An Online Ride-Sharing Path-Planning Strategy for Public Vehicle Systems

Ming Zhu, Xiao-Yang Liu, and Xiaodong Wang, *Fellow, IEEE*

***Abstract*— As efficient traffic-management platforms, public vehicle (PV) systems are envisioned to be a promising approach to solving traffic congestion and pollution for future smart cities. PV systems provide online/dynamic peer-to-peer ride-sharing services with the goal of serving a sufficient number of customers with a minimum number of vehicles and the lowest possible cost. A key component of the PV system is the online ride-sharing scheduling strategy. In this paper, an efficient path-planning strategy based on a greedy algorithm is proposed, which focuses on a limited potential search area for each vehicle by filtering out the requests that violate the passenger service quality level, so that the global search is reduced to a local search. Moreover, the proposed heuristic can be easily used in the future globally optimal algorithm (if it will exist) to speed the computation time. The performance of the proposed solution, such as reduction ratio of computational complexity, is analyzed. Simulations based on the Manhattan taxi data set show that the computing time is reduced by 22% compared with the exhaustive search method under the same service quality performance.**

***Index Terms*— Path planning problem, potential search area, public vehicle systems, online/dynamic peer-to-peer ride-sharing.**

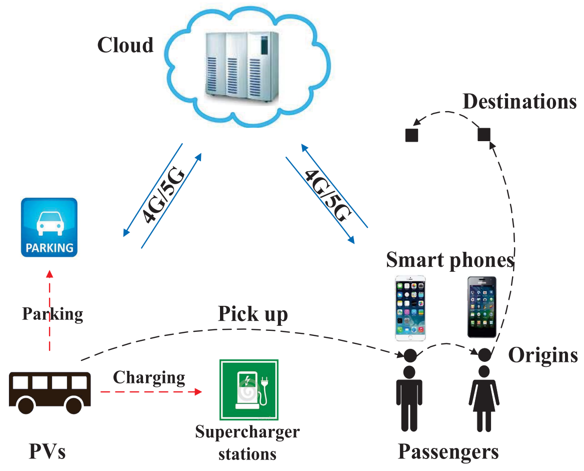


Fig. 1. Architecture of PV systems.

I. Introduction

EXISTING transportation systems are not satisfying since the slowly increasing road capacity can not sufficiently serve the rapidly increasing of cars, leading to high congestions, serious pollutions and potential health problems with large investments [1]. We hope that transportation policies and programs can serve economic, social, and environmental goals. Several research works show that the total industrial and consumer expenditure on transportation is about 10% of GDP (gross domestic product) in the world [2]. In the USA, around 16.7% of the household income is spent on transportation [3]. Besides the large investments, many countries are suffering from social problems such as traffic congestions [4] and fuel pollution-related diseases [5] brought by transportation. The vehicle fuel pollution accounts for 31% of all pollution in Beijing [6], and the Beijing city government has spent $277 billion during 2011-2013 [6] on air pollution to reduce the PM2.5 (particulate matter with diameter of 2.5 micrometers or less) [7]. To address the above problems, ride-sharing is a promising solution that is cost-effective.

As an application of sharing economy [8], public vehicle (PV) systems [9]–[12] provide low-cost peer-to-peer ride-sharing trips with ensured Quality of Service (QoS) for passengers. As shown in Fig. 1, a PV system consists of three parts: a cloud, PVs, and passengers. The blue solid lines denote the communications between them, and the dash black lines imply the scheduling task of a PV. In addition, future PVs are envisioned to be self-driving electric vehicles, therefore charging [13] is an important issue.

The operation of a PV system is as follows. If a passenger needs a trip service, he/she sends a request (including the earliest start time, the origin, and the destination, etc.) through a mobile internet device (e.g., smart phone) to a cloud. The cloud computes a path plan with a confirmed ride match to serve him/her. He/She can access the information of the confirmed PV through apps, e.g., vehicle ID, position, speed, and path. PVs serve passengers by traversing from origins

Manuscript received November 29, 2017; revised February 20, 2018; accepted March 18, 2018. Date of publication May 21, 2018; date of current version January 31, 2019. This work was supported in part by the Leading Talents Program of Guangdong Province under Grant 00201510, in part by the Basic Research Program of Shenzhen under Grant JCYJ20151117161854942, and in part by the Shenzhen Peacock Program under Grant KQTD2015071715073798. The Associate Editor for this paper was P. Kachroo. *(Corresponding author: Xiao-Yang Liu.)*

M. Zhu is with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China, and also with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: zhumingpassional@gmail.com; zhumingpassional@sjtu.edu.cn).

X.-Y. Liu is with the Electrical Engineering Department, Columbia University, New York, NY 10027 USA, and also with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: xiaoyang@ee.columbia.edu).

X. Wang is with the Department of Electrical Engineering, Columbia University, New York, NY 10027 USA (e-mail: wangx@ee.columbia.edu).

Digital Object Identifier 10.1109/TITS.2018.2821003

(pick-up points) to destinations (drop-off points). Compared with conventional public transportation systems such as buses and subways which also provide ride-sharing services, there is no last mile problem in PV systems, and PVs are more flexible since their paths are adapted to the trip demands of passengers, while the paths and the schedules of buses and subways are fixed.

PV systems are also different from existing ride-sharing systems (e.g., Uber Pool [14] and Didi [15]). There are several important differences. 1) PV systems are centralized systems which provide online/dynamic peer-to-peer ride-sharing services, while existing ride-sharing systems such as Uber Pool are distributed offline/static systems. 2) In PV systems, scheduling strategies are calculated by the cloud, while in existing ride-sharing systems, scheduling strategies are negotiated by drivers and riders. 3) PVs cooperate with each other to improve the traffic efficiency, e.g., by providing multi-hop ride-sharing paths [16], while in existing ride-sharing systems, drivers compete for more profit.

This paper investigates an efficient ride-sharing path planning problem in PV systems. The existing solutions [9], [17], [18] become inefficient since most of them are based on exhaustive search, and only some of them [19] consider computational efficiency while the QoS can not be guaranteed. This paper studies this problem in a practical way by exploiting QoS constraints and the geometry, and then a local search solution is proposed.

The challenges of our problem are as follows. 1) There is a trade-off between the objectives of passengers and PVs: PVs try to serve more passengers with the minimum energy cost or travel distance; where passengers want to arrive at their destinations as early as possible with the lowest cost. 2) The online/dynamic ride-sharing in PV systems involves multiple passengers' utilities. If the utility of a passenger is compromised by serving some other passengers, e.g., a long detour, he/she would choose other type of service such as taxi. 3) When the cloud receives new trip requests, new schedules will be calculated and paths of some PVs may change.

The contributions of this paper are given as follows:

- To the best of our knowledge, this is the first work that takes into account both the computational efficiency and QoS guarantee in online/dynamic ride-sharing for PV systems aiming at reducing the travel distance of vehicles.
- An efficient path planning strategy is proposed by restricting the search areas for PVs, reducing the global search to a local search. It is suitable for real-time implementation.
- The performance of the proposed solution is analyzed, e.g., the larger is the area of the city, the better performance (reduction ratio of computational complexity) the proposed solution will have.
- The strategy balances the utilities of passengers and PVs, i.e., providing high QoS (e.g., short waiting time, and less detour) for passengers with the lowest energy cost.
- Simulations with the trip requests based on the Manhattan taxi data set are performed to evaluate the proposed strategy. A large amount of computation can be saved. The travel distance of PVs is reduced, and the ride-sharing for passengers can be guaranteed compared with privately owned electric vehicles.

Related work on ride-sharing path planning problems in PV or PV-like systems is described in Section II. In Section III, the problem formulation is presented. Section IV and Section V present the solution and its performance analysis, respectively. Section VI provides simulation results. Section VII presents some discussions. And finally Section VIII concludes this paper.

II. Related Work

Some ride-sharing path planning strategies are restricted to some special cases such as common origins or destinations. In particular, Massobrio *et al.* [20] propose evolutionary algorithms to solve the one-origin-multi-destination taxi-sharing problem, where the QoS metrics include the total trip cost and the time delay of passengers. Naoum-Sawaya *et al.* [21] present a stochastic mixed integer programming model to optimize the allocation of cars to employees (from homes to work places) while taking into account the unforeseen events of car unavailability. It only focuses on ride-sharing with common destinations in large organizations, e.g., companies, hospitals, and universities. Shang *et al.* [17] propose a collective travel planning query to find the lowest cost path connecting multiple origins and a destination with limited number of meeting points. However, all the above solutions can not be used in the multi-origin-multi-destination scenario, which is more common in the real world.

Most of the current ride-sharing path planning strategies in the multi-origin-multi-destination scenario are exhaustive search methods, which incur high computational load at the cloud. Zhu *et al.* [9] propose an algorithm to reduce travel distance of vehicles with QoS constraints for passengers, e.g., detour, which needs to try each request and calculate corresponding detour constraints. Goel *et al.* [22] propose a solution which selects the optimal fixed positions of pick-up points to maximize the vehicle occupancy rates while preserving the passenger privacy and safety, where passengers do not need to provide their precise home or work positions. To maximize the vehicle occupancy and minimize the travel time with limited detour, Jung *et al.* [18] propose a hybrid-simulated annealing to dynamically assign passenger requests for online/dynamic ride-sharing, which is computationally expensive especially when the number of requests is large since multiple random perturbations and a large number of iterations are needed.

Less research works focus on how to reduce the computational complexity and restrict the search areas in online/dynamic ride-sharing in PV or PV-like systems [19]. To reduce the total travel distance of taxis, Ma *et al.* [19] propose an efficient ride-sharing path planning solution in serving dynamic queries, where "lazy shortest path" calculation is used by means of partitioning the whole road network into multiple grids, and the status of each vehicle should be updated according to preset intervals. This work only considers how to reduce the computational complexity based on the current distance between vehicles and origins/destinations of requests.

Ota *et al.* [23] propose a real-time and data-driven simulation framework to achieve the efficient analysis of taxi ride sharing by means of exploring parallelism and cache-coherent shortest path index, which also considers different stakeholders' interests and constraints, e.g., the waiting time, the maximum number of additional stops, and the maximum number of shared trips.

In summary, most online/dynamic ride-sharing solutions in PV or PV-like systems are based on computationally expensive exhaustive search. Only a few solutions focus on reducing the computational complexity [19], [23]. However, the road network should be divided into cells and the states of vehicles should be frequently updated, making the methods complicated. The proposed solution in this paper is more efficient, easy to implement, and can be used in large cities. Moreover, the QoS level of passengers can be guaranteed, e.g., short waiting time and less detour.

III. PV Path Problem

In this section, first the basic concepts for our problem are presented, and then the problem formulation and NP-completeness are presented.

A. Preliminaries

Assume that all PVs in a city constitute a set $\mathcal{P}$, and all requests constitute a set $\mathcal{R}$. Let $\mathcal{R}_u$ be a set of unscheduled requests, and $\mathcal{R}_s$ be a set of scheduled requests. Assume that once the ride match between any request r and PVs has been confirmed by the cloud, it does not change. Clearly, $\mathcal{R} = \mathcal{R}_u \bigcup \mathcal{R}_s$. Moreover, let $\mathcal{R}_s = \mathcal{R}_{s,1} \bigcup \mathcal{R}_{s,2}$, where $\mathcal{R}_{s,1}$ is a set of requests being served (has been picked up yet not dropped off) by PVs and $\mathcal{R}_{s,2}$ is the set of requests waiting to be served (the ride match has been confirmed yet not picked up).

The PV path (PVP) problem is an online ride-sharing path planning problem, where each request $r \in \mathcal{R}$ should be served by a corresponding PV, while on the path of any PV, it can serve other unscheduled requests if there exist available seats. Let $r = (n, t, o, d) \in \mathcal{R}$ denote a trip request, where n is the number of passengers, t is the earliest start time, o is the origin (pick-up point), and d is the destination (drop-off point). Assume that the passengers using the same request should be served together by a PV. For request $r_1 \in \mathcal{R}_{s,1}$, it has been picked up so that its origin is not important anymore and only its destination should be reached. However, for request $r_2 \in \mathcal{R}_{s,2}$, both the origin and the destination should be reached with the origin preceding the destination. Let $p \in \mathcal{P}$ denote a PV or its current position.

Next, to describe our problem clearly, three definitions are introduced: schedule, service list, and path. And then an example is presented to discuss their changes in the execution of a path planning strategy in PV systems.

Definition 1: The schedule for any request $r = (n, t, o, d)$, means that if a PV p is scheduled to r with the earliest start time t, p will transverse through the origin o and the destination d with o preceding d, and meanwhile p will pick up r (n passengers) at o, and drop off r at d.

In Definition 1, a schedule mainly determines the ride match between the request and PVs, and the precedence constraints between the origin and the destination. The schedule for a request makes sure that this request will be transported from the origin to the destination.

Definition 2: The service list of any PV p, denoted by L_p, is a list of requests p has to serve, including the requests being served (have been picked up by p, yet have not arrived at their destinations), and the requests waiting to be served (scheduled to p, yet have not been picked up), while the requests dropped off by p are not included.

In Definition 2, the service list L_p of PV p clearly points out which requests should be served and implies where to pick up or drop off corresponding passengers. Let $\mathcal{L}$ be a set of service lists of all PVs. L_p has two parts, the requests being served, and the requests waiting to be served by p. Once a new request is assigned to PV p, it will be put to the service list. Once a scheduled request is dropped off at its destination, it will be removed from the service list. As can be seen, the service list is determined by both the ride-sharing planning strategy and the motion states of PVs, and it dynamically changes over time.

Definition 3: The path of PV p, denoted by Q_p, is a sequence of points p has to serve, which includes the current position of this PV, the destinations of the requests being served by it and the origin-destination pairs of the requests waiting to be served by it.

As can be seen from Definition 3, the path of each PV dynamically changes over time, and is composed of the current position of this PV, the origin-destination pairs of requests waiting to be served and the destinations of requests being served. Let $\mathcal{Q}$ be a set of paths of all PVs. All the important variables and notations in this article are summarized in Table I, where PSA is the abbreviation of "potential search area" which will be discussed in the subsequent sections, and the unit of $D(i, j)$, $E(i, j)$, T_r, b, and B is km.

An example is shown to illustrate the changes of the schedule, path, and service list. Assume that the previous service list of PV p is $\{1, 2, 3\}$, and the previous path is $\{x \to o_1 \to o_2 \to o_3 \to d_2 \to d_1 \to d_3\}$ where x is the current position of p. The change of path has two cases. *First*, the service list does not change while the path changes. For example, after a certain time, the path becomes $\{x' \to o_2 \to o_3 \to d_2 \to d_1 \to d_3\}$ where x' is the new position of p. *Second*, both the service list and the new path change. For example, after a certain time, the cloud receives a new request r_4, and calculates the new ride-sharing path, and finally assigns p to serve r_4. The service list of p becomes $\{1, 2, 3, 4\}$, and the path becomes $\{x' \to o_1 \to o_2 \to o_3 \to o_4 \to d_2 \to d_1 \to d_3 \to d_4\}$.

B. Problem Formulation and NP-Completeness

To improve the profits, PVs try to serve more passengers with the shortest travel distance; however, this reduces the user satisfaction. To improve the service quality, passengers want to arrive their destinations as early as possible, i.e., to reduce the waiting time and travel time. To balance the utilities of

TABLE I
VARIABLES AND NOTATIONS

n	number of passengers of r.
t	earliest start time of r.
o	origin of r.
d	destination of r.
δ	detour ratio of request r.
Δ	maximum detour ratio.
b	buffer distance of request r.
B	buffer distance threshold.
w	waiting time of request r.
W	waiting time threshold.
C	capacity of PVs.
L_p	service list of PV p.
$D(i,j)$	shortest path distance from position i to j.
$E(i,j)$	Euclidean distance from position i to j.
p	a PV.
p_s	position of PV p at the schedule time of r.
$\mathcal{P}$	a set of all PVs in a city.
r	a request.
T_r	travel distance of request r.
$\mathcal{R}$	a set of all requests.
$\mathcal{R}_s$	a set of scheduled requests.
$\mathcal{R}_{s,1}$	a set of scheduled requests being served.
$\mathcal{R}_{s,2}$	a set of scheduled requests waiting to be served.
$\mathcal{R}_u$	a set of unscheduled requests.
L_p	service list of PV p.
$\mathcal{L}$	a set of service lists of all PVs.
Q_p	path of PV p.
$\mathcal{Q}$	a set of paths of all PVs.
η	ratio of PSA and optimal PSA (PSA$^{\text{opt}}$).
$\hat{A}_1/A_1$	PSA of p if the furthest request r has been picked up, and $A_1 = \lvert\hat{A}_1\rvert$.
$\hat{A}_2/A_2$	PSA of p if the furthest request r has not been picked up, and $A_2 = \lvert\hat{A}_2\rvert$.
$\hat{A}/A$	PSA of PV p, and $A = \lvert\hat{A}\rvert$.
$\hat{A}^{\text{opt}}/A^{\text{opt}}$	PSA$^{\text{opt}}$ of p, and $A^{\text{opt}} = \lvert\hat{A}^{\text{opt}}\rvert$.
$\hat{\beta}/\beta$	PSA determined by (o,d), and $\beta = \lvert\hat{\beta}\rvert$.
$\hat{\beta}^{\text{opt}}/\beta^{\text{opt}}$	PSA$^{\text{opt}}$ determined by (o,d), and $\beta^{\text{opt}} = \lvert\hat{\beta}^{\text{opt}}\rvert$.
$\hat{\alpha}/\alpha$	PSA determined by (p_s,o), and $\alpha = \lvert\hat{\alpha}\rvert$.
$\hat{\alpha}^{\text{opt}}/\alpha^{\text{opt}}$	PSA$^{\text{opt}}$ determined by (p_s,o), and $\alpha^{\text{opt}} = \lvert\hat{\alpha}^{\text{opt}}\rvert$.

PVs and passengers, the total utilities of both sides to achieve the best social welfare should be considered.

The detour ratio of a request is considered as the passenger QoS. Assume that p serves r. With respect to request r, the actual travel distance T_r is not shorter than $D(o,d)$, the shortest path distance from the origin o to the destination d. The detour ratio is defined as $\delta = (T_r - D(o,d))\ /\ D(o,d)$. The QoS constraint is then $\delta \leq \Delta$ where Δ is the maximum detour ratio set by the cloud, which aims at preserving the comfort [6] of passengers.

The latest arrival time is not considered here for several reasons. 1) It is hard to accurately predict the speed due to traffic congestions, emergencies, and accidents, although some researchers have proposed new solutions [24]. 2) It is hard to ensure that a passenger can arrive at his/her destination before the latest arrival time he/she set, even taking a car or taxi.

The PVP problem is formulated as follows. Given a set of PVs $\mathcal{P}$ on the road networks with the current service lists $\mathcal{L}$ and paths $\mathcal{Q}$, a set of scheduled requests $\mathcal{R}_{s,2}$ waiting to be served, and a set of unscheduled requests $\mathcal{R}_u$, the cloud determines the new service lists $\mathcal{L}'$ and new paths $\mathcal{Q}'$ of all PVs ensuring that all requests can be served from origins to destinations with QoS constraints (the detour ratio is no larger than the maximum value Δ), which aims to reduce the total travel distance of PVs and the waiting time of passengers with QoS guarantee.

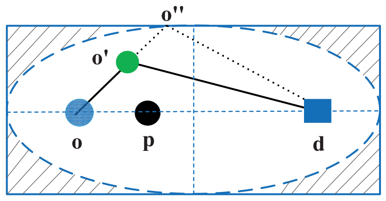


Fig. 2. PSA determined by the furthest request r which has been picked up: $\hat{A}_1 = \hat{\beta}$.

The NP-completeness of the PVP problem is discussed here. It is known to all that the dial-a-ride problem (DARP) is NP-complete [25], which is a special case of the PVP problem. Therefore, the PVP problem is also NP-complete.

Here, the differences between the two problems PVP and DARP are given as follows. In the DARP problem, all vehicles are based at a single depot, whereas, in the PVP problem, vehicles are distributed at different locations. The DARP problem is an offline/static ride-sharing process, i.e., all requests are known and all passengers stay at their origins waiting to be served, and it does not consider serving other unscheduled requests on the vehicle paths. On the other hand, the PVP problem focuses on online/dynamic ride-sharing so that more complex scenarios should be considered, e.g., the precedence constraints between current positions of PVs and the origin-destination pair of each request, and the predetermined match between each scheduled request and the corresponding PV.

IV. Proposed Solution

In this section, first some heuristics to restrict the potential search areas for PVs are introduced, and then a key routine of calculating the insertion cost of an origin-destination pair is described, and then an algorithm for online updating PV ride-sharing paths is proposed, and finally an example is shown to illustrate this algorithm.

A. Heuristics

The Potential Search Area (PSA), is an area that points out that the origins or/and destinations of requests in this area are possible to be served with ensured QoS (denoted by detour), while the other requests violating QoS constraints are not in this area and should be excluded.

r is named as the *furthest request* if its destination d is the last point on the path of the PV which serves r. In the following, PSA in two conditions are discussed, i.e., if the furthest request has been picked up or not. Then, two cases of PSA for a request are described, and then the PSA of a PV is derived. Here, the PSA of a request and the PSA of a PV are two distinct concepts, while the latter is derived from the former.

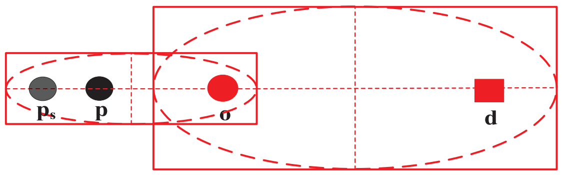


Fig. 3. PSA determined by the furthest request r which has not been picked up: $\hat{A}_2 = \hat{\alpha} \bigcup \hat{\beta}$.

1) The Furthest Request Has Been Picked Up: Assume that p serves r. As shown in Fig. 2, if new points (origins or destinations of other requests) between p and d are inserted, an area should be restricted such that δ (the detour ratio of r) does not exceed its maximum value Δ.

Lemma 1: Any point (the origin o' or destination d' of any request r') inserted between origin o and destination d of request r should fall in an ellipse determined by o and d, otherwise, the QoS will be violated.

Proof: In Fig. 2, a point, e.g., o' (origin of r') will be inserted between o and d. Let $D(i, \ldots, j)$ and $E(i, \ldots, j)$ denote the shortest path distance and the Euclidean distance from the first to the last location among positions $(i, i+1, \ldots, j)$, respectively. Obviously, $(1+\Delta)D(o,d) \geq D(o,o',o'',d) \geq E(o,o',o'',d)$, i.e., if an ellipse is drawn centering at o and d with the major axis $(1+\Delta)D(o,d)$, the insertion points should fall in this ellipse. Otherwise, δ (the detour ratio of r) will exceed the maximum value Δ, and the QoS of r will be violated. □

To simplify the computation, instead of an ellipse (denoted by the dashed blue line in Fig. 2), the corresponding rectangle (denoted by the solid blue rectangle in Fig. 2) is used to check if the points satisfy the QoS constraints. The enlarged area is limited, which is denoted by the shadowed areas in Fig. 2. Clearly, $D(o,d) \geq E(o,d)$. The PSA determined by (o,d) is denoted by $\hat{\beta}$. Let $\beta = |\hat{\beta}|$, which is shown by

$$\beta = (1+\Delta)\, D(o,d) \sqrt{(1+\Delta)^2 D^2(o,d) - E^2(o,d)}.$$

Finally, the PSA of p is got if the furthest request r has been picked up by $\hat{A}_1 = \hat{\beta}$. Let $A_1 = |\hat{A}_1|$, therefore, $A_1 = \beta$.

2) The Furthest Request Has Not Been Picked Up: Let the schedule time denote the time when schedule of r is confirmed. To limit the waiting time of r, the *buffer distance*, b (km) with a threshold B (km) is introduced, which denotes the travel distance of p from the schedule time to its pick-up time. Here, the position of PV p at the schedule time of r should be recorded, which is denoted by p_s just as shown in Fig. 3. Similarly, the PSA determined by (p_s, o) which is denoted by $\hat{\alpha}$ with $\alpha = |\hat{\alpha}|$, and the PSA determined by (o,d) which is denoted by $\hat{\beta}$ with $\beta = |\hat{\beta}|$ will be got. We obtain the PSA of p if the furthest request r has not been picked up denoted by $\hat{A}_2$ with $A_2 = |\hat{A}_2|$ through the union of $\hat{\alpha}$ and $\hat{\beta}$.

$$\alpha = W\sqrt{W^2 - E^2(p_s, o)},$$
$$\hat{A}_2 = \hat{\alpha} \bigcup \hat{\beta}.$$

Next, how to calculate the PSA of a PV is presented here. Assume that request r with the origin o and the destination d is the furthest request. The request r' with the destination d' is a request which is being served by p. We choose the PSA determined by the furthest request (i.e., r) not others (e.g., r') as the PSA of PV p. The reason is that, if we choose r' to calculate the PSA of p, we only consider the insertion positions between the current position of p and d', such that the insertion positions between d' and d are ignored.

Let $\hat{A}$ denote the PSA of PV p with $A = |\hat{A}|$, which has two cases and is determined by if the furthest request r has been picked up:

$$\hat{A} = \begin{cases} \hat{\beta}, & \text{if } r \text{ has been picked up,} \\ \hat{\alpha} \bigcup \hat{\beta}, & \text{otherwise,} \end{cases} \tag{1}$$

and

$$A = \begin{cases} \beta, & \text{if } r \text{ has been picked up,} \\ |\hat{\alpha} \bigcup \hat{\beta}|, & \text{otherwise.} \end{cases} \tag{2}$$

B. A Key Routine

Calculating the insertion cost of an origin-destination pair is a key routine in the proposed algorithm. For a new request $r \in R_u$, its origin o has to be visited before its destination d. Therefore, if p decides to take request r, both o and d need to be inserted into its current path with the precedence constraint being satisfied.

Definition 4: Assume that r is taken by p. The insertion cost $\phi_{r,p,i,j}$ at (i,j) is the additional travel distance of p if inserting an origin-destination pair (o,d) of r at the i^{th} and j^{th} positions, respectively on the path of p with o precedes d.

Let $Q_p = \{\theta_0, \theta_1, \ldots, \theta_K\}$ denote the current path of p, where θ_0 is the position of this PV. Let $D(\theta_i, \ldots, \theta_j) = D(\theta_i, \theta_{i+1}) + \ldots, +D(\theta_{j-1}, \theta_j)$ $(i < j)$ denote the sum of shortest path distance from θ_i to θ_{i+1}, ... and to θ_j. (3)~(6) describe four cases. The *first* case given by (3) means that d is not the last point of the path, and o immediately precedes d. The *second* case given by (4) means that d is the last point, and o immediately precedes d. The *third* case given by (5) means that d is the last point, and o does not immediately precede d. The *fourth* case given by (6) means that d is not the last point, and o does not immediately precede d. Let a function $Q_p = INSERT(r, p, i, j)$ returns Q_p (the path of p) by inserting the origin o and the destination d of r at the i^{th} and j^{th} positions, respectively, on the path of p.

$$\phi_{r,p,i,j} = D(\theta_i, o, d, \theta_{i+1}) - D(\theta_i, \theta_{i+1}), \quad \text{if } 0 \leq i \leq K-1, \quad j = i+1, \tag{3}$$
$$\phi_{r,p,i,j} = D(\theta_K, o, d), \quad \text{if } i = K, \quad j = i+1, \tag{4}$$
$$\phi_{r,p,i,j} = D(\theta_i, o, \theta_{i+1}) + D(\theta_K, d) - D(\theta_i, \theta_{i+1}), \quad \text{if } 0 \leq i \leq K-1, \quad j = K+1, \tag{5}$$
$$\phi_{r,p,i,j} = D(\theta_i, o, \theta_{i+1}) + D(\theta_j, d, \theta_{j+1}) - D(\theta_i, \theta_{i+1}) - D(\theta_j, \theta_{j+1}), \quad \text{if } 0 \leq i \leq K-1, \quad j \leq K, \quad j \neq i+1. \tag{6}$$

C. Algorithm

The basic idea of the proposed solution is that, a search area for each PV is drawn in three cases according to the insertion positions of origin-destination pairs of requests which

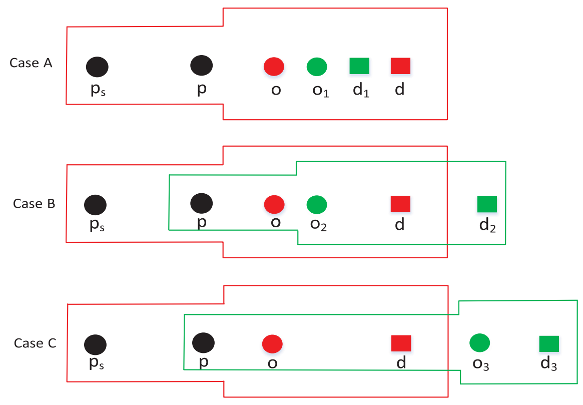


Fig. 4. Corresponding PSA in three insertion cases.

ensures that serving the requests out of this search area leads to the violation of QoS, so that only the requests in this limited search area are searched instead of the whole city.

Here, three insertion cases are discussed using PSA on the paths of PVs according to the insertion positions of requests. In order to insert the origin-destination pair of a new request r, select a position θ_i $(0 \leq i \leq K)$ of path of p to insert o after θ_i. Then from the positions after o select another position θ_j $(i+1 \leq j \leq K+1)$ to insert d. As shown in Fig. 4, there are three cases according to the final insertion positions of the origin and the destination:

Case $\mathbb{A}$: None of the origin (o_1) and the destination (d_1) becomes the last point of the new path.

Case $\mathbb{B}$: The destination (d_2) becomes the last point of the new path, while the origin (o_2) does not precede d_2 immediately.

Case $\mathbb{C}$: The destination (d_3) becomes the last point of the new path, and the origin (o_3) precedes the d_3 immediately.

In Fig. 4, the corresponding PSA in three insertion cases is depicted. d is the last point on the path before new requests are inserted. (o_i, d_i) is the origin-destination pair of request r_i $(i \in \{1, 2, 3\})$. The three requests r_1, r_2, and r_3 belong to cases $\mathbb{A}$, $\mathbb{B}$, and $\mathbb{C}$, respectively. The red lines denote the previous PSA of this PV, and the green lines in cases $\mathbb{B}$ and $\mathbb{C}$ denote the new PSA of this PV. Fig. 4 implies that the PSA in cases $\mathbb{B}$ and $\mathbb{C}$ has changed since the destination of the newly inserted request becomes the last point on the path. Finally, case $\mathbb{C}$ is discussed: all points of the previous path of p fall in the PSA $\hat{\alpha}$, which is determined by the current position of p and o since the PSA determined by o and d can be ignored.

If only restricting an area for each PV is considered, there may exist some "dead zones" such that some unscheduled requests do not satisfy QoS constraints of any PSA of PVs for a long time. To avoid this condition, a waiting time threshold W for each request is introduced. Let w denote the waiting time for request r. If w is larger than a threshold W, the insertion cost is directly calculated, ignoring the check about the buffer distance and PSA.

Algorithm 1 presents the proposed solution named as PSA-based Path planning algorithm (PSAP). Here, δ' and b' are the detour ratio and the buffer distance of request r', respectively. At some time t_0, the cloud selects the unscheduled requests $\mathcal{R}_u$, whose earliest start time is not later than t_0. Then the cloud sorts requests R_u in the descending order of their waiting time. In this algorithm, assume that the PSA of all PVs has been calculated using (1) and (2), which can be inferred by line 28.

The waiting time of requests is balanced in line 2 such that the waiting time of each request will not be too long. Lines 5∼6 check capacity constraints. Lines 9∼14 mean that, this algorithm first checks if the origin and/or destination of the request are/is in PSA, and then calculates the insertion cost in three cases ($\mathbb{A}$, $\mathbb{B}$, and $\mathbb{C}$). In lines 15 and 19, the QoS constraint is checked. Line 21 calculates the minimum insertion cost for PV p, and line 22 calculates the minimum insertion cost for all PVs. Lines 23∼26 imply that, if the cloud can find a suitable PV to serve the request, the path and the service list of this PV should be updated, and this request will be removed from the set of unscheduled requests.

Algorithm 1 PSAP

```
Input: $\mathcal{R}_u$, set of unscheduled requests;
       $\{Q_p\}_{p \in P}$, current paths of PVs;
       $\{L_p\}_{p \in P}$, current service lists of PVs;
Output: $\mathcal{R}'_u$, new set of unscheduled requests;
        $\{L'_p\}_{p \in P}$, new service lists of PVs;
        $\{Q'_p\}_{p \in P}$, new paths of PVs;
01: Initialization:$\{Q'_p\} \leftarrow \{Q_p\}$,$\{L'_p\} \leftarrow \{L_p\}$,$\mathcal{R}'_u \leftarrow \mathcal{R}_u$;
02: Sort the unscheduled requests in the descending
    order of their waiting time;
03: for $r \in \mathcal{R}_u$ do
04:   for $p \in \mathcal{P}$ do
05:     if $|L'_p| + n > C$ then
06:       $\phi_{r,p} \leftarrow \infty$;
07:     else
08:       if $w \leq W$
09:         if $o \in \hat{A}$ AND $d \in \hat{A}$
10:           Calculate $\{\phi_{r,p,i,j}\}_{i,j}$ of case $\mathbb{A}$;
11:         if $o \in \hat{A}$ AND $d \notin \hat{A}$
12:           Calculate $\{\phi_{r,p,i,j}\}_{i,j}$ of case $\mathbb{B}$;
13:         if All points on path of $p$ fall in $\hat{\alpha}$
14:           Calculate $\{\phi_{r,p,i,j}\}_{i,j}$ of case $\mathbb{C}$;
15:         if $\exists\ r' \in \{r \bigcup L_p\}$, $\delta' > \Delta$ OR $b' > B$
16:           $\phi_{r,p,i,j} \leftarrow \infty$;
17:       else
18:         Calculate $\{\phi_{r,p,i,j}\}_{i,j}$;
19:         if $\exists\ r' \in \{r \bigcup L_p\}$, $\delta' > \Delta$
20:           $\phi_{r,p,i,j} \leftarrow \infty$;
21:     $\phi_{r,p} \leftarrow \min\{\phi_{r,p,i',j'}\}_{i',j'}$;
22:   $\phi_r \leftarrow \phi_{r,p',i'',j''} = \min\{\phi_{r,p}\}_p$;
23:   if $\phi_r \neq \infty$ then
24:     $Q'_{p'} \leftarrow$ INSERT($r, p', i'', j''$); // Update path
25:     $L'_{p'} \leftarrow L'_{p'} \bigcup \{r\}$; // Update service list
26:     $\mathcal{R}'_u \leftarrow \{\mathcal{R}'_u \backslash r\}$; // Update the set of unsche-
        duled requests
27:     if $d$ becomes the last point on path of $p'$
28:       Calculate $A'$; // Update PSA of $p'$
```

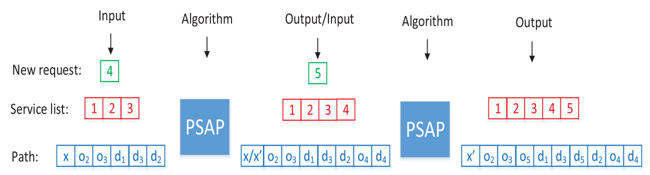


Fig. 5. An example of PSAP.

Lines 27∼28 means that, if destination of the newly inserted request becomes the last point on the path, the PSA of this PV should also be updated, since the PSA is determined by the furthest request.

D. Example

Fig. 5 shows an example to illustrate the PASP algorithm, which has two stages. In the *first* stage, a PV serves three requests 1, 2, and 3, i.e., the service list is $\{1, 2, 3\}$, and the current path is $\{x \to o_2 \to o_3 \to d_1 \to d_3 \to d_2\}$ where x is the current position of this PV, o_i is the pick-up point of request i and d_i is the drop-off point of request i with $i = \{1, 2, 3, 4, 5\}$. Request 1 has been picked up and others are still waiting to be served. Now, the cloud receives a new request 4, while only the case $\mathbb{C}$ satisfies QoS constraints, therefore, the new path becomes $\{x \to o_2 \to o_3 \to d_1 \to d_3 \to d_2 \to o_4 \to d_4\}$ and the new service list becomes $\{1, 2, 3, 4\}$.

In the *second* stage, the position of this PV becomes x', and request 1 still has not been dropped off, and requests 2, 3, and 4 have not been picked up, and now the cloud receives a new request 5. PSAP finds that only case $\mathbb{A}$ satisfies QoS constraints, and the path $\{x' \to o_2 \to o_3 \to o_5 \to d_1 \to d_3 \to d_5 \to d_2 \to o_4 \to d_4\}$ achieves the minimum insertion cost, which becomes the new path and $\{1, 2, 3, 4, 5\}$ becomes the new service list.

V. Performance Analysis

In this section, first the proposed solution PSAP is compared with an exhaustive search (ES) method, and then several properties about the gap between PSA and the optimal PSA (PSA$^{\text{opt}}$) on the performance is presented, and finally the reduction ratio of computational complexity is analyzed.

A. PSAP vs. ES

ES has to try each insertion position in the path of each PV, i.e., ignoring the process of checking requests which insertion case they belong to and directly calculating the insertion cost through (3)∼(6). The other process is the same as that of PSAP. That is, ES can be revised from PSAP: lines (9)∼(14) are replaced by "Calculate $\{\phi_{r,p,i,j}\}_{p,i,j}$ through (3)∼(6)" by trying each insertion position over all PVs, such that a large amount of computation is needed in ES. We see that PSAP and ES have the same service quality performance while different computational complexity. PSAP can greatly improve the computational efficiency by restricting the search areas compared with ES since PSAP only tries a part of requests and others are excluded.

Here, more details in PSAP are presented to show its advantages. As shown in Fig. 2, some requests which violate QoS are excluded, since the detour of at least a request is larger than Δ if they are not inserted in PSA. The scenario in Fig. 3 is similar. The travel distance of the PV from schedule time to pick-up time is limited, and the detour is limited. Therefore, the QoS of requests is preserved. However, all the above scenarios which violating QoS will be calculated in ES, leading to a large amount of computation.

B. Gap Between PSA and PSA^{opt}

Definition 5: If the furthest request r has been picked up, the optimal PSA (PSA^{opt}) of PV p is the ellipse centering at (o, d), otherwise, is the union of two ellipses centering at (p_s, o) and (o, d) respectively, just as discussed in Section IV-A.

In practice, PSA instead of PSA$^{\text{opt}}$ is used since the enlarged area is limited. There are two examples to discuss PSA$^{\text{opt}}$. In Fig. 2, PSA$^{\text{opt}}$ is the ellipse and the PSA is the corresponding rectangle. In Fig. 3, PSA$^{\text{opt}}$ is the union of two ellipses and the PSA is the union of two corresponding rectangles. The PSA is always a litter larger than PSA$^{\text{opt}}$.

Here, the gap between PSA and PSA$^{\text{opt}}$ is discussed to show how much computational complexity is enlarged using PSA than the latter one. Let $\hat{A}$ denotes PSA of p with $A = |\hat{A}|$, and $\hat{A}^{\text{opt}}$ denotes PSA$^{\text{opt}}$ of p with $A^{\text{opt}} = |\hat{A}^{\text{opt}}|$, and let η denote the ratio of PSA and PSA$^{\text{opt}}$, which is formulated as (7). Obviously, $\eta > 1$. In practice, the smaller the gap between PSA and PSA$^{\text{opt}}$ is, the better, since this means more computational complexity is saved. That is, the smaller η is, the more efficient the PSAP will be.

$$\eta = \frac{A}{A^{\text{opt}}}. \tag{7}$$

Theorem 1: With respect to any PV p, if the furthest request r has been picked up, the ratio of PSA and PSA^{opt} $\eta = \frac{A}{A^{opt}} = \frac{4}{\pi}$.

Theorem 1 can be obtained according to the area of the ellipse and corresponding rectangle. Next, let's discuss the gap between PSA and PSA$^{\text{opt}}$ of any PV p. Let r denote the furthest request on path of p.

From Fig. 3, if the furthest request has not been picked up, the PSA (or PSA$^{\text{opt}}$) of the PV is obtained by union of two PSA (or PSA$^{\text{opt}}$) determined by the request. Now let $\hat{\alpha}^{\text{opt}}$ denote PSA$^{\text{opt}}$ determined by (p_s, o) with $\alpha^{\text{opt}} = |\hat{\alpha}^{\text{opt}}|$, and $\hat{\beta}^{\text{opt}}$ denote PSA$^{\text{opt}}$ determined by (o, d) with $\beta^{\text{opt}} = |\hat{\beta}^{\text{opt}}|$, and $\hat{A}^{\text{opt}}$ denote PSA$^{\text{opt}}$ of PV p with $A^{\text{opt}} = |\hat{A}^{\text{opt}}|$. Let $\hat{A}^{\text{opt}} = \hat{\alpha}^{\text{opt}} \bigcup \hat{\beta}^{\text{opt}}$. Let $\hat{\mu} = \hat{\alpha} \bigcap \hat{\beta}$ with $\mu = |\hat{\mu}|$ and $\hat{\nu} = \hat{\alpha}^{\text{opt}} \bigcap \hat{\beta}^{\text{opt}}$ with $\nu = |\hat{\nu}|$. Clearly, $\hat{\mu} \supseteq \hat{\nu}$, and $\mu \geq \nu$.

Theorem 2: With respect to any PV p, no matter if the furthest request r has been picked up or not, $\eta = \frac{A}{A^{opt}} \in \left[\max\left(1, \frac{4}{\pi} + \frac{4\nu - \pi\mu}{\pi(\alpha^{opt}+\beta^{opt})}\right), \frac{4}{\pi} + \frac{(4-\pi)\mu}{\pi(\alpha^{opt}+\beta^{opt}-\nu)}\right]$.

Proof: From (1), (2), and Theorem 1, we know that η is involved by the state of the furthest request. Now, we discuss the *first* case, i.e., the furthest request r has not been picked up. From Theorem 1, we know that,

$$\frac{\alpha}{\alpha^{\text{opt}}} = \frac{\beta}{\beta^{\text{opt}}} = \frac{4}{\pi}.$$

Considering of the definition of μ and ν, we get the following:

$$\begin{aligned} A &= \alpha + \beta - \mu, \\ A^{\text{opt}} &= \alpha^{\text{opt}} + \beta^{\text{opt}} - \nu, \\ \eta &= \frac{\alpha + \beta - \mu}{\alpha^{\text{opt}} + \beta^{\text{opt}} - \nu}. \end{aligned}$$

And then η is reformulated as

$$\eta = \frac{4}{\pi} + \frac{4\nu - \pi\mu}{\pi(\alpha^{\text{opt}} + \beta^{\text{opt}} - \nu)}.$$

Considering that $\eta > 1$, we get

$$\max\left(1, \frac{4}{\pi} + \frac{4\nu - \pi\mu}{\pi(\alpha^{\text{opt}} + \beta^{\text{opt}})}\right) \leq \eta \leq \frac{4}{\pi} + \frac{(4-\pi)\mu}{\pi(\alpha^{\text{opt}} + \beta^{\text{opt}} - \nu)}.$$

Then, the *second* case is discussed, i.e., the furthest request r has been picked up. Obviously, $\mu = \nu = 0$, and $\eta = \frac{4}{\pi}$. So the above inequality about the scope of η is still workable.

Finally, Theorem 2 is proved. □

C. Reduction Ratio of Computational Complexity

The Reduction Ratio of Computational Complexity (RRCC) includes three cases: $\mathbb{A}$, $\mathbb{B}$, and $\mathbb{C}$. Let $M_{\mathbb{A}}$, $M_{\mathbb{B}}$, and $M_{\mathbb{C}}$ denote the number of insertion times of origin-destination pairs in PSAP in cases $\mathbb{A}$, $\mathbb{B}$, and $\mathbb{C}$. Let $N_{\mathbb{A}}$, $N_{\mathbb{B}}$, and $N_{\mathbb{C}}$ denote the number of insertion times of origin-destination pairs in ES in three cases. Let $\psi_{\mathbb{A}}$, $\psi_{\mathbb{B}}$, and $\psi_{\mathbb{C}}$ denote RRCC in three cases. For example, in case $\mathbb{A}$, $M_{\mathbb{A}}$ and $N_{\mathbb{A}}$ are recorded, and then $\psi_{\mathbb{A}}$ is obtained by

$$\psi_{\mathbb{A}} = (N_{\mathbb{A}} - M_{\mathbb{A}}) \,/\, N_{\mathbb{A}}.$$

$\psi_{\mathbb{B}}$ and $\psi_{\mathbb{C}}$ can be obtained in a similar way. In this subsection, the estimation for RRCC is mainly discussed since it directly affects the computational complexity of PSAP: the larger RRCC is, the more efficient PSAP will be.

Computing PSA involves several square operations, and computing if the origin or destination is in PSA only involves linear operations. Both of them have less computational complexity than the shortest path algorithm such as Dijkstra algorithm and calculating the minimum insertion cost [9]. Therefore, the cost of computing PSA and computing if the origin or destination is in PSA can be ignored when analyzing RRCC.

Let S denote the area of the city. If the PSA is smaller, more requests will be excluded, and the PSAP solution will be more efficient. For example, now insert an origin-destination pair (o', d') of request r' based on PSA $\hat{A}$. In case $\mathbb{A}$, both o' and d' are in the PSA $\hat{A}$.

Assume that the event $o' \in \hat{A}$ is independent with another event $d' \in \hat{A}$. In case $\mathbb{A}$, the probability of $o', d' \in \hat{A}$ is $\frac{A^2}{S^2}$. In case $\mathbb{B}$, the probability of $o' \in \hat{A}$ is $\frac{A}{S}$. In case $\mathbb{C}$, make sure that all the points on the path should fall in the PSA determined by the current position of p and o.

Generally, the execution time of case $\mathbb{A}$ is more than case $\mathbb{B}$, and the execution time of case $\mathbb{C}$ is the minimum among three cases, which can be inferred from Section IV-C. The execution time of case $\mathbb{C}$ is not meaningful since it needs at most one calculation for one request. The mathematical expectation of RRCC in cases $\mathbb{A}$ and $\mathbb{B}$ becomes a focus, and is obtained by

$$\mathbf{E}(\psi_{\mathbb{A}}) = 1 - A^2/S^2, \tag{8}$$

$$\mathbf{E}(\psi_{\mathbb{B}}) = 1 - A/S, \tag{9}$$

where $\mathbf{E}$ means the mathematical expectation.

On the current path of p $Q_p = \{\theta_0, \theta_1, \ldots, \theta_K\}$, insert o and d. Let i be the insertion position of o, and we get the following:

$$N_{\mathbb{A}} = \sum_{i=1}^{K-1}(K - i) = K(K-1)/2, \tag{10}$$

$$N_{\mathbb{B}} = K - 1, \tag{11}$$

$$N_{\mathbb{C}} = 1. \tag{12}$$

The total reduced number of insertion times of origin-destination pairs using PSAP compared with ES is denoted by I, and its mathematical expectation is obtained by

$$\begin{aligned} \mathbf{E}(I) &= N_{\mathbb{A}}\,\mathbf{E}(\psi_{\mathbb{A}}) + N_{\mathbb{B}}\,\mathbf{E}(\psi_{\mathbb{B}}) + N_{\mathbb{C}}\,\mathbf{E}(\psi_{\mathbb{C}}) \\ &\approx N_{\mathbb{A}}\,\mathbf{E}(\psi_{\mathbb{A}}) + N_{\mathbb{B}}\,\mathbf{E}(\psi_{\mathbb{B}}) \\ &= K(K-1)(1-A^2/S^2)/2 + (K-1)(1-A/S), \end{aligned} \tag{13}$$

where case $\mathbb{C}$ can be ignored since $N_{\mathbb{C}}$ is much smaller than $N_{\mathbb{A}}$ and $N_{\mathbb{B}}$.

Obviously, $\frac{A^2}{S^2} \leq \frac{A}{S} \leq 1$ and we get $\mathbf{E}(\psi_{\mathbb{A}}) \geq \mathbf{E}(\psi_{\mathbb{B}})$. From (10)∼(12), we know $N_{\mathbb{A}} \geq N_{\mathbb{B}} \geq N_{\mathbb{C}}$. From (13), we get that, cases $\mathbb{A}$ and $\mathbb{B}$ almost determine RRCC since generally $N_{\mathbb{A}}\,\mathbf{E}(\psi_{\mathbb{A}}) \geq N_{\mathbb{B}}\,\mathbf{E}(\psi_{\mathbb{B}}) \geq N_{\mathbb{C}}\,\mathbf{E}(\psi_{\mathbb{C}})$.

As can be seen from (8), (9), and (13), if the area of the city S is larger, RRCC will be larger, and the proposed solution will have higher computational efficiency.

VI. Performance Evaluation

In this section, first the simulation settings are discussed, and then the results to evaluate the performance of the proposed solution are presented.

A. Simulation Settings

Simulation settings include: PV setting, privately owned electric vehicle (POEV) setting, road map data, taxi data, and parameter setting.

1) PV Setting: Electric vehicles, Tesla Model-S [26] are used to study the transportation patterns of PVs, although they are not self-driving vehicles now. The number of seats of each PV is 5. Assuming that all PVs travel along the shortest path between any two positions (origins or destinations of requests) with the identical speed 30 km/h. The initial positions of PVs are randomly distributed in the road network which follows a uniform distribution. PVs always travel to serve passengers as long as new requests are assigned, and PVs stay at their positions if no requests are assigned to them.

2) POEV Setting: The Tesla Model-S [26] are also used to explore the traffic characteristics of POEVs. The settings of POEVs are the same as PVs except that POEVs only provide the origin-to-destination ride service for their owners along the shortest path instead of the ride-sharing path. Clearly, the number of POEVs is the half the number of requests if each POEV is used twice a day, which is much larger than that of PVs.

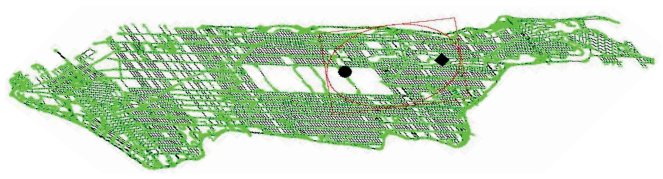


Fig. 6. Manhattan in New York City.

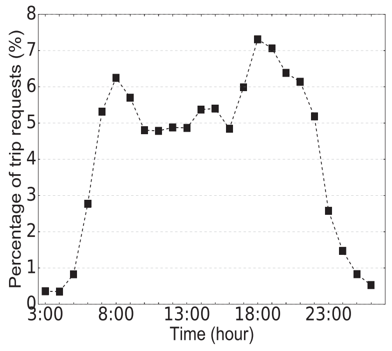


Fig. 7. Distribution of requests in one day.

3) Road Map Data: PSAP and ES are performed in the Manhattan road network with 60 km^2 in New York City, which is depicted by Fig. 6 where black lines denote road segments, and green points denote nodes. The road map is extracted through the openstreetmap [27] with six types of ways: primary, secondary, tertiary, motorway, motorway_link, and residential. Finally, 3,900 ways and 29,792 nodes are filtered.

The black circle and black square in Fig. 6 are the origin and the destination of a request, respectively. Assume that this request is the furthest and is just picked up by a PV at the origin, and the corresponding PSA and PSA$^{\text{opt}}$ are denoted by the red rectangle and the red ellipse, respectively. We can see that, in case $\mathbb{A}$, the search area denoted by the PSA is much smaller than the whole road network of the city, which implies that the search space is largely reduced and the computational efficiency is improved.

4) Taxi Data: The simulations use the taxi data set (yellow records) of the New York City on March 1, 2016 [28]. Each record contains several useful fields for this study, including pick-up time, drop-off time, trip distance, latitudes/longitudes of origins, latitudes/longitudes of destinations, fares, tax, tip, and total payment. Then the trip records with origin-destination pairs falling in Mahattan are selected.

Let's discuss three traffic characteristics in the Mahattan taxi data set. The *first* one is the distribution of these trip requests (only in Manhattan) in each hour on March 1 (Tuesday), 2016, which is shown by Fig. 7. The busiest time occurs at 8:00 and 18:00, respectively in morning and evening rush hours. The *second* one is the PDF of trip fare. As can be seen from Fig. 8, most of the trip fare is between 5∼15 US dollars. The *third* one is the PDF of trip distance. As shown in Fig. 9, most of the trip distance is less than 6 km.

Assume that the pick-up time of each request is equal to the earliest start time. All PVs travel as the ride-sharing path

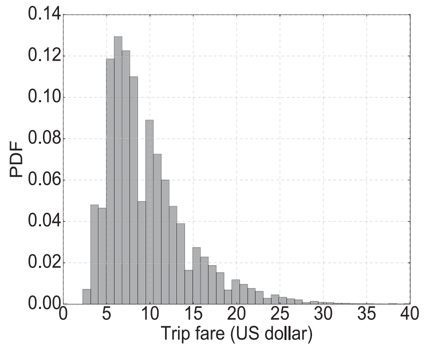


Fig. 8. PDF of trip fare.

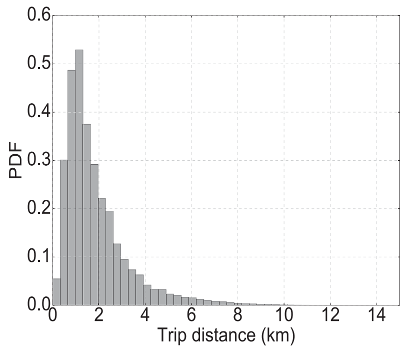


Fig. 9. PDF of trip distance.

computed by the cloud, and stop at the destinations of requests for the next schedule task. To make the performance more stable, the requests with Euclidean distance no less than 3 km are selected, and then the total number of trip requests of the day in Mahattan is 75,014, and the number is reduced to 3,112 in the period 11:00∼11:59.

5) Parameter Setting: Passengers prefer low detour ratio, which reflects the QoS of passengers and will be widely accepted if the value is not larger than the maximum value $\Delta = 0.2$. The waiting time threshold W (minute) is 4. If W is too large, some requests which fall in dead zones may have to wait for long time, however, if it is too small, the effect of PSAP is not clear since it directly calculate the insertion cost ignoring the check about PSA. The buffer distance threshold B (km) is set to 6. If B is set too large, the PSA of PVs may be too large just as the whole city and PSAP may spend more time on computing the paths of PVs, while if it is too small, most requests will fall in the limited search area and they will be assigned only after the waiting time is larger than the threshold W.

We implement PSAP and ES through a computer with an Intel Core-i7 (3.4 GHz and 32 GB of RAM) using C++ under Windows OS. In the 24-hour simulations, the time begins at 3:00 for two reasons. 1) This time is one of the most important shift handover time in taxi companies in many cities such as Shenzhen [29]. 2) The number of trip requests of this time is almost the minimum of one day in multiple cities such as Shanghai [9], Tokyo [30], and New York City [28].

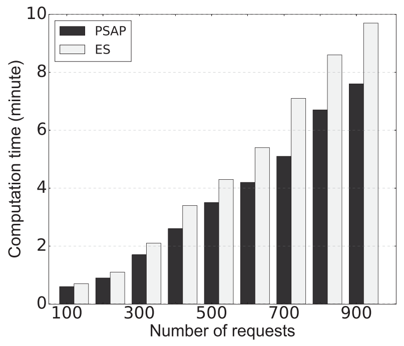


Fig. 10. Computation time comparison.

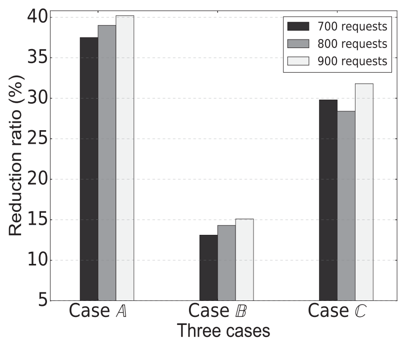


Fig. 11. RRCC in three cases.

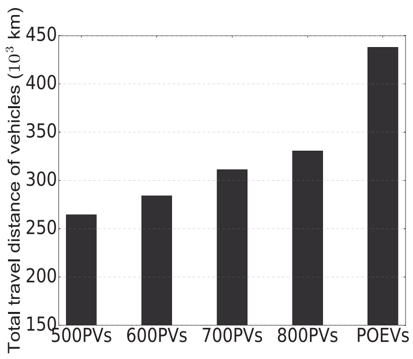


Fig. 12. Total travel distance.

B. Results

Here, two metrics are introduced: sharing rate and saved travel distance. 1) The sharing rate should imply the traffic sharing scenario in PV systems, and is denoted by the following

$$\text{sharing rate} = \frac{\sharp \text{ of requests being served}}{\sharp \text{ of moving vehicles}}. \qquad (14)$$

We see that the sharing rate of POEVs is always one which does not change since there is no sharing among different requests. 2) The saved travel distance is denoted by the sum of the shortest path distance of each origin-destination pair of requests minus the total travel distance of PVs. The saved travel distance implies the amount of saved energy of PV systems.

PSAP and ES have the same service quality and traffic performance. In the following, the computational complexity of PSAP and ES are shown, and then the traffic performance of PVs using PSAP and POEVs are presented to explore their transportation patterns.

The simulation results are presented in six items: computation time comparison, RRCC, total travel distance, utilization rate, sharing rate, and saved travel distance.

The *first* metric is the computation time, which is shown by Fig. 10 where the number of requests varies from 100 to 900 during 11:00∼11:59 and the number of PVs is 70. We get that, when the number of requests increases, the computing time of PSAP and ES both increases, however, the computing time using PSAP is reduced by 22% compared with ES. ES tries all the possible insertion positions while PSAP only tries a part of them since the requests violating QoS constraints are excluded. As can be seen from (8) and (9), if the urban area increases, more computation will be reduced using PSAP, i.e., PSAP will have better performance in traffic big data scenarios.

The *second* metric is RRCC, which is displayed by Fig. 11 where the number of PVs is 70, and the number of requests is set to 700, 800, and 900, respectively during 11:00∼11:59. The RRCC in case $\mathbb{A}$ is much higher than that in case $\mathbb{B}$, and the RRCC in case $\mathbb{C}$ is between the other two cases. The results can be inferred from (8) and (9). For example, when the number of requests is 900, the RRCC in three cases $\mathbb{A}$, $\mathbb{B}$ and $\mathbb{C}$ is 40.2%, 15.1%, and 31.8%, respectively. The result can be explained as follows. In case $\mathbb{A}$, both the origin and destination of a new request should be in PSA of the PV, while in case $\mathbb{B}$, only the origin should be in PSA. Therefore, the number of requests which satisfy case $\mathbb{A}$ is much smaller than that of case $\mathbb{B}$. In case $\mathbb{C}$, all the points of the current path of one PV should fall in PSA, which is related to the number of requests which are being served and waiting to be served, therefore, it is hard to predict the result in case $\mathbb{C}$ under such particular constraints.

Next, 500∼800 PVs are put to the system to serve 75,014 requests of the day. The average waiting-travel time of requests is (18.7, 14.1), (3.2, 13.7), (2.5, 13.1) and (2.1, 12.7) minutes using 500, 600, 700, and 800 PVs, respectively. This means that if more PVs are put to the system, the waiting and travel time will be less, since the service quality level will improve with less passengers sharing one vehicle.

The *third* metric is the total travel distance. As can be seen from Fig. 12, when more than 500 PVs are put to serve passengers, the total travel distance of PVs increases slowly since passengers share common paths during their trips. The travel distance of PVs is reduced by 24% ∼39% compared with that by POEVs when the number of PVs varies from 500 to 800.

The *fourth* metric is the utilization rate. As shown in Fig. 13, when 500 PVs are put to the system, during 8:00∼24:00, almost all PVs are busy on serving passengers since the number of PVs is too small to serve such a large number of requests. While if more PVs, e.g., 800, are used, the utilization rate of PVs almost increases to one during peak time such as 8:00, yet decreases obviously during non-peak time such as 12:00.

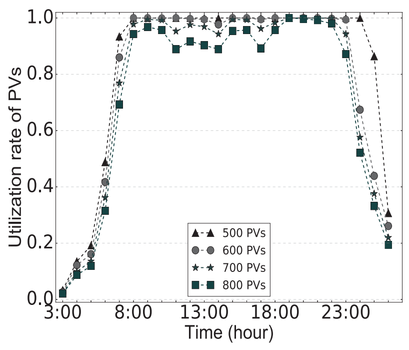


Fig. 13. Utilization rate.

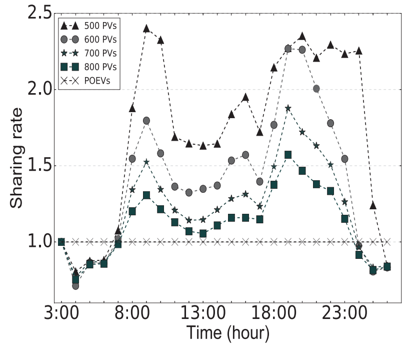


Fig. 14. Sharing rate.

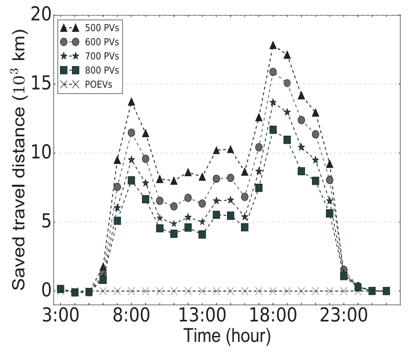


Fig. 15. Saved travel distance.

The *fifth* metric is the sharing rate, which is shown in Fig. 14. The less PVs are used, the larger the sharing rate will be. At non-peak time such as 4:00, the sharing rate is less than one, since very few passengers share PVs, however, PVs have to travel some distance to pick up them. However, at peak time such as 19:00, the sharing rate improves obviously since more passengers share PVs.

The *sixth* metric is the saved travel distance, which is is depicted by Fig. 15. At non-peak time, the saved travel distance is very small, even is negative at 4:00, since very few passengers share PVs at this time. However, at peak time, the saved travel distance increases largely especially at 8:00 and 18:00 since the more passengers join ride-sharing, the more the saved travel distance will be.

VII. Discussion

First, the energy efficiency of PV systems is discussed. Assume that the travel distance of PVs and POEVs are positively correlated with the energy consumption. From Fig. 12, we can infer that the average energy cost by PVs will be dropped by 24%∼39% compared with POEVs. In PV systems, although passengers may sacrifice some comfort during their trips, the discomfort emerged from ride-sharing is limited since PASP tries to balance the waiting time of passengers and restrict the detour of each passenger to a tolerated level. Therefore, the QoS of passengers is guaranteed, and the low-cost ride-sharing will attract more passengers to choose PV systems.

Second, the particular features of PV systems in peak and non-peak time are discussed. According to Fig. 12 and Fig. 15 which show the total travel distance and the saved travel distance, respectively, we can infer that, if too many PVs join the system, the profits will be reduced for the sake of parking fees; however, too few PVs can save more travel distance with the cost of degrading the service quality level of passengers. At traffic peak time, the saved travel distance is much more than that in non-peak time, since more passengers share PVs. Therefore, the number of PVs put to the system is an important factor which balances the PV system profits and passenger QoS. In summary, PV systems are more practical and profitable in traffic peak time of crowded urban areas; however, in traffic non-peak time or suburban areas with few passengers, the performance of PV systems is as good as that in the former scenario.

VIII. Conclusion

To deal with the online/dynamic ride-sharing path planning problem for PV systems, this article proposes a solution based on a limited potential search area for each vehicle to filter out the requests that violate passenger QoS constraints such as detour, therefore, the global search is reduced to a local search and the computational complexity is reduced. It also considers the comfort of passengers (e.g., waiting time and detour) and the total travel distance of PVs. Therefore, passengers can enjoy their peer-to-peer ride-sharing services with sacrificing a little ride comfort. Moreover, the proposed solution can be easily extended to the future globally optimal algorithm (if it will exist) to speed the computation time where all the scheduling can be changed only if the passenger has not been picked up. This article also analyzes the reduction ratio of computational complexity using the proposed solution. The simulations based on Manhattan taxi data sets evaluate the computational efficiency of the proposed solution.

The future research work will focus on the ride-sharing path planning methods with more preferences, e.g., the maximum number of shared persons, the pick-up/drop-off point selection, and the point of interest selection. In addition, the future research work should also consider the ride-sharing path planning solutions aiming at solving the common last mile problem from homes or work places to subway stations or bus stops.

REFERENCES

[1] T. Litman, "Transportation and public health," *Annu. Rev. Public Health*, vol. 34, pp. 217–233, Jan. 2013.

[2] A. E. Pisarski, "The transportation challenge—Moving the U.S. economy," Cambridge Systematics, Inc., Boston Logistics Group, Inc., Nat. Chamber Found., Washington, DC, USA, Tech. Rep., 2008.

[3] (2013). *Commuting in America 2013: The National Report on Commuting Patterns and Trends*. [Online]. Available: http://www.prtctransit.org/docs/Commuting_in_America_2013.pdf

[4] M. Zhu, J. Hu, L. Kong, R. Shen, W. Shu, and M.-Y. Wu, "An algorithm of lane change using two-lane nasch model in traffic networks," in *Proc. IEEE Int. Conf. Connected Vehicles Expo (ICCVE)*, Dec. 2013, pp. 241–246.

[5] P. Hystad, P. A. Demers, K. C. Johnson, R. M. Carpiano, and M. Brauer, "Long-term residential exposure to air pollution and lung cancer risk," *LWW Epidemiol.*, vol. 24, no. 5, pp. 762–772, 2013.

[6] M. Zhu, R. Shen, W. Shu, and M.-Y. Wu, "Traffic efficiency improvement and passengers comfort in ridesharing systems in VANETs," in *Proc. IEEE Int. Conf. Connected Vehicles Expo (ICCVE)*, Oct. 2015, pp. 116–121.

[7] (2018). *Beijing Air Pollution: Real-Time Air Quality Index (AQI)*. [Online]. Available: http://aqicn.org/city/beijing/

[8] J. Hamari, M. Sjöklint, and A. Ukkonen, "The sharing economy: Why people participate in collaborative consumption," *J. Assoc. Inf. Sci. Technol.*, vol. 67, no. 9, pp. 2047–2059, 2016.

[9] M. Zhu *et al.*, "Public vehicles for future urban transportation," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 12, pp. 3344–3353, Dec. 2016.

[10] M. Zhu, X.-Y. Liu, and X. Wang, "Joint transportation and charging scheduling in public vehicle systems—A game theoretic approach," *IEEE Trans. Intell. Transp. Syst.*, to be published.

[11] M. Zhu, X.-Y. Liu, M. Qiu, R. Shen, W. Shu, and M.-Y. Wu, "Traffic big data based path planning strategy in public vehicle systems," in *Proc. IEEE/ACM Int. Symp. Quality Service (IWQoS)*, Jun. 2016, pp. 1–2.

[12] M. Zhu, L. Kong, X.-Y. Liu, R. Shen, W. Shu, and M.-Y. Wu, "A public vehicle system with multiple origin-destination pairs on traffic networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–6.

[13] M. Zhu, X.-Y. Liu, L. Kong, R. Shen, W. Shu, and M.-Y. Wu, "The charging-scheduling problem for electric vehicle networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2014, pp. 3178–3183.

[14] (2018). *Homepage of Uber*. [Online]. Available: https://www.uber.com/

[15] (2018). *Homepage of DiDi*. [Online]. Available: http://www.xiaojukeji.com

[16] M. Zhu, X.-Y. Liu, M. Qiu, R. Shen, W. Shu, and M.-Y. Wu, "Transfer problem in a cloud-based public vehicle system with sustainable discomfort," *Mobile Netw. Appl.*, vol. 21, no. 5, pp. 890–900, 2016.

[17] S. Shang, L. Chen, Z. Wei, C. S. Jensen, J.-R. Wen, and P. Kalnis, "Collective travel planning in spatial networks," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 5, pp. 1132–1146, May 2016.

[18] J. Jung, R. Jayakrishnan, and J. Y. Park, "Dynamic shared-taxi dispatch algorithm with hybrid-simulated annealing," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 31, no. 4, pp. 275–291, 2016.

[19] S. Ma, Y. Zheng, and O. Wolfson, "Real-time city-scale taxi ridesharing," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 7, pp. 1782–1795, Jul. 2015.

[20] R. Massobrio, G. Fagúndez, and S. Nesmachnow, "Multiobjective evolutionary algorithms for the taxi sharing problem," *Int. J. Metaheuristics*, vol. 5, no. 1, pp. 67–90, 2016.

[21] J. Naoum-Sawaya *et al.*, "Stochastic optimization approach for the car placement problem in ridesharing systems," *Transp. Res. B, Methodol.*, vol. 80, pp. 173–184, Oct. 2015.

[22] P. Goel, L. Kulik, and K. Ramamohanarao, "Optimal pick up point selection for effective ride sharing," *IEEE Trans. Big Data*, vol. 3, no. 2, pp. 154–168, Jun. 2017.

[23] M. Ota, H. Vo, C. Silva, and J. Freire, "STaRS: Simulating taxi ride sharing at scale," *IEEE Trans. Big Data*, vol. 3, no. 3, pp. 349–361, Sep. 2017.

[24] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," *Transp. Res. C, Emerg. Technol.*, vol. 79, pp. 1–17, Jun. 2017.

[25] M. A. Masmoudi, K. Braekers, M. Masmoudi, and A. Dammak, "A hybrid genetic algorithm for the heterogeneous Dial-A-Ride problem," *Comput. Oper. Res.*, vol. 81, pp. 1–13, May 2017.

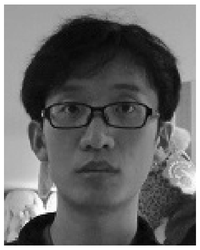
[26] (2018). *Homepage of Tesla Model-S*. [Online]. Available: http://my.teslamotors.com/models/design

[27] (2018). *Homepage of Openstreetmap*. [Online]. Available: http://www.openstreetmap.org/

[28] (2018). *New York City Taxi and Limousine Commission (TLC) Trip Data*. [Online]. Available: http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

[29] Z. Tian *et al.*, "Real-time charging station recommendation system for electric-vehicle taxis," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 11, pp. 3098–3109, Nov. 2016.

[30] B. Atasoy, T. Ikeda, X. Song, and M. E. Ben-Akiva, "The concept and impact analysis of a flexible mobility on demand system," *Transp. Res. C, Emerg. Technol.*, vol. 56, pp. 373–392, Jul. 2015.



Ming Zhu received the Ph.D. degree in computer science and engineering from Shanghai Jiao Tong University, Shanghai, China. He is currently a Post-Doctoral Researcher and an Assistant Researcher with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. A part of this work is finished in Shanghai Jiao Tong University.

His research interests are in the area of big data, artificial intelligence, Internet of Things, and wireless communications.

Xiao-Yang Liu received the B.Eng. degree in computer science from Huazhong University of Science and Technology, China, in 2010. He is currently pursuing the joint Ph.D. degree with the Department of Electrical Engineering, Columbia University, and with the Department of Computer Science and Engineering, Shanghai Jiao Tong University.

His research interests include tensor theory, deep learning, nonconvex optimization, big data analysis and homomorphic encryption, cyber-security, and wireless communication.

Xiaodong Wang (S'98–M'98–SM'04–F'08) received the Ph.D. degree in electrical engineering from Princeton University. He is currently a Professor of electrical engineering with Columbia University, New York NY, USA. He has authored the book *Wireless Communication Systems: Advanced Techniques for Signal Reception* (Prentice Hall, 2003). His research interests fall in the general areas of computing, signal processing, and communications. He has authored extensively in these areas. His current research interests include wireless communications, statistical signal processing, and genomic signal processing. He received the 1999 NSF CAREER Award, the 2001 IEEE Communications Society and Information Theory Society Joint Paper Award, and the 2011 IEEE Communication Society Award for Outstanding Paper on New Communication Topics. He has served as an Associate Editor for IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON SIGNAL PROCESSING, and IEEE TRANSACTIONS ON INFORMATION THEORY. He is an ISI Highly Cited Author.