the passenger assignment algorithm does not need to execute the *Forward process*, since the passengers boarding at the subsequent stations υ_+ in the future cannot be obtained, and the algorithm selects the shortest route from the reasonable candidate line set $\mathcal{R}$. After all passengers are assigned buses, i.e., $\mathfrak{P} = \varnothing$, the algorithm generates solution $\mathcal{D}$.

The proposed ADPT algorithm can generate dynamic schedules for shared buses based on the passenger flows in real time to reduce the operating cost of bus companies and guarantee the user experience of each passenger. ADPT has a strong robustness and has no requirement for passenger flow distributions. It is able to cope with the dynamic vehicle issues (e.g., traffic jams). The superiority of ADPT is that it can cope with the BSRP problem in the scenario with real-time and dynamic passenger flows. However, TIGAR works well for the scenario with similar passenger flow distributions. Thus, ADPT is more robust and applicable than TIGAR.

E. Time Complexity Analysis

In this subsection, we analyze the time complexity of TIGAR and ADPT in the worst-case scenario.

Proposition 2: *The time complexity of TIGAR is* $\mathcal{O}(N^2M^2 + NMLV)$, *where* N *is the operating time of shared bus companies,* M *is the maximum number of passengers arriving at stations per second in the worst case,* L *is the number of candidate lines, and* V *is the number of stations.*

Proof: The complexity of TIGAR containts two parts, i.e., trip generation and trip assignment. The former mainly contains a while loop, whose complexity is $\mathcal{O}(|\mathcal{P}|)$, and $\mathcal{P}$ is the set of all passengers. The complexity of searching the first arrival passenger is $\mathcal{O}(|\mathcal{P}|)$, and that of obtaining set $\mathcal{R}$ is $\mathcal{O}(|\mathcal{C}|)$, where $|\mathcal{C}|$ is the number of candidate lines. Due to (6b) for the load factor of shared buses, the number of passengers picked up in stations υ_+ is lower than $\lambda L_f^\wedge$, where λ is the number of seats in the bus. The complexity of trip generation

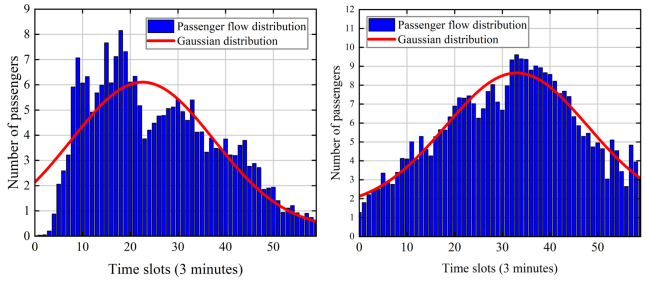


Fig. 5. Passenger flow distributions: (a) Morning peak (06:30:00 - 09:30:00), (b) Evening peak (17:30:00 - 20:30:00).

is $\mathcal{O}(|\mathcal{P}|\,(|\mathcal{P}|+|\mathcal{C}|+|\mathcal{C}|\,(\lambda L_f^{\wedge}+V)))$ in the worst case. In trip assignment, the time complexity of the two while loops is $\mathcal{O}(|\mathcal{S}|)$, and that for visiting $\mathcal{L}_\varepsilon$ is $\mathcal{O}(|\mathcal{B}|)$ in the worst case, where $\mathcal{S}$ and $\mathcal{B}$ are the sets of trips and required shared buses to meet the travel demands of all passengers, respectively. It is obvious that the complexity of trip assignment is $\mathcal{O}(|\mathcal{S}|\,(|\mathcal{B}|+|\mathcal{S}|))$. Finally, the time complexity of TIGAR is $\mathcal{O}(|\mathcal{P}|\,(|\mathcal{P}|+|\mathcal{C}|+|\mathcal{C}|\,(\lambda L_f^{\wedge}+V))+|\mathcal{S}|\,(|\mathcal{B}|+|\mathcal{S}|))$. Since $|\mathcal{P}|>|\mathcal{S}|$, $|\mathcal{C}|\,V>|\mathcal{B}|$, and $\lambda L_f^{\wedge}$ is a constant, it can be simplified to $\mathcal{O}(|\mathcal{P}|^2+|\mathcal{P}|\,|\mathcal{C}|\,V)$.

By analyzing the real-world shared bus data, we find the passenger flow distribution can be represented by *Gaussian* distribution, as shown in Fig. 5. We can calculate the maximum number of passengers arriving at stations in a time slot in the worst case as:

$$M=\mathfrak{A}+\frac{\mathfrak{B}}{\mathfrak{C}\sqrt{\frac{\pi}{2}}}, \tag{18}$$

where the values of variables $\mathfrak{A},\mathfrak{B}$ and $\mathfrak{C}$ are distinct for different passenger flow distributions. The number of all passengers P can be calculated by NM in the worst case. Obviously, N is a variable related with the number of stations V and influences the number of buses $|\mathcal{B}|$. Define the number of candidate lines as L, and then the time complexity of TIGAR is $\mathcal{O}(N^2M^2+NMLV)$. □

Proposition 3: The time complexity of ADPT is $\mathcal{O}(MLV)$.

Proof: ADPT is an online algorithm, which needs to execute the passenger assignment algorithm one time per second to meet the travel demands of all passengers. For visiting the sets of $\mathfrak{P}$ and $\mathcal{L}_\varkappa$, their time complexity is $\mathcal{O}(M)$ and $\mathcal{O}(|\mathcal{B}|)$ in the worst case, respectively. The time complexity of determining the route and departure time of each trip is $\mathcal{O}(|\mathcal{C}|\,V)$, and that of determining the start and end of buses is $\mathcal{O}(|\mathcal{B}|)$. Therefore, the time complexity of ADPT is $\mathcal{O}(M(|\mathcal{B}|+|\mathcal{C}|\,V)\ +|\mathcal{B}|+|\mathcal{B}|)$. Since $|\mathcal{B}|<|\mathcal{C}|\,V$, it can be simplified to $\mathcal{O}(M\,|\mathcal{C}|\,V)$. Since the number of candidate lines is L, the time complexity of ADPT is $\mathcal{O}(MLV)$. □

We can observe that ADPT only needs to deal with the passengers arriving at stations per second rather than all passengers, and can achieve the goal of significantly reducing the time complexity compared with TIGAR.

V. Performance Evaluation

In this section, we evaluate the performance of TIGAR and ADPT based on a real-world order data set, collected by Panda bus company for the shared bus from 13 March 2017 to 9 September 2017 in Yongkang City, Shanghai (China). The shared bus operational scenario is between the residential areas and subway stations, which contains nine shared bus stations, and hundreds of passengers ride the shared buses every day. The order data set includes the passenger data as well as the GPS data of buses, and contains near 50,000 ride records. Six features are extracted from the data set: 1) Boarding station ID; 2) Station ID of getting off the bus; 3) Arrival time of passengers to the station; 4) Boarding time; 5) Distance between two stations; and 6) Running time between two stations at different time.

The extracted traffic topology is shown in Fig. 2 and the generated candidate line set is illustrated in Fig. 4. We compare our algorithms with HOSVD [21] and ACO-based method [20] with respect to three performance metrics, i.e., the total distance of all trips, average number of passengers picked up in each trip, and number of required buses.

A. Comparison in Terms of Waiting Time

The experimental results of various schemes for different upper bounds of waiting time $T_w^{\wedge}$ in one week is shown in Fig. 6, and the number of seats is fixed to 18. For different schemes, the values of total distance and average number of passengers are the average of the experimental results obtained in one week, while the number of required buses is the maximum value of the experimental results in one week. Fig. 6(a) shows the performance of the total distance for different methods. We can discover that the total distances of TIGAR and ADPT are both shorter than those of HOSVD and ACO-based methods. This is because TIGAR and ADPT can select suitable routes from the candidate line set and generate fewer trips, while HOSVD and ACO select routes based on the real-world traffic topology, but some routes are unsuitable for the running of shared buses due to the long length. This result shows that TIGAR and ADPT can greatly reduce the operating cost by decreasing the length of routes, which can demonstrate the effectiveness of the extracted traffic topology and candidate line set. Herein, ADPT is slightly inferior to TIGAR, because the offline algorithm TIGAR can optimize the bus scheduling and route planning from the global perspective, and ADPT can gradually obtain the local passenger demands in real time as time goes by.

From Fig. 6(b), we can observe that the average number of the passengers of TIGAR and ADPT is both higher than that of HOSVD and ACO, i.e., our solutions can reduce the operating cost by optimizing the number of passengers picked up in each trip. TIGAR performs the best because it can generate the minimal number of trips and guarantee the waiting time of each passenger, while ADPT, HOSVD and ACO have to guarantee the waiting time of passengers by adding extra trips. The performance of ADPT is better than that of HOSVD and ACO, since only if the existing trips cannot meet the demands

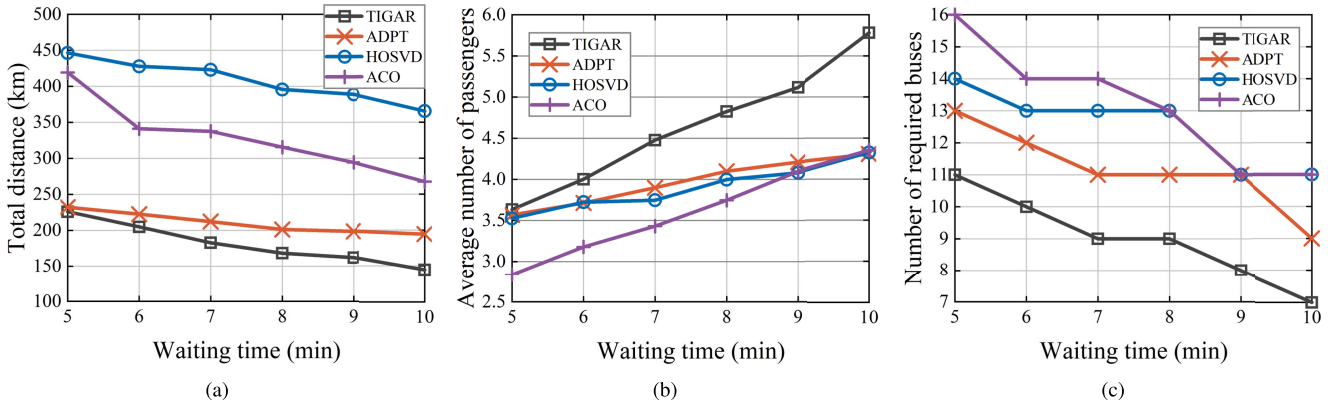


Fig. 6. Performance comparison for different waiting time in one week: (a) Total distance, (b) Average number of passengers, (c) Number of required buses.

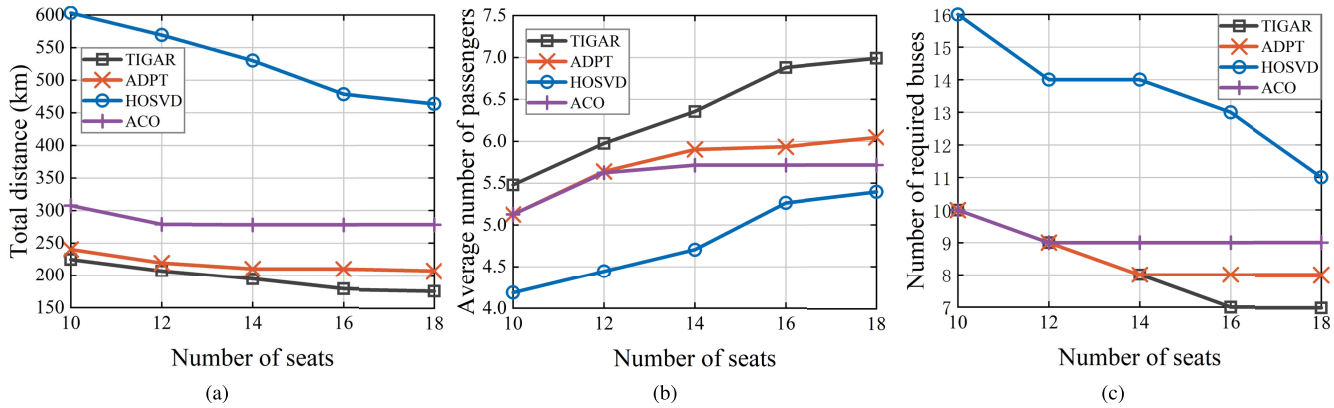


Fig. 7. Performance comparison for different number of seats in one day: (a) Total distance, (b) Average number of passengers, (c) Number of required buses.

of passengers, it adds one trip. However, HOSVD and ACO add trips by monitoring the waiting time of passengers, which results in more trips.

In Fig. 6(c), it can be seen that TIGAR and ADPT require fewer buses than HOSVD and ACO to meet the travel demands of all passengers for different upper bounds of waiting time. The reason is that TIGAR and ADPT can generate fewer trips and determine better departure time for each trip. They can take full advantages of shared buses, i.e., giving priority to the running buses to provide services for passengers as much as possible. However, HOSVD and ACO generate more trips and lack the optimization of departure time, resulting in more required buses.

B. Comparison for Different Number of Seats

The performance of various schemes for different number of seats in one day is shown in Fig. 7, while the upper bound of waiting time $T_w^{\wedge}$ is fixed to 10 minutes. We can observe that TIGAR and ADPT have shorter route length, larger average number of passengers and fewer number of required buses than the counterparts in HOSVD and ACO for different number of seats. The reason is similar with that illustrated before. The performance of ACO is better than that of HOSVD for different number of seats. However, for different upper bounds of waiting time, the performance of ACO is worse than HOSVD in the average number of passengers and the number of required buses. It indicates that ACO is sensitive to the upper bound of waiting time, and HOSVD is sensitive to the number of seats. The performance of TIGAR and ADPT is always superior to those achieved by HOSVD and ACO, illustrating they are robust to both the upper bound of waiting time and the number of seats. The number of required buses of TIGAR, ADPT and ACO no longer improves when the number of seats reaches 16, 14 and 12, respectively. The reason is that the number of buses is enough to meet the travel demands of all passengers.

C. Comparison With Real-World Scheduling Data

We compare the four schemes with the real-world scheduling data regulated by the Panda bus company. The number of seats is fixed to 18, and the upper bound of waiting time is fixed to 10 minutes. In order to intuitively evaluate the improvement brought by different methods over the real-world scheduling of the bus company, we define Performance Improvement Rate (PIR) to measure their achieved performance in terms of total route distance, average number of passengers in each trip, and the number of required buses. It equals to the gap between a method performance

TABLE II
PERFORMANCE IMPROVEMENT RATE COMPARED WITH REAL DATA

Metric	Method	Weekday					Weekend	
		Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Total distance	TIGAR	**21.77%**	**41.32%**	**36.32%**	**34.98%**	**33.49%**	**65.27%**	**65.27%**
	ADPT	18.72%	25.98%	8.7%	17.97%	6.53%	62.55%	62.98%
	HOSVD	-86.81%	-86.47%	-75.58%	-51.38%	-88.74%	29.80%	33.29%
	ACO	-37.96%	-3.29%	-33.01%	-25.24%	-27.21%	33.87%	47.85%
Average number of passengers	TIGAR	**25%**	**66.67%**	**53.81%**	**53.75%**	**48.21%**	**185%**	**185.33%**
	ADPT	21.18%	33.33%	8.1%	21%	5.18%	166%	166.67%
	HOSVD	4.12%	8.04%	2.38%	11%	5.18%	150%	166.67%
	ACO	-20%	17.65%	17.62%	8%	-4.78%	135%	134.67%
Number of required buses	TIGAR	**12.5%**	**25%**	**12.5%**	**25%**	**25%**	**50%**	**62.5%**
	ADPT	0%	12.5%	-12.5%	12.5%	-12.5%	**50%**	50%
	HOSVD	-37.5%	-37.5%	-37.5%	-37.5%	-37.5%	25%	37.5%
	ACO	-37.5%	-12.5%	-12.5%	-37.5%	-12.5%	37.5%	37.5%

and the real-world scheduling performance divided by latter. A positive PIR means the performance of the method is better than the latter, and vice versa. The absolute value of PIR is positively related to the gap between the method's and the real-world scheduling performance.

Table II shows the PIRs of four schemes for different performance metrics in one week. On weekdays, we can discover that TIGAR and ADPT can both reduce the total route length, comparing with the real-world scheduling data. The reason is that we extract a traffic topology suitable for the shared bus operations, and plan routes based on the topology, while the routes of the real-world scheduling are overlaped and circuitous. The performance of HOSVD and ACO is worse than the real-world scheduling data, because they plan routes based on the real-world traffic topology, but generate longer routes than the real-world scheduling. The average number of passengers of TIGAR and ADPT is larger than the real-world scheduling data, because TIGAR can generate the minimum number of trips, and ADPT adds one trip only if the existing trips cannot meet the travel demands of passengers. However, the number of trips of the real-world scheduling is fixed and relatively large, determined by the experience of decision-makers. The performance of HOSVD is better than the real-world scheduling data, because the tensor-based station selection can optimize the number of passengers when the number of seats is enough, while the performance of iterative heuristic ACO is not stable. The number of the required buses of TIGAR is fewer than the real-world scheduling data. The reason is that TIGAR can generate the minimum number of trips as well as an optimal departure time to meet the travel demands. Although HOSVD generates less trips than the real-world scheduling data, it requires more buses, because it plans longer routes than real-world scheduling data and lacks the optimization of departure time. ADPT generates less trips than the real-world scheduling data, but it requires more buses, since the number of buses in the real-world scheduling is determined by the experience of decision-makers, ignoring the personal experience of each passenger. However, the number of additional required shared buses in ADPT is very small, and generally another one shared bus is enough to meet the personal user experience compared with the real-world scheduling, which is acceptable. It is worth noting that at weekends, the performance of these four methods makes a great improvement for all metrics, comparing with the real-world scheduling data. The reason is that the number of passengers is relatively small at weekends. The four schemes can plan routes and schedule buses based on the intraday passenger flows, while the real-world scheduling has a fixed timetable and has to schedule buses as that on weekdays. Note that the waiting time of each passenger in TIGAR, ADPT, HOSVD and ACO is no more than 10 minutes, but the waiting time of passengers in the real-world scheduling cannot be guaranteed. This result also demonstrates that the dynamic schedule generated by our solutions is more suitable for the shared buses than the fixed scheduling to reduce the operating cost of bus companies and guarantee the user experience of each passenger.

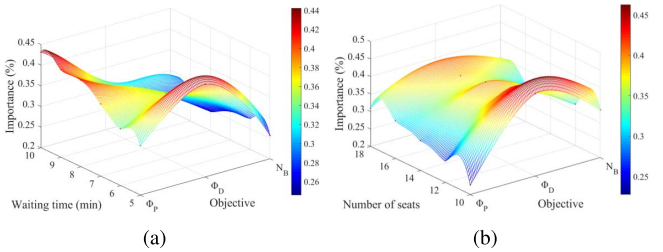


Fig. 8. Importance of objectives for different: (a) Upper bounds of waiting time, and (b) Number of seats.

D. Importance of Objectives

Fig. 8(a) shows the importance of three objectives, i.e., Φ_P, Φ_D and N_B, for different upper bounds of waiting time $T_w^{\wedge}$. The importance of an optimization objective can be calculated by the ratio of PIR for the objective of the average performance of the four schemes, e.g., for objective Φ_P: $PIR_{\Phi_P} = \frac{\overline{\Phi}_P - \check{\Phi}_P}{\check{\Phi}_P}$, and $W_{\Phi_P} = \frac{PIR_{\Phi_P}}{PIR_{\Phi_P}+PIR_{\Phi_D}+PIR_{N_B}}$. Herein, PIR_{Φ_P} is the PIR of the average performance of the four schemes for objective Φ_P, W_{Φ_P} is the importance of Φ_P for reducing the operating cost, $\overline{\Phi}_P$ is the average performance of the four schemes (i.e., TIGAR, ADPT, HOSVD and ACO), and $\check{\Phi}_P$ is

the performance benchmark of Φ_P, which equals the worst performance result of the four schemes. It is obvious that the importance of Φ_P increases with waiting time, while Φ_D decreases. Among three objectives, Φ_P is the most important, indicating that bus companies can mainly reduce the operating cost by increasing the average number of passengers picked up by each bus for different upper bounds of waiting time. The importance of these three objectives for different number of seats is illustrated in Fig. 8(b). We can discover that the importance of Φ_P increases as the number of seats increases, while the increasing extent is small. Φ_D and N_B are both more significant than Φ_P. The most important objective is Φ_D, indicating that bus companies can mainly reduce the operating cost by decreasing the route length for various available number of seats.

VI. Conclusion

In this paper, we propose a joint bus scheduling and route planning framework to solve the formulated BSRP problem, considering both the operating cost and user experience. We first extract a traffic topology suitable for the running of shared buses, based on which the candidate lines are generated. In order to effectively cope with the similar passenger flow distributions, an offline algorithm, i.e., TIGAR, is developed to generate the minimum number of trips with the optimal route and departure time, by performing trip generation and trip assignment. Furthermore, an online algorithm, i.e., ADPT, is investigated to handle the dynamic passenger flows, which can reduce the time complexity by planning routes and scheduling buses for passengers in real time. Based on the real-world data set, extensive experiments demonstrate that the proposed schemes outperform other schemes with shorter routes, larger average number of passengers and fewer required buses. Compared with the real-world scheduling data, TIGAR is superior in all three aspects, i.e., the route length, average number of passengers and the number of required buses, while ADPT has some minor performance degradation in the number of required buses.

References

[1] Z. Ning, J. Huang, X. Wang, J. J. P. C. Rodrigues, and L. Guo, "Mobile edge computing-enabled Internet of vehicles: Toward energy-efficient scheduling," *IEEE Netw.*, vol. 33, no. 5, pp. 198–205, Sep. 2019.

[2] C. Courcoubetis and A. Dimakis, "Throughput and pricing of ridesharing systems," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 640–648.

[3] G. Xiong *et al.*, "Cyber-physical-social system in intelligent transportation," *IEEE/CAA J. Automatica Sinica*, vol. 2, no. 3, pp. 320–333, Jul. 2015.

[4] M. Ghahramani, M. Zhou, and G. Wang, "Urban sensing based on mobile phone data: Approaches, applications, and challenges," *IEEE/CAA J. Automatica Sinica*, vol. 7, no. 3, pp. 627–637, May 2020.

[5] Y. Wang, D. Zhang, L. Hu, Y. Yang, and L. H. Lee, "A data-driven and optimal bus scheduling model with time-dependent traffic and demand," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2443–2452, Sep. 2017.

[6] M. Wen, E. Linde, S. Ropke, P. Mirchandani, and A. Larsen, "An adaptive large neighborhood search heuristic for the electric vehicle scheduling problem," *Comput. Oper. Res.*, vol. 76, pp. 73–83, Dec. 2016.

[7] Y. Liu *et al.*, "Intelligent bus routing with heterogeneous human mobility patterns," *Knowl. Inf. Syst.*, vol. 50, no. 2, pp. 383–415, Feb. 2017.

[8] S. P. Chuah, H. Wu, Y. Lu, L. Yu, and S. Bressan, "Bus routes design and optimization via taxi data analytics," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2016, pp. 2417–2420.

[9] X. Zuo, C. Chen, W. Tan, and M. Zhou, "Vehicle scheduling of an urban bus line via an improved multiobjective genetic algorithm," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 1030–1041, Apr. 2015.

[10] V. Boyer, O. J. Ibarra-Rojas, and Y. Á. Ríos-Solís, "Vehicle and crew scheduling for flexible bus transportation systems," *Transp. Res. B, Methodol.*, vol. 112, pp. 216–229, Jun. 2018.

[11] T. Liu, A. Ceder, J. Ma, W. Guan, and L. Zhou, "Graphical human–machine interactive approach for integrated bus transit scheduling: Lessons gained from a large bus company," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 4, pp. 1023–1028, Apr. 2017.

[12] T. Otsuki and K. Aihara, "New variable depth local search for multiple depot vehicle scheduling problems," *J. Heuristics*, vol. 22, no. 4, pp. 567–585, Aug. 2016.

[13] L. Desfontaines and G. Desaulniers, "Multiple depot vehicle scheduling with controlled trip shifting," *Transp. Res. B, Methodol.*, vol. 113, pp. 34–53, Jul. 2018.

[14] J. Zhang *et al.*, "A real-time passenger flow estimation and prediction method for urban bus transit systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 11, pp. 3168–3178, Nov. 2017.

[15] C. Tang, A. Ceder, S. Zhao, and Y.-E. Ge, "Vehicle scheduling of single-line bus service using operational strategies," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 30, pp. 1149–1159, Mar. 2019.

[16] B. A. Kumar, G. H. Prasath, and L. Vanajakshi, "Dynamic bus scheduling based on real-time demand and travel time," *Int. J. Civil Eng.*, vol. 17, no. 9, pp. 1481–1489, Sep. 2019.

[17] E. H.-C. Lu, H.-S. Chen, and V. S. Tseng, "An efficient framework for multirequest route planning in urban environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 4, pp. 869–879, Apr. 2017.

[18] L. Wang and J. Lu, "A memetic algorithm with competition for the capacitated green vehicle routing problem," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 2, pp. 516–526, Mar. 2019.

[19] C. Chen, D. Zhang, N. Li, and Z.-H. Zhou, "B-planner: Planning bidirectional night bus routes using large-scale taxi GPS traces," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 4, pp. 1451–1465, Aug. 2014.

[20] D. Cerotti, S. Distefano, G. Merlino, and A. Puliafito, "A crowd-cooperative approach for intelligent transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1529–1539, Jun. 2017.

[21] X. Zhu, R. Hao, H. Chi, and X. Du, "FineRoute: Personalized and time-aware route recommendation based on check-ins," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10461–10469, Nov. 2017.

[22] Q. Lin, L. Deng, J. Sun, and M. Chen, "Optimal demand-aware ride-sharing routing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 2699–2707.

[23] Y. Yao, Z. Peng, and B. Xiao, "Parallel hyper-heuristic algorithm for multi-objective route planning in a smart city," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10307–10318, Nov. 2018.

[24] J. Li *et al.*, "A traffic prediction enabled double rewarded value iteration network for route planning," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4170–4181, May 2019.

[25] M. Michaelis and A. Schöbel, "Integrating line planning, timetabling, and vehicle scheduling: A customer-oriented heuristic," *Public Transp.*, vol. 1, no. 3, p. 211, Aug. 2009.

[26] A. Schöbel, "An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation," *Transp. Res. C, Emerg. Technol.*, vol. 74, pp. 348–365, Jan. 2017.

[27] X. Kong, M. Li, T. Tang, K. Tian, L. Moreira-Matias, and F. Xia, "Shared subway shuttle bus route planning based on transport data analytics," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 4, pp. 1507–1520, Oct. 2018.

[28] Y. Pei and M. W. Mutka, "STARS: Static relays for remote sensing in multirobot real-time search and monitoring," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 10, pp. 2079–2089, Oct. 2013.

[29] X. Meng, J. Li, X. Dai, and J. Dou, "Variable neighborhood search for a colored traveling salesman problem," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 4, pp. 1018–1026, Apr. 2018.

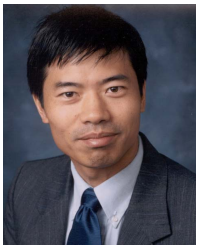
[30] C. Wang, J. Li, F. Ye, and Y. Yang, "NETWRAP: An NDN based real-TimeWireless recharging framework for wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 6, pp. 1283–1297, Jun. 2014.

[31] Q. Kang, S. Feng, M. Zhou, A. C. Ammari, and K. Sedraoui, "Optimal load scheduling of plug-in hybrid electric vehicles via weight-aggregation multi-objective evolutionary algorithms," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2557–2568, Sep. 2017.

[32] H. Yuan, J. Bi, W. Tan, M. Zhou, B. H. Li, and J. Li, "TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.
[33] Y. Liu *et al.*, "Solving NP-hard problems with physarum-based ant colony system," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 14, no. 1, pp. 108–120, Jan. 2017.
[34] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *Eur. J. Oper. Res.*, vol. 59, no. 2, pp. 231–247, 1992.
[35] Z.-H. Zou, Y. Yun, and J.-N. Sun, "Entropy method for determination of weight of evaluating indicators in fuzzy synthetic evaluation for water quality assessment," *J. Environ. Sci.*, vol. 18, no. 5, pp. 1020–1023, Sep. 2006.
[36] V. Vatter. (2004). *Graphs, Flows, and the Ford-Fulkerson Algorithm*. Tersedia di. [Online]. Available: http://www.math.ufl.edu/~vatter/teaching/flow.pdf.[diakses2013]

Zhaolong Ning (Senior Member, IEEE) received the M.S. and Ph.D. degrees from Northeastern University, Shenyang, China. He was a Research Fellow with Kyushu University, Japan. He is currently a Distinguished Professor with the Chongqing University of Posts and Telecommunications, China, and an Associate Professor with the Dalian University of Technology, China. He has published over 120 scientific papers in international journals and conferences. His research interests include the Internet of Things, mobile edge computing, and resource management.

Shouming Sun received the B.S. degree from the Dalian University of Technology, Dalian, China, in 2018, where he is currently pursuing the M.S. degree. His research interests include mobile edge computing, artificial intelligence, and blockchain.



MengChu Zhou (Fellow, IEEE) joined the New Jersey Institute of Technology in 1990, where he is currently a Distinguished Professor. He has over 900 publications, including 12 books, more than 600 journal articles (more than 450 in IEEE TRANSACTIONS), and 29 book-chapters and holds 26 patents. His research interests include Petri nets, intelligent automation, the Internet of Things, and big data. He was a recipient of the Humboldt Research Award for U.S. Senior Scientists from Alexander von Humboldt Foundation, the Franklin V. Taylor Memorial Award, and the Norbert Wiener Award from IEEE Systems, Man and Cybernetics Society. He is a fellow of IFAC, AAAS, and CAA. He is the Founding Editor of IEEE Press Book Series on *Systems Science and Engineering* and the Editor-in-Chief of IEEE/CAA JOURNAL OF AUTOMATICA SINICA.

Xiping Hu (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from The University of British Columbia, Vancouver, Canada. He is currently a Professor with Lanzhou University, China. He was the Co-Founder and CTO of Bravolol Limited, Hong Kong, a leading language learning mobile application company with over 100 million users, and listed as top 2 language education platform globally. He has around 90 articles published and presented in prestigious conferences and journals. His research areas include distributed intelligent systems, crowdsensing, social networks, and cloud computing.

Xiaojie Wang received the M.S. degree from Northeastern University, China, in 2011, and the Ph.D. degree from the Dalian University of Technology, Dalian, China, in 2019. From 2011 to 2015, she was a Software Engineer with NeuSoft Corporation, China. From 2019 to 2020, she held a post-doctoral position with The Hong Kong Polytechnic University. She is currently a Distinguished Professor with the Chongqing University of Posts and Telecommunications, China. Her research interests include the Internet of Things, mobile edge computing, and machine learning.

Lei Guo received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2006. He is currently a Full Professor with the Chongqing University of Posts and Telecommunications, Chongqing, China. He has authored or coauthored more than 200 technical papers in international journals and conferences. His current research interests include communication networks, optical communications, and wireless communications. He is also an editor for several international journals.

Bin Hu (Senior Member, IEEE) is currently a Professor with Lanzhou University and a Guest Professor with ETH Zürich, Switzerland. He is a fellow of IET, the co-chairs of IEEE SMC TC on Cognitive Computing, a Member-at-Large of ACM China, and the Vice President of International Society for Social Neuroscience (China Committee). He has published more than 100 papers in peer reviewed journals, conferences, and book chapters. He has served as the chairs/co-chairs for many IEEE international conferences/workshops, and an associate editor for peer reviewed journals on cognitive science and pervasive computing, such as IEEE TRANSACTIONS ON AFFECTIVE COMPUTING, *Brain Informatics*, and *IET Communications*.

Ricky Y. K. Kwok (Fellow, IEEE) received the B.Sc. degree in computer engineering from The University of Hong Kong in 1991, and the M.Phil. and Ph.D. degrees in computer science from The Hong Kong University of Science and Technology (HKUST) in 1994 and 1997, respectively. His research focus has been on designing efficient communication protocols and robust resources management algorithms toward enabling large scale distributed mobile computing. In these research areas, he has authored one textbook, coauthored another two textbooks, and published more than 200 technical papers in various leading journals, research books, and refereed international conference proceedings. He is a fellow of the HKIE and the IET.