Received April 4, 2019, accepted May 22, 2019, date of publication June 4, 2019, date of current version June 19, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2920756

Bus Pooling: A Large-Scale Bus Ridesharing Service

KAIJUN LIU[1], JINGWEI ZHANG[1], AND QING YANG[2]
[1]Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China
[2]Guangxi Key Laboratory of Automatic Detection Technology and Instrument, Guilin University of Electronic Technology, Guilin 541004, China

Corresponding author: Qing Yang (gtyqing@hotmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61862013, Grant 61462017, Grant U1501252, Grant U1711263, and Grant 61662015, and in part by the Guangxi Natural Science Foundation of China under Grant 2017GXNSFAA198035 and Grant2018GXNSFAA281199.

ABSTRACT Ridesharing, a shared service that uses the information and knowledge matching, can efficiently utilize scattered social resources to reduce the demand for vehicles in urban road networks. However, car ridesharing has the problems of low capacity and high cost, and it cannot satisfy demands for recurring, long-distance, and low-cost trips. In this paper, we formally define the bus ridesharing problem and propose a large-scale bus ridesharing service to resolve this problem. In our proposed model, the rider can use an online bus-hailing service to upload his or her trip demand and wait to be picked up when it gathers enough people. The provider assigns drivers to riders after integrating the matched ride requests. To maximize ridesharing's success rate, we developed both exact algorithms and approximate algorithms to optimize the ride-matching service. A real-life dataset that contains 65,065-trip instances extracted from 10,585 Shanghai taxis from one day (Apr 1, 2018) is used to demonstrate that our proposed service can provide higher cost performance and on-demand bus services for every ride request. Meanwhile, it reduces the number of vehicles used by 92% and 96% and the amount of oil used by 87% and 92% compared with car ridesharing and no ridesharing, respectively.

INDEX TERMS Ridesharing, bus pooling, capacitated clustering problem, location-allocation problem.

I. INTRODUCTION

At present, the speed of urban traffic infrastructure construction cannot keep up with the rapid growth of traffic demand, especially the traffic jams that occur in cities during rush hours. The traffic problem is gradually being exposed as a weakness of the city. In solving urban traffic problems, ridesharing [1], a shared service that utilizes information and knowledge matching, can effectively use scattered social resources to reduce the demand for vehicles in the urban road network. It is an effective approach to solving the difficulty of taxi-hailing and ease traffic congestion.

Existing ridesharing systems (e.g., Uber, Lyft, Didi, Olacabs) are only focused on car ridesharing, but they have the problems of low capacity and high cost and cannot satisfy demands for recurring, long-distance, and low-cost trips. Assume that millions of commuters living in a metropolis go to work from one district to another every weekday. They have to go back and forth at least once a day and spend a lot of time and money on transportation. Taxi ridesharing to work is convenient and fast, but it is not sustainable for salaried people because it is uneconomical. By contrast, taking a bus is economical, but the time to wait and to pick up/drop off passengers at bus stops is unpredictable, and transfers are a big problem. Carpooling maybe a compromise option, but it's hard to find consistent travel companions.

In this paper, we study the bus ridesharing problem in a practical setting and design a bus ridesharing service system to resolve this problem. To facilitate a better comprehension of our new problem, we start with a demonstration depicted in Fig. 7. The rider can use an online bus-hailing service to upload his or her trip demand, and wait to be picked up by a public bus when it gathers enough people. The provider assigns drivers to riders after integrating the matched ride requests. In addition, the driver's trip must follow the established route with its departure, time-window, capacity, and cost constraints.

To realize this idea, we need to solve the combinatorial optimization. The work of this study consists of three steps as follows: firstly, select a search criterion as needed and

The associate editor coordinating the review of this manuscript and approving it for publication was Roberto Sacile.

solve the capacitated clustering problem [2] of trip demands. Secondly, solve the location-allocation problem [3] of the pickup/delivery point for passengers in each vehicle. Thirdly, prune by using constraints.

To the best of our knowledge, our work is the first to consider bus ridesharing for the problems of recurring, long distance and low-cost trip demands. We place our problem in a practical setting by exploiting a real city's road network and an enormous set of historical taxi trajectory data. The contributions of this paper are multi-dimensional:

- We propose a bus ridesharing service, formally define the problem of bus ridesharing, and, according to the standards of commercial products, develop a bus ridesharing system based on microservice architecture, which can be applied in real life.
- We propose three methods to solve the capacitated clustering problem and propose two methods with performance guarantees to query the global shortest path on road networks. Besides we also include a location-based service to make our results more accurate.
- We define an evaluation benchmark for the bus ridesharing service model and compare the effectiveness and efficiency of the above-mentioned algorithms of our proposed system.
- We perform extensive experiments to validate the effectiveness of bus ridesharing as well as the scalability and efficiency of our proposed bus ridesharing service and compare the time, price, physical exertion, and cost performance of bus pooling, driving, taxi, taxi ridesharing, electric-bike sharing, and bike sharing under the same road conditions.

II. RELATED WORK

In this section, we review previous studies of ridesharing services, including taxi ridesharing, carpooling, and slugging, and analyze the differences between them and bus pooling.

A. TAXI RIDESHARING

Taxi ridesharing is a typical ride-share service that accepts taxi passengers' real-time ride requests sent from smartphones and schedules proper taxis to pick them up via ridesharing. The focuses of current studies are mainly on real-time systems and options for different constraints. For the former, the core is to devise a real-time matching algorithm that can quickly determine the best vehicle to satisfy incoming service requests, e.g., [4]–[7]. For the latter, the studies are reflected in the combination optimization by considering different constraints, such as waiting time, price, route, scheduling, and payment, e.g., [8]–[11]. In contrast, bus pooling offers an affordable mass service rather than personalized and efficient services. Hence, we pay more attention to the effectiveness and scalability of bus pooling.

B. CARPOOLING

Carpooling is an economical ride-share service for passengers who share transportation to the same direction of travel

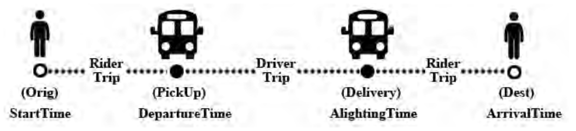


FIGURE 1. The Bus Ridesharing Service Model.

in a private vehicle with other travel companions. As a collaborative model of ridesharing, the hardest part is finding consistent travel companions in a short time. To address this issue, the majority of current studies in the field of carpooling is mainly on requirement mining, e.g., [12]–[15]. Other studies on carpooling focuses on the effectiveness of ridesharing services, such as minimizing the travel costs of vehicles, maximizing the ridesharing's success rate, and improving riders' satisfaction, e.g., [16]–[18]. In contrast, bus pooling is a ride-share service based on demand integration, and its target population are commuters and people taking long-distance trips. The trip demand uploaded by riders are relatively constant and regular.

C. SLUGGING

Slugging is a variation of ride-share commuting and hitchhiking. The slugging problem was proposed in [19]. It assumes that passenger A abandons her trip and joins B's trip (i.e., A's trip is merged into B's trip) and considers a vehicle's capacity constraint to be two to five passengers. From the procedure, slugging is very similar to our proposed model if only one pickup/delivery point is considered in a practical setting. But in fact, a better pickup/delivery point for passengers in our proposed model is designed. Rather than B waiting for A, it is more reasonable to pick the location nearest to both A and B. In addition, we assume a homogeneous fleet of vehicles with a 30-seat capacity (or even more) because our goal is to solve the problem of high capacity.

III. MODEL

A. PROBLEM DEFINITION

1) BUS RIDESHARING PROBLEM

Given a homogeneous fleet of drivers, where each driver is providing ride-share service with an associated capacity limit, and a ride request from a rider, located at an origin to go to a destination with a pair of corresponding expected times. The provider assigns drivers to riders after integrating the matched ride requests. The rider is prompted to walk to the driver's origin, board at the driver's departure time in the time-window, alight at the driver's destination, then walk from there to the rider's own destination (Fig. 1). The objective is to maximize ridesharing's success rate.

2) ASSUMPTION

(1) The driver can arrive at the pickup point on time.
(2) The driver won't have breakdowns or traffic accidents en route.
(3) If the driver arrives at the pickup point earlier than the upper bound of the time window, he or she needs to wait

TABLE 1. **Notations.**

Notation	Definition	Notation	Definition
R	Set of riders	D	Set of drivers
V_o	Set of origins	V_d	Set of destinations
$V\,'$	Other points within the study area (excluding V_o and V_d)	Z	$Z = \{V_o \cup V\,' \cup V_d\}$
$\mid \mathbb{R} \mid$	Number of individuals in set $\mathbb{R}$	$\overrightarrow{AB}$	A path that starts at A and ends at B
R_{trip}	Rider's trip	D_{trip}	Driver's trip
T^1_{r-trip}	Traveling time from the origin to the pickup point	T_{d-trip}	Traveling time from the pickup point to the delivery point
T^2_{r-trip}	Traveling time from the delivery point to the destination	(φ, λ)	Coordinate (φ is latitude, λ is longitude)
$\delta(i, j)$	Shortest distance between point i and point j	$\rho_{i,j}$	Trip similarity measures between trip i and trip j

until the upper bound of the time window unless the last rider be picked up.

(4) The capacity of all drivers can meet the maximum service requirements.

(5) Not every ride request is satisfied since this is subject to the number of riders on a bus.

B. DEFINITIONS

1) RIDE REQUEST

Each ride request r = $\langle Orig, Dest, ScheduledTime, Deadline\rangle$ has an origin *Orig* and a destination *Dest*. *ScheduledTime* denotes the scheduled time of the service proposed by the rider. Note that it is a unilateral intention rather than the scheduled departure time $ScheduledTime^{\triangle}$, which is determined by the scheduled time of all rides on a bus. *Deadline* denotes the latest time of arrival at the destination that the rider can afford.

2) DRIVER

Each driver d = $\langle Capacity, Threshold, EarliestTime, LatestTime\rangle$ has two static values: *Capacity* for the number of seats on a bus and *Threshold* for the number of riders that meet the departure threshold. Each driver has a given time-window when riders are required to arrive; if they are early, they have to wait and if they are late, the driver can refuse to pick them up. The lower bound of the time-window, *EarliestTime*, denotes the time when the driver arrives at the pickup point, which is the earliest start time for the service. The upper bound of the time-window, *LatestTime*, denotes the departure time when the driver leaves the pickup point, which is the latest start time for the service. At this moment, the driver will leave in any case.

3) RIDER TRIP

$R_{trip} = \langle StartTime, PickupTime, AlightingTime, ArrivalTime\rangle$ denotes rider trip information. *StartTime* is the time that the rider leaves the origin. *PickupTime*, *AlightingTime* and *ArrivalTime* are the times that the rider arrives at the pickup point, the delivery point and the destination, i.e.,

$$StartTime + T^1_{r-trip} = PickupTime \tag{1}$$

$$AlightingTime + T^2_{r-trip} = ArrivalTime \tag{2}$$

4) DRIVER TRIP

$D_{trip} = \langle Pickup, Delivery, DepartureTime, AlightingTime\rangle$ denotes driver trip information. *Pickup* is the departure location, and *DepartureTime* is the time of departure. *Delivery* is the terminal location, and *AlightingTime* is the time of alighting, i.e.,

$$DepartureTime + T_{d-trip} = AlightingTime \tag{3}$$

C. MATHEMATICAL FORMULATION

The bus ridesharing problem can be stated as follows:

$$min \sum_{i\in R} \sum_{j\in D} d_{ij}x_{ij} \tag{4}$$

subject to:

$$d_{ij} = \delta(\overrightarrow{Orig_{ij}, Pickup_{ij}}) + \delta(\overrightarrow{Delivery_{ij}, Dest_{ij}}) \tag{5}$$

$$\forall Orig \in V^o, Dest \in V^d, Pickup, Delivery \in V' \tag{6}$$

$$\sum_{j\in D} x_{ij} = 1, \forall i \in R \tag{7}$$

$$x_{ij} \le y_j, \forall i \in R, j \in D \tag{8}$$

$$\sum_{i\in R} y_j \le p, \forall j \in D \tag{9}$$

$$\sum_{i\in R} q_i x_{ij} \le Q_j, \forall j \in D \tag{10}$$

$$x_{ij}, y_j \in \{0, 1\}, \forall i \in R, j \in D \tag{11}$$

where,

- R is the set of riders.
- D is the set of drivers, with $p = |D|$.
- d_{ij} is passenger mileage.
- q_i is the ride request of the rider i.
- Q_j is the maximum capacity of driver j.
- x_{ij} is 1, if the rider i is assigned to driver j and 0 otherwise.
- y_j is 1, if driver j is used and 0 otherwise.

The objective (4) minimizes total passenger mileage, hence increases valid ride requests, with the assurance of maintains the maximum ridesharing's success rate (i.e., the percentage of successful ride requests). Constraint (7) ensures that every rider is assigned to exactly one driver and constraint (8) restricts a rider to driver j, denoted by a binary variable y_j equal to one, to be selected only among the riders assigned to the driver j. Restrictions (9) and (10) ensure that exactly

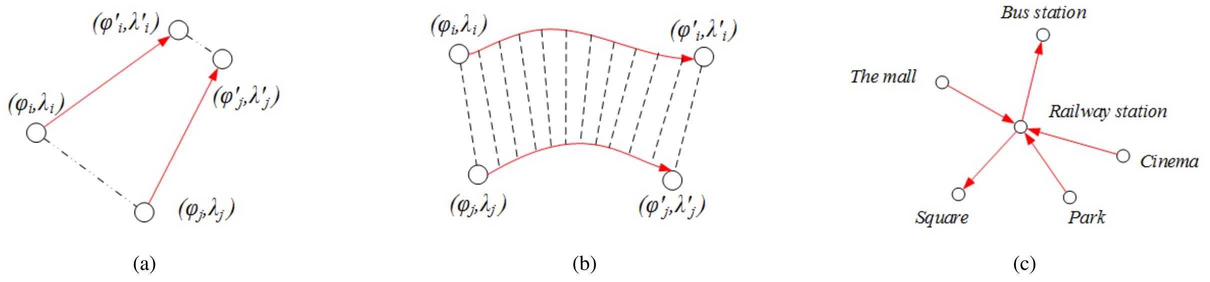


FIGURE 2. Primary Search Criteria. (a) OD-pair and time. (b) Routing and time. (c) Keyword/list.

p drivers will be created and that every driver does not exceed its capacity limit respectively, Constraints (11) specifies the decision variables.

D. CONSTRAINTS

1) DEPARTURE CONSTRAINT

The actual departure time is determined by when the last rider is picked up and the scheduled departure time, i.e.,

$$DepartureTime = min\begin{Bmatrix} max(PickupTime) \\ ScheduledTime^{\triangle} \end{Bmatrix} \tag{12}$$

2) TIME-WINDOW CONSTRAINT

Each rider shall arrive at the pickup point earlier than the departure time and arrive at the destination no later than the deadline time, i.e.,

$$PickupTime \leq DepartureTime \tag{13}$$

$$ArrivalTime \leq Deadline \tag{14}$$

3) CAPACITY CONSTRAINT

On the one hand, each driver can only afford a limited trip demand at any time. On the other hand, in order to reduce operation cost, the driver needs to exceed the threshold of the number of riders to provide services, i.e.,

$$Threshold \leq\mid d \mid\leq Capacity \tag{15}$$

4) COST CONSTRAINT

Each rider has a specific tolerance for passenger mileage. The maximum passenger mileage per rider must be less than a given value *Mileage*, i.e.,

$$\delta(\overrightarrow{Orig, Pickup}) + \delta(\overrightarrow{Delivery, Dest}) \leq Mileage \tag{16}$$

IV. SOLUTION APPROACH

A. PHASE 1: MATCHING AGENCY

The first step of this study is to solve the matching agency problem [20]. Matching agencies use ridesharing offers and requests received from drivers and riders, respectively, to find suitable ridesharing matches. The primary search criteria refer to what information is used by the system to form driver-rider matches. The criteria that can be applied to the bus ridesharing problem are as follows:

1) PRIMARY SEARCH CRITERIA

a: OD-PAIR AND TIME

OD-pair and time matches a request and an offer by the trips' origins, destinations and times. Since each trip has only one origin and only one destination, each trip is denoted as a directed line segment that start from the origin and end at the destination, such as $\overrightarrow{AB}$. For example, two directed line segments are given in Fig. 2(a). Generally, the distance between OD-pairs is regarded as the search criteria on trip routes, which is the distance between (φ_i, λ_i) and (φ_j, λ_j) plus the sum of the distance between (φ'_i, λ'_i) and (φ'_j, λ'_j), i.e.,

$$\rho_{i,j} = \delta[\overrightarrow{(\varphi_i, \lambda_i), (\varphi_j, \lambda_j)}] + \delta[\overrightarrow{(\varphi'_i, \lambda'_i), (\varphi'_j, \lambda'_j)}] \tag{17}$$

However, there may be errors in the real world, e.g., if there is a river or a valley between the two points, even though they are close together, two routes may be completely opposite.

b: ROUTING AND TIME

Routing and time matches the route and the time from the origin to the destination. Essentially, each route is a spatial trajectory represented by a sequence of timestamped geo-coordinates [21]. In order to obtain more accurate results, we calculate the similarity of trajectories to find similar routes. Many existing studies have focused on defining trajectory similarity measures, such as Dynamic Time Warping (DTW) [22], Longest Common Sub-Sequence (LCSS) [23], Edit Distance on Real Sequence (EDR) [24], etc. In this study, we try to take Dynamic Time Warping as a method to measuring dissimilarity that calculates an optimal match between two given time series with certain restrictions and rules because no noise points exist in this study.

c: DYNAMIC TIME WARPING

Dynamic Time Warping is an algorithm which can measure the divergence between two sequences with different phases and lengths. As depicted in Fig. 2(b), it solves this discrepancy between intuition and calculated matching distance by recovering optimal alignments between sample points in the two sequences. Let $Q = \{q_1, \cdots, q_n\}$ and $C = \{c_1, \cdots, c_m\}$ be the two sequences respectively. To align them, we construct an *n*-by-*m* matrix where the

(i^{th}, j^{th}) element of the matrix contains the distance $d(q_i, c_j)$ between the two points q_i and c_j. A warping path $W = \{w_1, \cdots, w_K\}$ that starts at (1, 1) element of the matrix and ends at (m, n), where $K \in [max(m, n), m + n - 1)$, with the least cumulative cost between Q and C from the distance matrix minimizes the total warping cost is defined as follow:

$$DTW(Q, C) = min(\frac{\sqrt{\sum_{k=1}^{K} W_k}}{K}) \tag{18}$$

The cumulative distance $D(i, j)$ as the distance $d(i, j)$ found in the current cell and the minimum of the cumulative distances of the adjacent elements, equation (19) can be used to derive the distance between Q and C based on dynamic programming [25]:

$$D(i, j) = d(q_i, c_j) + min \begin{Bmatrix} D(i, j-1) \\ D(i-1, j) \\ D(i-1, j-1) \end{Bmatrix} \tag{19}$$

d: KEYWORD/LIST

Keyword/list searches a request and an offer by keywords on predefined lists. In practical applications, shuttle buses usually travel from one urban landmark to another, such as a station, park, cinema, square, and so on [Fig. 2(c)]. We put the names of landmarks into keyword lists and measure the distances between these landmarks as the criteria.

2) CAPACITATED CLUSTERING

In subsequent work, we cluster the demands with capacity constraints by using the primary search criteria. Generally, this particular clustering problem is called a capacitated clustering problem.

3) CAPACITATED CLUSTERING PROBLEM

Given a set of weighted individuals is to be partitioned into clusters such that, the total weight of the individuals in each cluster does not exceed a given cluster capacity. The objective is to find a set of centers that minimises the total scatter of individuals allocated to these centers [26].

4) CONTRACTION-BASED METHOD

Our contraction-based method starts by initializing $\mathcal{P}$ and $\mathcal{Q}$ to contain the first θ and the remaining individuals in $\mathcal{R}$, respectively (line 3) and initializing the scatter upper bound *UB* and the cursor variable *cursor* to facilitate pruning in the following enumerations of new candidates (line 5). Then, we insert an unvisited candidate from $\mathcal{Q}$ to $\mathcal{P}$ (line 6), and iteratively examine the scatter of the individuals in $\mathcal{P}$ (line 7-12). The purpose is to measure the dissimilarity between two individuals in $\mathcal{P}$. We also maintain a set $\mathcal{L}$ to contain the complete list of the sum of the scatter of the enumerated candidates. After each iteration of $\mathcal{P}$ is completed, the one who has the maximum scatter in $\mathcal{P}$ is removed (line 13). Scan all the candidates stored in $\mathcal{Q}$ until all the candidates have been visited (line 4); submit the rest as a cluster (line 14) and remove them from $\mathcal{R}$ (line 15). Continue in this way; stop when all the candidates stored in $\mathcal{R}$ have been clustered (line 2).

Algorithm 1 Contraction-Based Method

```
   Input: Requests : $\mathcal{R}$, Capacity : $\theta$
   Output: Clusters : $\mathcal{C}$
1  $\mathcal{C} \leftarrow \varnothing$;
2  while $\mathcal{R} \neq \varnothing$ do
3      $\mathcal{P} \leftarrow \{r_0, \ldots, r_\theta\}$, $\mathcal{Q} \leftarrow \{r_{\theta+1}, \ldots, r_k\}$;
4      for $i = 0; i <| \mathcal{Q} |; i++$ do
5          $UB \leftarrow 0$, $cursor \leftarrow -1$, $\mathcal{L} \leftarrow \varnothing$;
6          $\mathcal{P} \leftarrow \mathcal{P} \cup r_i$;
7          for $j = 0; j <| \mathcal{P} |; j++$ do
8              for $k = 0; k <| \mathcal{P} |; k++$ do
9                  $\mathcal{L}[j] += \rho_{r_j, r_k}$;
10                 if $\mathcal{L}[j] > UB$ then
11                     $UB \leftarrow \mathcal{L}[j]$;
12                     $cursor \leftarrow j$;
13         $\mathcal{P} \leftarrow \mathcal{P}^{r_{cursor}}$;
14     Insert $\mathcal{P}$ into $\mathcal{C}$;
15     Remove $\mathcal{P}$ from $\mathcal{R}$;
16 return $\mathcal{C}$;
```

5) INSERTION-BASED METHOD

The contraction-based method can quickly solve the problem but will lead to the problem of dissimilarity imbalance between clusters because it is easy to fall into the trap of local optima. The candidate may be clustered to the local optimal first and miss the global optimum later. To resolve this problem, we further propose an insertion-based method to fix the problem of "unfairness". It is called insertion-based because it is similar to insertion sort. In each iteration, the first remaining entry of the input is removed and inserted into the result in the correct position, thus extending the result. The first stage of the method is to construct the elite solution. We first examine the maximum number of clusters k (line 1) and insert the top-k pair candidates as the elite solutions Ω into k clusters respectively (line 5) after working out the dissimilarity measures between all candidates (line 2-4). The rest of $\mathcal{R}$ are regarded as candidate solutions (line 6-8). The second stage of the method is insertion, in which we insert an unvisited candidate into each cluster one by one and calculate the value added to the average dissimilarity measure of each cluster (line 9-15). The value added is the standard for evaluating the correspondence between a candidate and clusters, and it may be positive or negative. Using the calculation result, we find the optimal one with the smallest value added and add a corresponding candidate to the corresponding cluster (line 16-17). Finally, stop when we reach the maximum capacity of clusters.

Algorithm 2 Insertion-Based Method

```
   Input: Requests : $\mathcal{R}$, Capacity : $\theta$
   Output: Clusters : $\mathcal{C}$
1  $\mathcal{C} \leftarrow \varnothing, \mathcal{L} \leftarrow \varnothing, n \leftarrow \mid \mathcal{R} \mid, k \leftarrow \lfloor \frac{n}{\theta} \rfloor$;
2  for $r_i \in \mathcal{R}$ do
3      for $r_j \in \mathcal{R}$ do
4          $\mathcal{L} \leftarrow \rho_{r_i,r_j}$;
5  $\Omega \leftarrow$ Select top k from $\mathcal{R}$ order by $\rho$ asc;
6  for $s \in \Omega$ do
7      $\mathcal{C} \leftarrow \mathcal{C} \cup \{s\}$;
8  Remove $\Omega$ from $\mathcal{R}$;
9  while $\mid \mathcal{R} \mid > n - \mid \mathcal{C} \mid \times \theta$ do
10     for $r \in \mathcal{R}$ do
11         for $c \subset \mathcal{C}$ do
12             if $\mid c \mid < \theta$ then
13                 $\varpi \leftarrow c_{weight}$;
14                 $c \leftarrow c \cup \{r\}$;
15                 $added \leftarrow c_{weight} - \varpi$;
16     Insert $r_{min(added)}$ into $c_{\triangle}$;
17     Remove $r_{min(added)}$ from $\mathcal{R}$;
18 return $\mathcal{C}$;
```

Algorithm 3 Dynamic Grid-Based Method

```
   Input: Requests : $\mathcal{R}$, Capacity : $\theta$, Zoom : $\mathcal{Z}$,
          GeoInfo : $lng_{max}$, $lng_{min}$, $lat_{max}$, $lat_{min}$
   Output: Clusters : $\mathcal{C}$
1  $\mathcal{C} \leftarrow \varnothing; c \leftarrow \varnothing, \mathcal{G} \leftarrow \varnothing, \mathcal{M}[\,][\,] \leftarrow \varnothing$;
2  $lng = lng_{max} - lng_{min}$, $lat = lat_{max} - lat_{min}$;
3  $lng_{baseline} = lng_{min}$, $lat_{baseline} = lat_{min}$;
4  for $\mathcal{Z}$ to 0 do
5      for $i \leftarrow 0$ to $\mathcal{Z}$ do
6          for $j \leftarrow 0$ to $\mathcal{Z}$ do
7              $Cell_{lng\_lower} \leftarrow lng_{baseline} + \frac{lng \times (i-1)}{\mathcal{Z}}$;
8              $Cell_{lng\_upper} \leftarrow lng_{baseline} + \frac{lng \times i}{\mathcal{Z}}$;
9              $Cell_{lat\_lower} \leftarrow lat_{baseline} + \frac{lat \times (j-1)}{\mathcal{Z}}$;
10             $Cell_{lat\_upper} \leftarrow lat_{baseline} + \frac{lat \times j}{\mathcal{Z}}$;
11             $\mathcal{G} \leftarrow \mathcal{G} \cup Cell$;
12     for $r \in \mathcal{R}$ do
13         for $Cell \in \mathcal{G}$ do
14             if $r_{orig} \in Cell$ then
15                 $r^k_{orig} \leftarrow Cell^k$;
16             if $r_{dest} \in Cell$ then
17                 $r^k_{dest} \leftarrow Cell^k$;
18     for $r \in \mathcal{R}$ do
19         $\mathcal{M}[r^k_{orig}][r^k_{dest}] += 1$;
20     for $m[i][j] \in \mathcal{M}[\,][\,]$ do
21         if $\mid m[i][j] \mid \geq \theta$ then
22             $\mathcal{S} \leftarrow$ Select all from $\mathcal{R}$ where $r^k_{orig} =$
               $i$ and $r^k_{dest} = j$;
23             for $r_x \in \mathcal{S}$ do
24                 for $r_y \in \mathcal{S}$ do
25                     $\mathcal{L} \leftarrow \mathcal{L} \cup \rho_{r_x,r_y}$;
26             $c \leftarrow$ Select top k from $\mathcal{S}$ order by $\rho$ asc;
27             Insert $c$ into $\mathcal{C}$;
28             Remove $c$ from $\mathcal{R}$;
29 return $\mathcal{C}$;
```

6) DYNAMIC GRID-BASED METHOD

Although the insertion-based method solves the problem of dissimilarity imbalance and calculates the exact solutions, it is not scalable to k because the number of candidates grows exponentially with k. Once k is large, the computation becomes expensive, and the memory cost is unaffordable. We further propose an approximation algorithm named the dynamic grid-based method with performance guarantees to solve the problem. As depicted in Fig. 3 (points and line denote OD-pair and trip respectively), we divide the space into $n \times n$ cells and use cells as spatial units to filter points, where n is a parameter. Each cell is uniquely identified by real numbers $[1, \cdots, n^2]$ called the identity of the cell, which are coefficients corresponding to the left vertical line and top horizontal line of the cell, respectively (line 4-11). To merge and classify for statistics, the origin and the destination of each trajectory is put into the corresponding cells determined by latitude and longitude after the range of each cell has been calculated (line 12-17). As depicted in Fig. 4, an $n^2 \times n^2$ matrix is created to store the number of occurrences from the origins to the destinations of all trajectories (line 18-19). The aim is to quickly count the number of movements between cells. Subsequently, let n keep getting smaller (line 4); the individuals in the cell are regarded as a candidate solution $\mathcal{S}$ once the number of individuals in the cell reaches the threshold θ (line 21). At the same time, the trip from cell i to cell j is considered to have gathered enough riders. Using the trajectory similarity measure, the top-k optimal candidates are selected from the candidate solutions as a cluster and removed from $\mathcal{R}$ (line 22-28). Finally, stop when n equals a given value.

In conclusion, the capacitated clustering problem is solved by the proposed methods. Consequently, we can provide bus ridesharing services to these riders.

B. PHASE 2: THE PICKUP/DELIVERY POINT

1) OVERVIEW

The second step of this study is to solve the location-allocation problem. After capacitated clustering, we prepare to allocate a location for riders be picked up and dropped off with the assurance of maintains the minimum cost. That is, find the pickup/delivery point closest to all the origins/destinations of each cluster. At present, a plethora

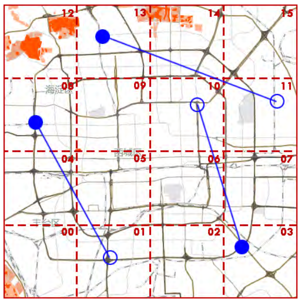


FIGURE 3. Grid Partition.

$$\begin{array}{c|cccccc|}
 & g_1 & g_2 & \cdots\cdots & g_n & \cdots\cdots & g_{2n} \\
g_1 & \varnothing & 0 & \cdots\cdots & 0 & \cdots\cdots & 0 \\
g_2 & 0 & \varnothing & \cdots\cdots & 0 & \cdots\cdots & 0 \\
\vdots & \vdots & & \ddots & & & \vdots \\
g_n & 0 & 0 & \cdots\cdots & \varnothing & \cdots\cdots & 0 \\
\vdots & \vdots & & & & \ddots & \vdots \\
g_{2n} & 0 & 0 & \cdots\cdots & 0 & \cdots\cdots & \varnothing
\end{array}$$

FIGURE 4. Grid Origin-Destination Matrix.

of existing studies have calculated the shortest distance between two locations in a road network. The state-of-the-art approaches can be classified into two categories: spatial coherence based methods and vertex importance based approaches [27]. However, the above-mentioned approaches are both based on a dataset containing an undirected graph that represents a part of the road network. This is different from our situation. To be more universal, we only consider the trajectory data, but do not consider the road network data-set within the study area. In this study, we propose two approximate methods to calculate the global shortest path distances between multiple locations in the road network without the road network dataset. Then, further than that, we still have to consider allowed parking locations in the road network for practical application. For example, urban roads are divided into expressways, arterial roads, secondary trunk roads and branch roads by road classification and function division in China. Among them, only arterial roads, secondary trunk roads and branch roads can support bus stops such as bus bays and request stops. Therefore, the following principles should be met:

1) on an arterial road, secondary trunk road, or branch road;
2) roadside parking or short-term parking is allowed.

a: THE PICKUP/DELIVERY POINT

Given a subset of points and the range of the spatial query in a $d \times d$ rectangle, if a certain point on an arterial road, secondary trunk road, or branch road where roadside parking or short-term parking is allowed connects a subset of points in the rectangle by a set of paths (out of a given set) with minimum cost, the point is defined as the pickup/delivery point (Fig. 5). Assume that there are at least two such points in the rectangle.

Algorithm 4 Random Search

```
Input: $CoordinateSet : \mathcal{L}, Zoom : \mathcal{Z}, Initial\ Points : \mathcal{K}, Iteration : \mathcal{N}, Computations : \mathcal{V}$
Output: $Coordinate : \mathcal{C}$
1  $\mathcal{G} \leftarrow \varnothing, \mathcal{T} \leftarrow \varnothing, \mathcal{L} \leftarrow \varnothing$;
2  $lng_{max} \leftarrow List_{lng\_max}, lng_{min} \leftarrow List_{lng\_min}$;
3  $lat_{max} \leftarrow List_{lat\_max}, lat_{min} \leftarrow List_{lat\_min}$;
4  $lng \leftarrow lng_{max} - lng_{min}, lat \leftarrow lat_{max} - lat_{min}$;
5  $lng_{baseline} \leftarrow lng_{min}, lat_{baseline} \leftarrow lat_{min}$;
6  for $i \leftarrow 0$ to $\mathcal{Z}$ do
7      for $j \leftarrow 0$ to $\mathcal{Z}$ do
8          $Cell_{lng\_lower} \leftarrow lng_{baseline} + \frac{lng \times (i-1)}{\mathcal{Z}}$;
9          $Cell_{lng\_upper} \leftarrow lng_{baseline} + \frac{lng \times i}{\mathcal{Z}}$;
10         $Cell_{lat\_lower} \leftarrow lat_{baseline} + \frac{lat \times (j-1)}{\mathcal{Z}}$;
11         $Cell_{lat\_upper} \leftarrow lat_{baseline} + \frac{lat \times j}{\mathcal{Z}}$;
12         $\mathcal{G} \leftarrow \mathcal{G} \cup Cell$;
13 for $Cell \in \mathcal{G}$ do
14     $\mathcal{T} \leftarrow \mathcal{T} \cup Terminal(Cell, \mathcal{L}, \mathcal{K})$;
15 $\mathcal{G} \leftarrow Evaluate(\mathcal{T}, \mathcal{G})$;
16 for $i \leftarrow 0$ to $\mathcal{N}$ do
17     for $Cell \in \mathcal{G}$ do
18         $\mathcal{L} \leftarrow \mathcal{L} \cup Terminal(Cell, \mathcal{L}, \mathcal{V} \times Cell_{Pr})$;
19         $\mathcal{G} \leftarrow Evaluate(\mathcal{L}, \mathcal{G})$;
20 return $\mathcal{L}_{min}$;
```

Algorithm 5 Terminal($Cell, \mathcal{L}, \mathcal{K}$)

```
1  for $i \leftarrow 0$ to $\mathcal{K}$ do
2      $t_{Coordinate_{lng}} \leftarrow Cell_{lng\_lower} + RandomSeed \times (Cell_{lng\_upper} - Cell_{lng\_lower})$;
       $t_{Coordinate_{lat}} \leftarrow Cell_{lat\_lower} + RandomSeed \times (Cell_{lat\_upper} - Cell_{lat\_lower})$;
3      for $Vertex \in \mathcal{L}$ do
4          $Dist \leftarrow Dist + \delta(t, Vertex)$;
5      $t \leftarrow \langle Cell, Pr, Coordinate, Dist \rangle$;
6  return $t_{min(Dist)}$;
```

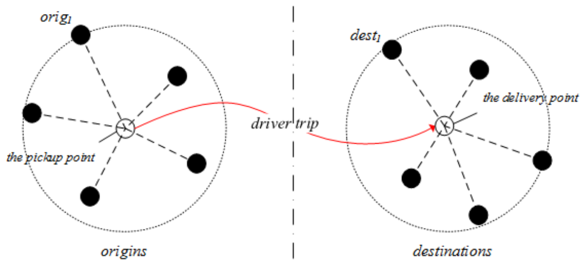


FIGURE 5. The Pickup/Delivery Point.

2) RANDOM SEARCH

In the study of location allocation, the potential location of the pickup/delivery point in the real world is uncertain according to the Fermat-Torricelli problem. Therefore, we plan to calculate an approximate optimal solution based on the divide and conquer principle. We divide the space into $n \times n$ cells and use the cells as spatial units to allocate locations. Each cell is uniquely identified by real numbers $[1, \cdots, n^2]$ called the identity of the cell, which are coefficients corresponding to left vertical line and top horizontal line of the cell, respectively (line 6-12). Subsequently, m random locations are allocated to each cell, and their cost to the origins/destinations of each cluster is calculated (line 13-14), where m is a parameter. Obviously, the bigger the value m, the more accurate the solution, but the higher the computational cost. To reduce computational cost, we compute probability by an evaluation function based on the minimum cost for each cell and put the probability that each cell has the optimal solution into the tabu list as a tabu element (line 15). In this way, we allocate computations to each cell by using the probability (line 18) and iterative solution and keep the tabu list up to date (line 19). Finally, stop when we reach $\mathcal{N}$ iterations in which the optimal solution converges and output the current solution as the optimal solution.

Algorithm 6 Evaluate($\mathcal{L}$, $\mathcal{G}$)

```
1 $sum \leftarrow 0$;
2 for $t \in \mathcal{L}$ do
3     $sum \leftarrow sum + \frac{1}{t_{Dist}}$;
4 for $t \in \mathcal{L}$ do
5     $cell_{Pr} \leftarrow \lfloor \frac{1}{t_{Dist} \times Sum} \rfloor$;
6 return $\mathcal{G}$;
```

The evaluation function can be stated as follows:

$$Pr_i = \frac{\frac{1}{dist\_min_j}}{\frac{1}{dist\_min_1} + \cdots + \frac{1}{dist\_min_j} + \cdots + \frac{1}{dist\_min_n}} \tag{20}$$

subject to:

$$dist_i = \delta(\overrightarrow{Orig, Terminal_i})x_i + \delta(\overrightarrow{Terminal_i, Dest})y_i$$
$$dist\_min_j = min(dist_1, \cdots, dist_m)$$
$$x_i + y_i = 1$$
$$x_i, y_i \in \{0, 1\}$$
$$0 < Pr_i < 1$$
$$\forall i = 0, 1, \cdots, m$$
$$\forall j = 0, 1, \cdots, n^2 - 1$$

where,

$$x_i = \begin{cases} 1 & \text{if the aim is to allocate the pickup point,} \\ 0 & \text{otherwise.} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{if the aim is to allocate the delivery point,} \\ 0 & \text{otherwise.} \end{cases}$$

3) CENTER-BASED SEARCH

Empirically, the pickup/delivery point is the central point of the point set; even if it is not the central point, it also may be near the central point. If we start with the neighborhood search in the central point, it should be accurate and efficient. Consequently, we further propose a center-based search method. To be precise, a random location is allocated from a circle centered on the central point with the radius increasing at a given step size, and stop when a random location is located on the road network that meets the conditions.

Algorithm 7 Center-Based Search

```
  Input: $CoordinateSet : \mathcal{L}$, $Step\ Size : step$
  Output: $Coordinates : t$
1 $\mathcal{P} \leftarrow Query\ a\ central\ point(\mathcal{L})$;
2 $i \leftarrow 0$;
3 while $\mathcal{S} == \varnothing$ do
4     $t \leftarrow Query\ a\ random\ location(\mathcal{P}, i \times step)$;
5     $\mathcal{S} \leftarrow Query\ a\ set\ of\ recommend\ stops(t)$;
6     $i++$;
7 return $\mathcal{S}_{min}$;
```

4) THE RECOMMEND STOPS

In practice, the pickup/delivery points allocated by above-mentioned methods may not allow parking on the map. Therefore, a location-based service (e.g., Google Maps, Bing Maps, Baidu Map, Amap) is used to search the recommended stops near the potential location, whereas this way may return multiple locations. We continue to pick the optimum by calculating the shortest path distances.

C. PHASE 3: PRUNING

To satisfy demands of riders, we prune by using constraints after above-mentioned two steps of this study. The steps are as follows:

Step1 calculate T^1_{r-trip}
Step2 using (1), calculate $PickupTime$.
Step3 using (12), calculate $DepartureTime$.
Step4 using (13), prune.
Step5 calculate T_{d-trip}.
Step6 using (3), calculate $AlightingTime$.
Step7 calculate T^2_{r-trip}.
Step8 using (2), calculate $ArrivalTime$.
Step9 using (14), prune.
Step10 calculate $\delta(\overrightarrow{Orig, Pickup})$ and $\delta(\overrightarrow{Delivery, Dest})$.
Step11 using (15), prune.
Step12 using (16), prune.

V. THE FRAMEWORK

The system design of bus pooling adopts service-oriented architecture (SOA). Fig. 6 shows the framework of the bus ridesharing service. The main components include two business modules (i.e., matching module and terminal module).

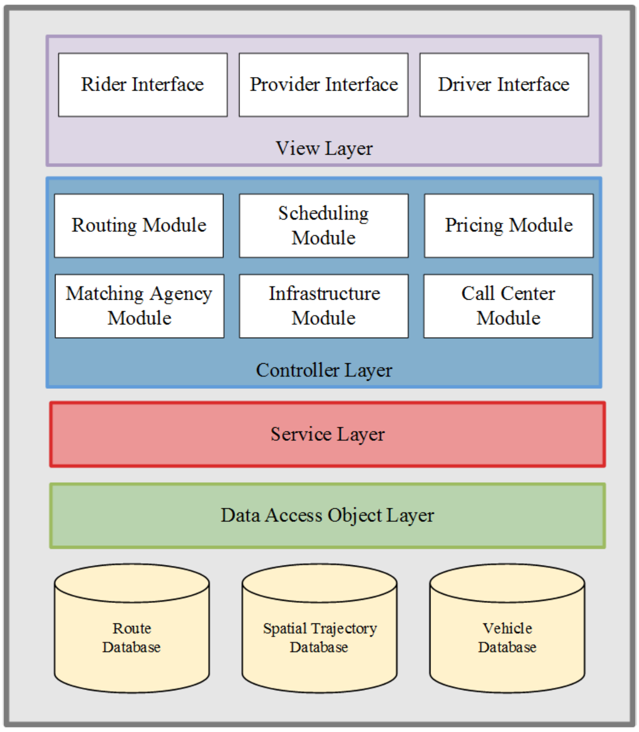


FIGURE 6. **The Framework of the Bus Pooling Service.**

The matching module implements the function of efficient ride-matching and integration of trip demands. The terminal module implements the function of allocating the optimal pickup/delivery point for riders.

The rider sends a ride request to the system by using a smart device; the request is queued and sorted by the scheduled time, then partitioned by time steps. The system batches trip demands of requests based on time steps. Firstly, the matching module analyzes the spatial trajectory of trips and computes the similarity of spatial trajectories to find the similar trips. Secondly, the terminal module allocates the optimal pickup/delivery point for riders by using algorithms and location-based service (to query the duration and distance data of the route). The results are pushed to riders and drivers are notified by a system prompt, and drivers and riders to wait at the designated location in the time window. In addition, any rider can modify or cancel the service before it is completed.

VI. EXPERIMENTS

A. DESIGNED EXPERIMENTS

Four experiments are designed to evaluate the bus ridesharing service:

(1) Compare the effectiveness and efficiency of capacitated clustering algorithms;
(2) Compare the effectiveness and efficiency of location-allocation algorithms;
(3) Evaluate the effectiveness and scalability of bus pooling;
(4) Evaluate the efficiency of bus pooling. This includes two parts: (i) compare the time, price, physical exertion, and cost performance between bus pooling, driving, taxi, taxi ridesharing, electric-bike sharing, and bike sharing; (ii) compare total vehicles, total oil consumption, and total time consumption between bus pooling, driving, and taxi ridesharing.

B. EXPERIMENTAL SETUP

1) ENVIRONMENT

All computational experiments were performed in a Lenovo ThinkStation P318 personal computer with Intel Core i7-7700 CPU that is processor of 3.60GHz, 2400MHz with 8.00G of RAM memory on Microsoft Windows 10 version 1803. The whole implementation were developed in the Java language and has been complied using the Java SE Runtime Environment version 1.8.0.

2) DATASET

We conducted experiments using a taxi GPS trajectory data set. The dataset contains 65,065-trip instances from 10,585 Shanghai taxis from one day (Apr 1, 2018). We take the records of taxi passengers picked up and dropped off that were generated by GPS receivers as the trip demand of riders in a period of time in the whole city. To show an overview, statistics are classified by time steps, total mileage, and demand distribution (Fig. 8). Not only that, we conducted experiments using 100 groups of coordinate sets containing three coordinates as experimental samples to compare the effectiveness and efficiency of location-allocation algorithms. The above-mentioned datasets in the experiment are available at the web page http://dx.doi.org/10.21227/2877-mk46.

C. EVALUATION METRICS

The proposed solution approach groups the goal into two subgoals, the capacitated clustering problem (CCP) and the location-allocation problem (LAP), to solve. Thus, we propose three metrics to evaluate methods for subgoals.

1) MINIMUM SCATTER

The average of scatter between each cluster n individual in p clusters for the capacitated clustering problem. This represents the dissimilarity of ride requests on a bus.

$$\rho = \frac{2 \times \sum_{g=1}^{p} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \rho_{ij}}{p \times n \times (n-1)} \tag{21}$$

2) MINIMUM COST

The total of the shortest path distance from each cluster's pickup/delivery point in p clusters to corresponding cluster's n origins/destinations for the location-allocation problem. The shortest path distance between any two points is used as the cost indicator.

$$d = \sum_{g=1}^{p} \sum_{i=1}^{n} \delta[\overrightarrow{(\varphi_i, \lambda_i), (\varphi', \lambda')}] \tag{22}$$

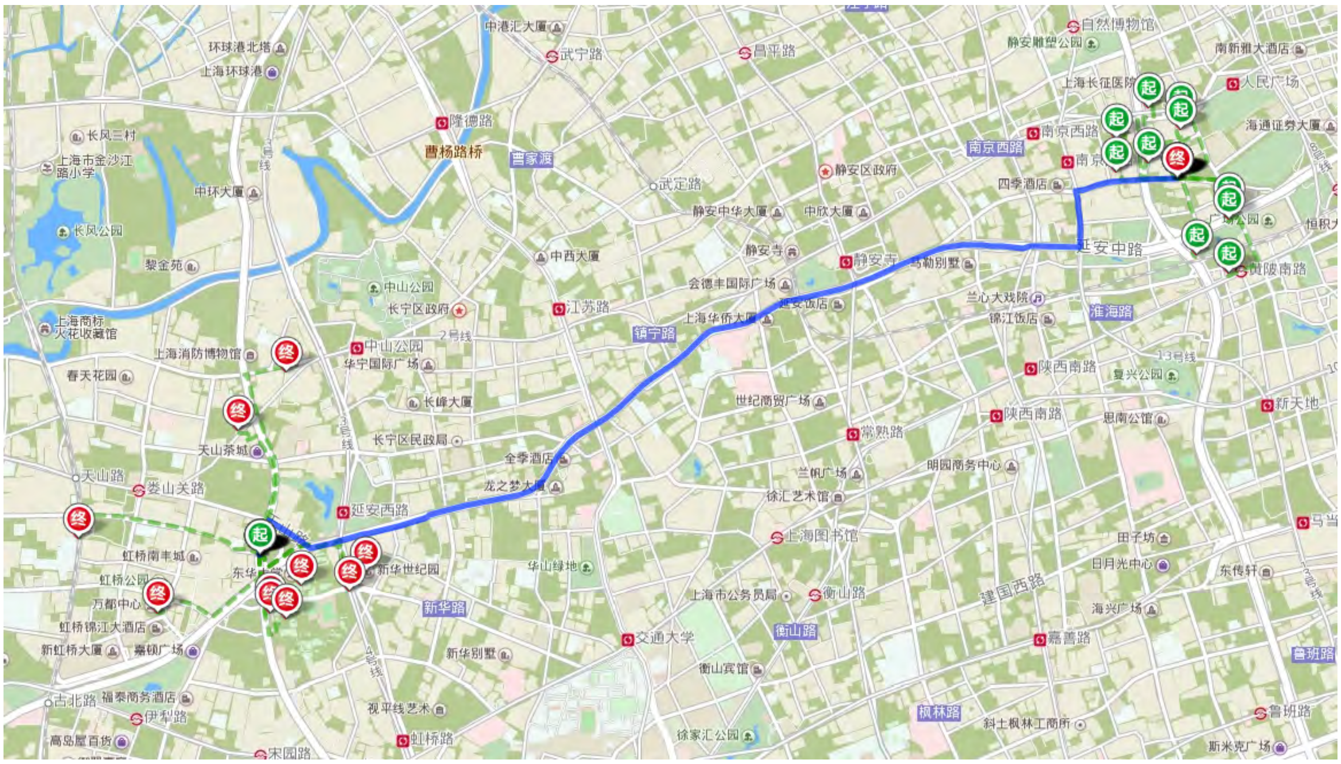


FIGURE 7. A Demonstration of the Bus Ridesharing Service.

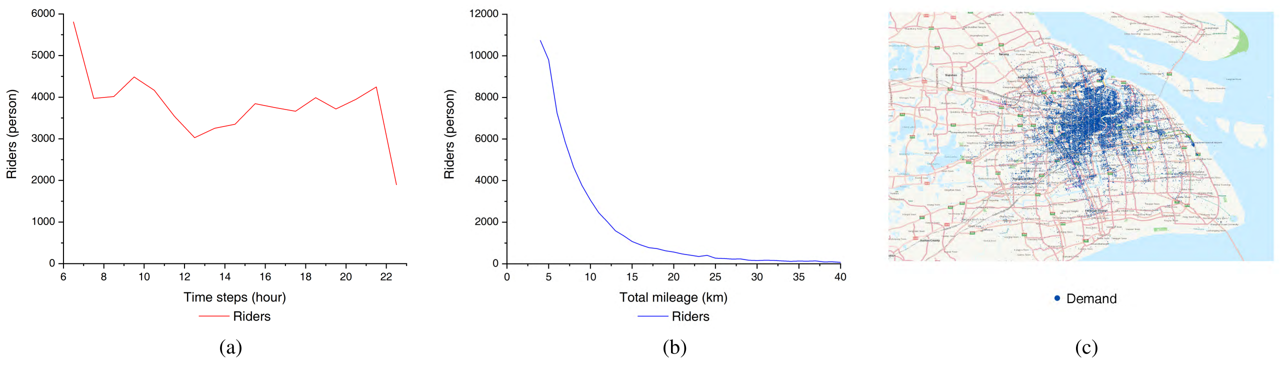


FIGURE 8. Dataset Statistics. (a) Time steps. (b) Total mileage. (c) Demand distribution.

3) TRAFFIC IMPEDANCE MODEL

The traffic impedance model reflects various factors that the traffic users consider when choosing transportation and their importance. Considering the double standard of time and price, it is not accurate to evaluate the bus ridesharing service by using the normal traffic impedance model. Thus, we propose a time-price traffic impedance model as follows:

$$Q = \alpha \times T + (1 - \alpha) \times E \tag{23}$$

where Q is the impedance of the traffic model, T is the time, E is the price, and α is the preference coefficient.

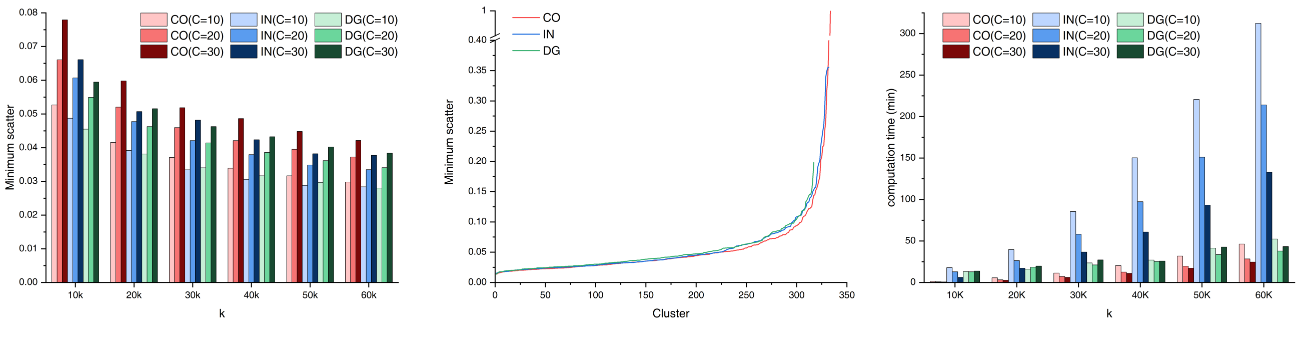
In order to solve the comparability between different standards, we finish the linear transformation of data with min-max normalization. The normalized value is defined as:

$$X^* = \frac{max - x}{max - min} \tag{24}$$

D. EXPERIMENTAL RESULTS

1) CONTRACTION-BASED(CO) VS. INSERTION-BASED(IN) VS. DYNAMIC GRID-BASED(DG)

In the experiment, we selected 10k, 20k, 30k, 40k, 50k and 60k instances respectively by ascending order of departure



(a) (b) (c)

FIGURE 9. Comparing the Effectiveness and Efficiency of Capacitated Clustering Algorithm. (a) Effectiveness (overview). (b) Effectiveness (detail). (c) Efficiency.

TABLE 2. Parameters evaluated in the experiments.

Notation	Definition	Default
θ	Maximum capacity of a driver	30
$threshold$	Total riders of departure threshold	15
$mileage$	Maximum passenger mileage	2 (km)
	Dynamic Grid-Based	
$\mathcal{Z}$	Size of grid cells	185
lng_{max}	Longitude upper bound of the query range	121.98
lng_{min}	Longitude lower bound of the query range	121.05
lat_{max}	Latitude upper bound of the query range	31.5
lat_{min}	Latitude lower bound of the query range	30.7
	Random Search	
$\mathcal{Z}$	Size of grid cells	4
$\mathcal{K}$	Number of random points to compute the initial solution of each cell	2
$\mathcal{N}$	Iterations	3
$\mathcal{V}$	Total number of random points to be distributed per iteration	16
	Center-Based Search	
$step$	Distance increment for the next search	10

time, clustering with $\theta = 10, 20, 30$, and calculated the minimum scatter of the cluster by using (21).

a: EFFECTIVENESS

As shown in Fig. 9(a), DG is the best, IN is the second-best, and CO is the worst. CO is a greed-based exact method that may calculate the optimal solution in a feasible amount of time, but it is easy to fall into the trap of local optima. As shown in Fig. 9(b), CO is better than the other algorithms at first, but eventually, it ends up worse. IN is better than CO because each individual has the opportunity to calculate all clusters. DG is an approximation method based on the divide and conquer principle, which narrows the potential search scope, neglects the bad solutions, and improves the accuracy of the solutions. The advantages of DG could allow it to escape the trap of local optima and finally achieve the global optimum. In terms of k, when k increases, the more combinatorial optimizations are available, and the better the effectiveness. In terms of capacity, the more riders on a bus, the greater the dissimilarity, and the worse the compactness.

b: EFFICIENCY

As shown in Fig. 9(c), the efficiency of CO is obviously better than that of the others. IN is not scalable when k is large. This is because the number of candidates grows exponentially with k, resulting in high computational cost and memory consumption. On the contrary, CO has the simplest steps and the fewest computations, which results in the shortest time. DG can achieve a better tradeoff between efficiency and effectiveness. The computational time of DG is not linearly dependent on k or the number of grid cells but on the solution. The earlier the solution appears, the lower the computational cost. In terms of k, that is as plain as the nose on your face: when k increases, it means that the workload is larger, and the computational cost is higher. In terms of capacity, the greater the capacity, the smaller the number of clusters, and the less the computational cost.

2) RANDOM SEARCH(RS) VS. CENTER-BASED SEARCH(CS)

In the experiment, the sum of global shortest path distances is calculated. The smaller the sum, the better. First of all, we try to figure out the number of iterations in which the algorithm converges to the local optimal solution. As shown in Fig. 10(a), RS converges in four iterations, and CS converges in five iterations. Consequently, the experimental evaluation score was calculated no less than five iterations. On the one hand, the effectiveness of RS is obviously better than that of CS, especially if the distance between the two locations is larger. Not only that, the effectiveness of RS has better stability and less fluctuation [Fig. 10(b)]. On the other hand, RS sacrifices efficiency to improve effectiveness and has a higher computational cost than CS [Fig. 10(c)]. The advantage of RS is randomized adaptive search, which optimizes computation through probability. Although both of them are approximation algorithms, the difference is that RS adopts the method of random sampling and probability gradual approximation, whereas CS infers the location distribution of the optimal solution by taking advantage of prior knowledge, and it gradually approaches the optimal solution from optimal to inferior.

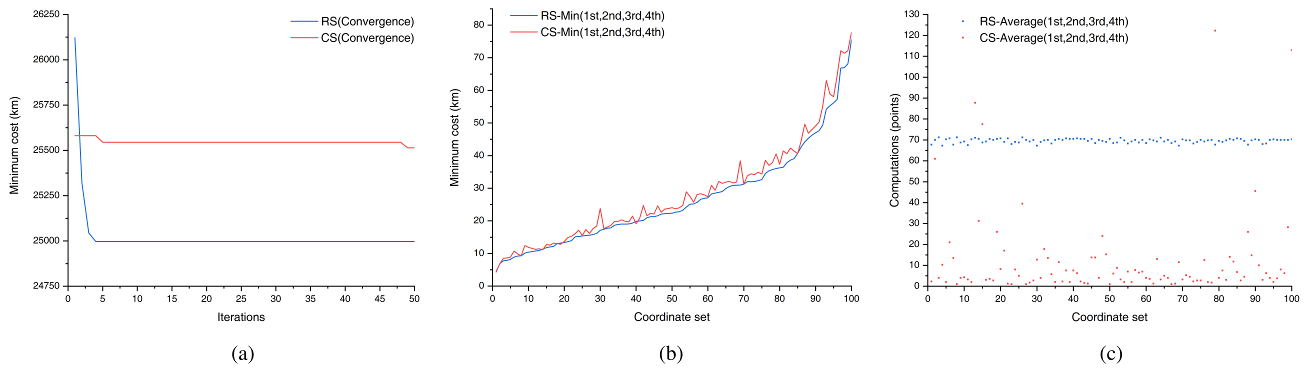


FIGURE 10. **Comparing the Effectiveness and Efficiency of Location-Allocation Algorithm. (a) Convergence. (b) Effectiveness. (c) Efficiency.**

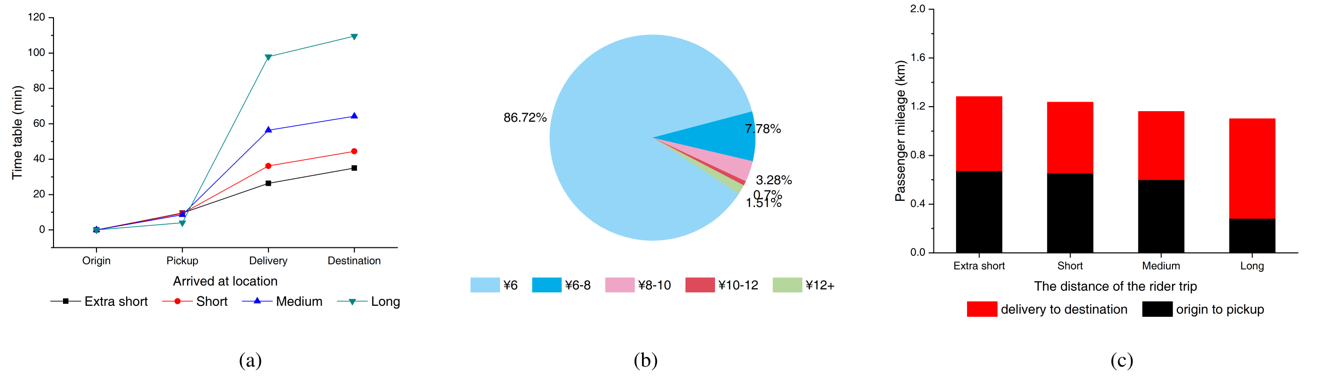


FIGURE 11. **Effectiveness and Scalability of Bus Pooling (a) Currently accepted time. (b) Currently accepted price. (c) Currently accepted passenger mileage.**

3) EFFECTIVENESS AND SCALABILITY OF BUS POOLING

Generally, people are sensitive to price, time, and convenience of trip. Therefore, we analyze the effectiveness of the bus pooling by comparing currently accepted price, time, and passenger mileage. Further, we evaluate the scalability of the bus pooling by examining how the price, time, and passenger mileage change as the mileage of ride requests increases. To distinguish, trips are classified into four categories by mileage: extra-short (0-5km), short (5-10km), medium (10-15km), or long (15km or more).

a: CURRENTLY ACCEPTED TIME

According to a survey conducted by the social survey center of China Youth Daily, 43.4% of respondents spend half an hour to an hour commuting every day. More than 90% of respondents would only accept a maximum commute of less than two hours. As shown in Fig. 11(a), the time taken for extra-short, short, medium, and long distances is 34.97 min, 44.43 min, 64.29 min, and 109.53 min, respectively. All types of commuting time are less than two hours. Meanwhile, it is convenient for passengers not to transfer and to spend no time waiting for and picking up/droping off passengers at bus stops.

b: CURRENTLY ACCEPTED PRICE

As shown in Fig. 11(b), the fare is only ¥6 for 86.72% of riders. The rest are ¥6 to ¥8 (7.78%), ¥8 to ¥10 (3.28%), ¥10 to ¥12 (0.7%), and more than ¥12 (1.51%).

c: CURRENTLY ACCEPTED PASSENGER MILEAGE

The average passenger mileage is only 1.2 km [Fig. 11(c)]. The changes of passenger mileage were not significant for extra-short, short, medium, and long distances. If the speed of walking and cycling are as in Table 3, it only takes 7.2-17 min to reach the destination. We believe that this time is acceptable to most people.

4) EFFICIENCY OF BUS POOLING

Different modes of transport were used to complete all trip demands in the experiment, including bus pooling, driving, taxi, taxi ridesharing, electric-bike sharing, and bike sharing. The purpose was to analyze the efficiency of different modes of transport by comparing time, price, physical exertion, and cost performance.

a: TIME, PRICE, AND PHYSICAL EXERTION

Table 3 gives all the pricing and speeds of transportation in the experiment. Notice that i) all speeds is the average

TABLE 3. Pricing and speeds of transportation in the experiments.

Transportation	Pricing	Mileage	Speed
Driving	¥3/km		20 km/h
Bus Pooling	¥6 ¥0.6/extra per km ¥0.3/extra per km	0km<x≤10km 10km<x≤15km 15km<x	15 km/h
Taxi	¥14 ¥2.4/extra per km ¥3.6/extra per km	0km<x≤3km 3km<x≤10km 10km<x	18 km/h
Taxi Ridesharing	0.737×[¥13+max(¥0,(t-6)×¥0.5)] 0.737×[¥13+(x-3)×¥1.6+max(¥0,(t-6)×¥0.5)] 0.737×[¥13+(20-3)×¥1.6+(x-20)×¥2.4+max(¥0,(t-6)×¥0.5)]	0km<x≤3km 3km<x≤20km 20km<x	18 km/h
Bike Sharing	¥1/15min		10 km/h
Electric-Bike Sharing	¥3/30min		12 km/h
Walking	¥0		4.22 km/h

x: total mileage   t: elapsed time

after taking into account traffic jams and road conditions. ii) the speed of driving is faster than taxi and taxi ridesharing because it saves time waiting for a ride. iii) using a pre-booked ride-hailing service saves more time than taking a taxi in practice. iv) electric-bike sharing cannot satisfy demands for long distance trip because of the limitation of battery capacity. In terms of time consumption, bike sharing is the most time-consuming, followed by bus pooling, electric-bike sharing, taxi, taxi ridesharing and driving [Fig. 12(a)]. To be precise, driving, taxi ridesharing, and taxi take 55%, 63%, and 69% of the time of bus pooling respectively in extra-short, short, medium, and long distances. On the opposite, prices of taxi, driving, and taxi ridesharing are 4.93 times, 4.49 times, and 4.15 times that of bus pooling respectively for extra-short, short, medium and long distances. On price alone, bike sharing, electric-bike sharing, and bus pooling are cheaper [Fig. 12(b)]. The numerical result indicates that driving, taxi, and taxi ridesharing are convenient and fast but cannot satisfy demands for recurring, long-distance, and low-cost trips. In terms of physical exertion, bike sharing consumes more strength. On average, bike sharing consumes 9.69 times as much energy per trip as bus pooling [Fig. 12(c)]. Obviously, cycling all the way from the origin to the destination is more tiring.

b: TRAFFIC IMPEDANCE MODEL

It is well known that price and quality are directly proportional. Our purpose is not to request the best transportation but to find suitable transportation for people with different preferences. Equation (23) shows that $Q \in (0, 1)$ is the evaluation score, the higher the better. $\alpha = 0.25, 0.75, 0.5$ indicates that the passenger is price-sensitive, is time-sensitive, and treats price and time as equal, respectively. Fig. 12(d) illustrates that driving and taxi ridesharing are more appropriate for time-sensitive passengers. Fig. 12(e) illustrates that electric-bike sharing is best for extra-short, short, and medium distances and bus pooling is more cost-effective than others for long distance, when time and price criteria are equal. Fig. 12(f) illustrates that bus pooling is the best choice for people who seek long-distance and low-cost trip. In addition, we found with the increase of distance of ride requests, the cost performance of bus pooling increases gradually, whereas electric-bike sharing is on the contrary. This highlights their respective advantages. Compared to driving and taxi ridesharing, the cost performance of taxis remains low.

c: TOTAL VEHICLES(TV), TOTAL OIL CONSUMPTION(TOC), AND TOTAL TIME CONSUMPTION(TTC)

At last, the social cost and social benefit caused by applying bus pooling are assessed from vehicles used, oil consumption and time consumption. The total number of vehicles for driving and taxi ridesharing is 22.75 times and 14.27 times that of bus pooling, respectively [Fig. 12(g)]. Assume that the oil consumption of a bus is 17.1 XL/100KM and a car 9.12 XL/100KM, then the total oil consumption of driving and taxi ridesharing is 12.84 times and 7.49 times that of bus pooling respectively [Fig. 12(h)]. To be precise, bus pooling reduces the number of vehicles used by 92% and 96% and the amount of oil used by 87% and 92% compared with taxi ridesharing and no-ridesharing, respectively. The numerical result indicates that bus pooling can make full use of resources and is very energy-efficient and economical. On the contrary, driving achieves at most 49% savings of all passengers' time [Fig. 12(i)]. In addition, from the trend of the ratio of bus pooling and driving, the increase of the distance of rider requests leads to the gradual decline of ridesharings success rate.

VII. POTENTIAL FUTURE DIRECTION

In this section, we propose the roadmap of bus pooling research. Generally, a new service needs to undergoes several stages of evolution, such as basic service, service support, value-added service and knowledge discovery. In the study of ridesharing, whether it is car ridesharing or bus ridesharing, ridesharing service is to assume the role of the media. The core is to optimize a ride-matching problem, that is, to match the most suitable supplier for the demander and to match the most suitable demander for the supplier. The new challenge of bus ridesharing service calls for i) route (or trip) generation for each bus to perform multiple

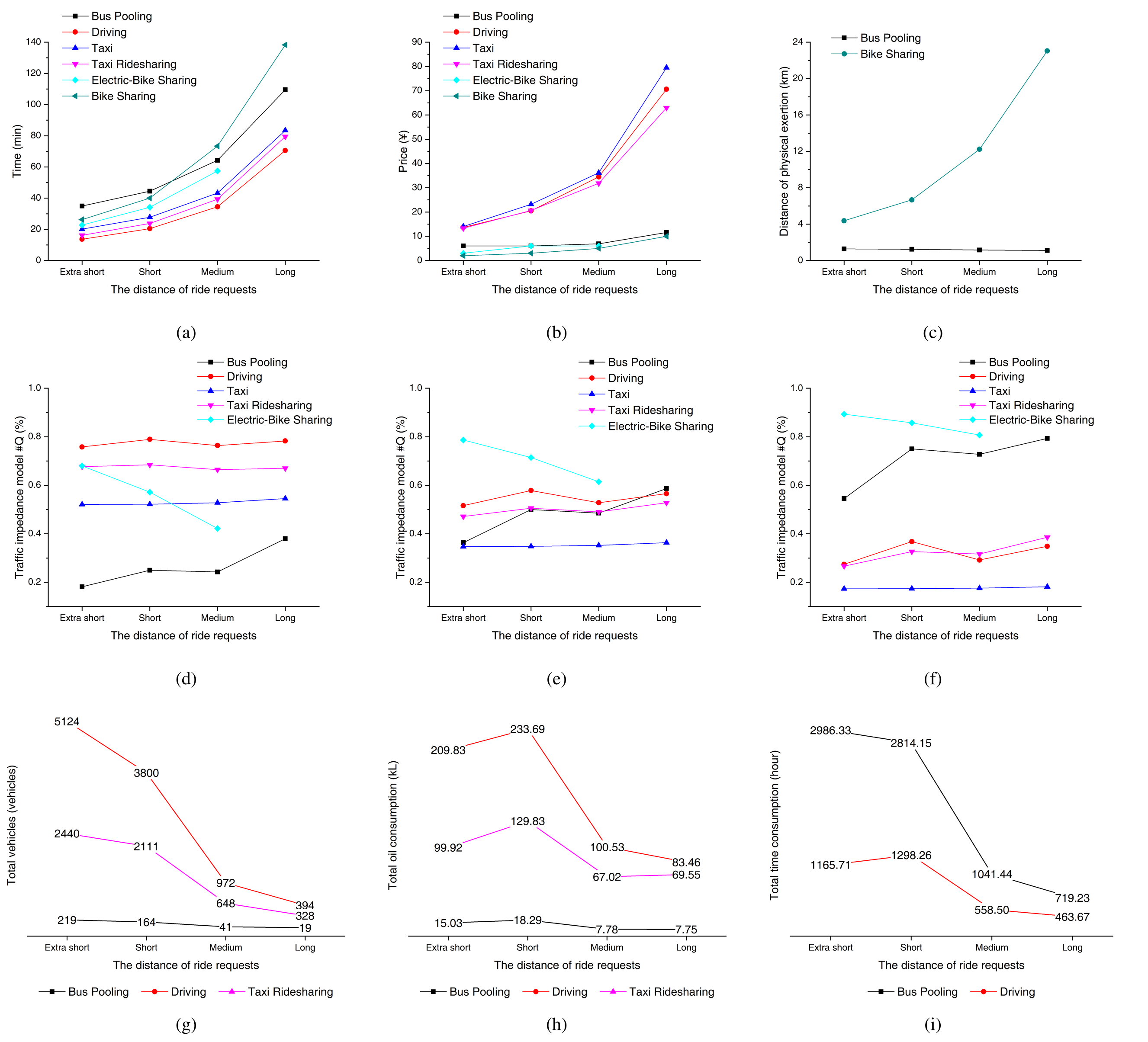


FIGURE 12. Efficiency of Bus Pooling. (a) Time. (b) Price. (c) Physical exertion. (d) #Q ($\alpha = 0:25$). (e) #Q ($\alpha = 0:5$). (f) #Q ($\alpha = 0:75$). (g) TV that carry the same number of people. (h) TOC that carry the same number of people. (i) TTC that carry the same number of people.

transports, multiple pickup/delivery points selection, vehicle and crew scheduling, crew rostering, depot location, emergency processing and decision that can provide support and guarantee for business management and operations. ii) demands for response to services (real-time or non-real-time) and combination optimization of constraints (such as walking distance, driving time, waiting time, price, route, etc.) that can provide more diverse services to satisfy the personal demands of different target markets. iii) route prediction and mining base on historical data that can discover potentially knowledge and open up scenarios.

VIII. CONCLUSION

In this paper, we proposed a large-scale bus ridesharing service: it allows riders to customise the bus route on demand and pick up and drop off at any desired location. This study focused on the ride-matching optimization problem, which deals with how to find suitable ridesharing matches. The proposed solution approach groups the goal into two subgoals, the capacitated clustering problem (CCP) and the location-allocation problem (LAP), to solve. The former is the clustering of travel demand. Due to the limited number of seats on a bus, each cluster has a certain capacity limit. The latter is to allocate a location for a set of riders be picked up and dropped off with the assurance of maintains the minimum cost. The experimental results show that our proposed service is economical, energy-efficient, higher in cost performance than driving, taxi, taxi ridesharing, electric-bike sharing, and bike sharing. It provides decision support for government authorities to manage urban bus planning and commercial operation of public transportation corporations.

REFERENCES

[1] E. Ferguson, "The rise and fall of the American carpool: 1970–1990," *Transportation*, vol. 24, pp. 349–376, Nov. 1997.

[2] J. M. Mulvey and M. P. Beck, "Solving capacitated clustering problems," *Eur. J. Oper. Res.*, vol. 18, pp. 339–348, Dec. 1984.

[3] L. Cooper, "Location-allocation problems," *Oper. Res.*, vol. 11, no. 3, pp. 331–343, 1963.

[4] S. Ma, Y. Zheng, and O. Wolfson, "T-share: A large-scale dynamic taxi ridesharing service," in *Proc. IEEE 29th Int. Conf. Data Eng. (ICDE)*, Apr. 2013, pp. 410–421.

[5] C. Tian, Y. Huang, Z. Liu, F. Bastani, and R. Jin, "Noah: A dynamic ridesharing system," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2013, pp. 985–988.

[6] N. Masoud and R. Jayakrishnan, "A decomposition algorithm to solve the multi-hop peer-to-peer ride-matching problem," *Transp. Res. B, Methodol.*, vol. 99, pp. 1–29, May 2017.

[7] P. Ehsani and J. Y. Yu, "The merits of sharing a ride," in *Proc. Annu. Allerton Conf. Commun., Control, Comput.*, Oct. 2017, pp. 776–782.

[8] B. Cao, L. Alarabi, M. F. Mokbel, and A. Basalamah, "SHAREK: A scalable dynamic ride sharing system," in *Proc. IEEE Int. Conf. Mobile Data Manage.*, Jun. 2015, pp. 4–13.

[9] X. Duan, C. Jin, X. Wang, A. Zhou, and K. Yue, "Real-time personalized taxi-sharing," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, vol. 9643, Apr. 2016, pp. 451–465.

[10] L. Chen, Y. Gao, Z. Liu, X. Xiao, C. Jensen, and Y. Zhu, "PTrider: A price-and-time-aware ridesharing system," *Proc. VLDB Endowment*, vol. 11, pp. 1938–1941, Aug. 2018.

[11] Y. Huang, R. Jin, F. Bastani, and X. S. Wang, "Large scale real-time ridesharing with service guarantee on road networks," *Proc. VLDB Endowment*, vol. 7, pp. 2017–2028, Oct. 2013.

[12] Y. Fu, Y. Fang, C. Jiang, and J. Cheng, "Dynamic ride sharing community service on traffic information grid," in *Proc. Int. Conf. Intell. Comput. Technol. Automat. (ICICTA)*, vol. 2, Oct. 2008, pp. 348–352.

[13] X. Xing, T. Warden, T. Nicolai, and O. Herzog, "SMIZE: A spontaneous ride-sharing system for individual urban transit," in *Proc. German Conf. Multiagent Syst. Technol.*, Sep. 2009, pp. 165–176.

[14] V. Armant and K. N. Brown, "Minimizing the driving distance in ride sharing systems," in *Proc. IEEE 26th Int. Conf. Tools Artif. Intell.*, Nov. 2014, pp. 568–575.

[15] R. S. Thangaraj, K. Mukherjee, G. Raravi, A. Metrewar, N. Annamaneni, and K. Chattopadhyay, "Xhare-a-Ride: A search optimized dynamic ride sharing system with approximation guarantee," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 1117–1128.

[16] S.-C. Huang, M.-K. Jiau, and K.-H. Chong, "A heuristic multi-objective optimization algorithm for solving the carpool services problem featuring high-occupancy-vehicle itineraries," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 8, pp. 2663–2674, Aug. 2017.

[17] N. Ta, G. Li, T. Zhao, J. Feng, H. Ma, and Z. Gong, "An efficient ride-sharing framework for maximizing shared route," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 2, pp. 219–233, Feb. 2017.

[18] P. Cheng, H. Xin, and L. Chen, "Utility-aware ridesharing on road networks," in *Proc. ACM Int. Conf. Manage. Data*, May 2017, pp. 1197–1210.

[19] S. Ma and O. Wolfson, "Analysis and evaluation of the slugging form of ridesharing," in *Proc. ACM Int. Symp. Adv. Geograph. Inf. Syst.*, Nov. 2013, pp. 64–73.

[20] M. Furuhata, M. Dessouky, F. Ordóóñez, M.-E. Brunet, X. Wang, and S. Koenig, "Ridesharing: The state-of-the-art and future directions," *Transp. Res. B, Methodol.*, vol. 57, pp. 28–46, Nov. 2013.

[21] Y. Zheng, "Trajectory data mining: An overview," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 3, May 2015, Art. no. 29.

[22] B.-K. Yi, H. V. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *Proc. 14th Int. Conf. Data Eng. (ICDE)*, Feb. 1998, pp. 201–208.

[23] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proc. 18th Int. Conf. Data Eng. (ICDE)*, Washington, DC, USA, Feb./Mar. 2002, pp. 673–684.

[24] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2005, pp. 491–502.

[25] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in *Proc. 1st SIAM Int. Conf. Data Mining*, vol. 1, Jan. 2002, pp. 1–11.

[26] S. Ahmadi and I. H. Osman, "Greedy random adaptive memory programming search for the capacitated clustering problem," *Eur. J. Oper. Res.*, vol. 162, no. 1, pp. 30–44, 2005.

[27] L. Wu, X. Xiao, D. Deng, G. Cong, A. D. Zhu, and S. Zhou, "Shortest path and distance queries on road networks: An experimental evaluation," *Proc. VLDB Endowment*, vol. 5, no. 5, pp. 406–417, Jan. 2012.

KAIJUN LIU received the bachelor's degree from Sichuan Normal University, China, in 2010. He is currently pursuing the Ph.D. degree with the Guilin University of Electronic Technology, China. His research interests include spatial data management, systems analysis and design, and software architecture.

JINGWEI ZHANG received the Ph.D. degree from East China Normal University, China, in 2012. He is currently a Professor with the School of Computer and Information Security, Guilin University of Electronic Technology, China. His research interests include massive data management, distributed computing frameworks, Web data analysis, and big data analytics for emerging applications.

QING YANG is currently an Associate Professor with the School of Electronics Engineering and Automation, Guilin University of Electronic Technology, China. Her research interests include intelligent information processing, social network analysis, and large-scale data processing optimization.