# A Public Vehicle System with Multiple Origin-Destination Pairs on Traffic Networks

Ming Zhu*, Linghe Kong†*, Xiao-Yang Liu*, Ruimin Shen*, Wei Shu‡*, Min-You Wu*
*Shanghai Jiao Tong University, China
†McGill University, Canada.
‡University of New Mexico, USA
*{zhumingpassional, linghe.kong, yanglet, rmshen, mwu}@sjtu.edu.cn, ‡shu@ece.unm.edu

***Abstract*—Substantial technology advances have been made in areas of autonomous and connected vehicles, which open a wide landscape for future transportation systems. We propose a new type of transportation system, Public Vehicle (PV) system, to provide effective, comfortable, and convenient service. The PV system is to improve the efficiency of current transportation systems, *e.g.*, taxi system. Meanwhile, the design of such a system targets on significant reduction in energy consumption, traffic congestion, and provides solutions with affordable cost. The key issue of implementing an effective PV system is to design efficient scheduling algorithms. We formulate it as the PV Path (PVP) problem, and prove it is NP-Complete. Then we introduce a real time approach, which is based on solutions of the Traveling Salesman Problem (TSP) and it can serve people efficiently with lower costs. Our results show that to achieve the same performance (*e.g.*, the total time: waiting and travel time), the number of vehicles can be reduced by 47%-69%, compared with taxis. The number of vehicles on roads is reduced, thus traffic congestion is relieved.**

## I. Introduction

Public transportation systems such as buses or subways provide a low-cost travel option with a fixed geographic route and a fixed schedule for passengers. However, passengers may need to transfer by walking with inconvenience. In modern cities, cars and taxis serve majority of transportation needs for personal travel and daily commuting. Private cars or taxis provide convenient services at high cost. Impacts of non-shared rides on social cost, such as energy consumption, pollution, and traffic congestions, are paid attention to recent years. The amount of consumed fuels, *e.g.*, gasoline, diesel, and biofuels increases with population growth. How to use less number of vehicles and consume less amount of energy to meet the demands of transportation efficiently is a hot topic to be addressed.

In order to overcome the inefficiency of non-shared rides, vehicle sharing is certainly a direction to pursue. Intelligent Transport Systems (ITS) are promising for modern cities aiming at providing innovative services. Cloud and mobile computing [1] makes it possible for traffic resource sharing between strangers. Researchers have proposed methodologies and incentives for carpool or ridesharing to share trips with low-cost and fun. With the advances in area of autonomous cars and connected vehicles, autonomous transportation services will become possible in the near future. Google has built a driverless or autonomous car, which may be used in traffic system [2]. Electric vehicles are more and more popular and may be greatly used in future transportation systems, therefore the charing-scheduling problem [3] become an important issue. We propose such a system, Public Vehicle (PV) system, to provide convenient and low-cost transportation service. In this system, public-owned vehicles will serve people on demand. A PV can pick up or drop off people at required locations upon requests. The price will be much less than cars or taxis, depending on travel distance, sharing demands, and advanced requests. The system has several advantages compared to current taxi or bus system. It is of low cost compared to taxis. It provides a more convenient solution to the last mile problem compared with buses, because people do not need to walk to stops or transfer. With well-designed incentives, this platform will improve traffic efficiency, reduce the energy consumption and lower the cost of current transportation systems. Recent years, ridesharing systems, *e.g.*, Uber [4], become popular in many countries, while they are not as dynamic as PV system.

Current ridesharing [5] or carpool [6] provide services for people with common subpaths (similar origins and destinations). Zhang *et al.* propose one method [7] to reduce total trip distance, and a fare model prompting passengers to join. Nevertheless, there is one geographical constraint: the carpool starts at the locations where passengers are together, *e.g.*, airports or stations, and the destinations should be close. Certainly, it is important to build a model [8] to analyze transportation systems with presence of uncertainty. Dutta *et al.* introduce a solution of calibration to obtain optimal parameters when creating a reliable model [9]. Ma *et al.* propose a taxi searching algorithm [11] to retrieve candidate taxis that are likely to satisfy a user query. Dial-a-ride problem (DARP) [10] is proposed to construct vehicle routes for people, and some settings should be satisfied: the time windows, precedence constraints, and capacity.

Multi hop ride sharing (MHRS) is proposed to combine multiple rides to improve the ride sharing solution [12]. However, the comfort of passengers would be greatly reduced with too many transfers.

Some solutions of the above are based on personal decisions, which may be far from the good/optimal solutions. PV system has several constraints compared with traditional carpool/ridesharing or taxi sharing, and it is not based on personal decisions. The paths of PVs are calculated by our

proposed algorithm. The contribution of this paper is: (1). We propose a new type of vehicles, public vehicles to improve traffic efficiency and reduce congestions on traffic networks. (2). We formulate the PVP problem under several constraints, *e.g.*, capacity, precedence, and prove it is NP-Complete. (3). Then we introduce a practical algorithm, through which requests can be tackled efficiently. (4). We analyze the performance of PVs and taxis through large-scale simulations with 100,000 requests of one day.

The rest of this paper is organized as follows: In section II, we introduce the scenario of PV system and then formulate the PVP problem. Section III proves the NP-Completeness of this problem. Section IV describes the proposed algorithm. Section V shows the simulations of PVs and taxis, and then analyzes the performance. Section VI concludes this work.

## II. Public Vehicle and Problem Formulation

In this section, we first introduce public vehicle, then analyze the difference between PVs and carpool. Finally, we propose a linear programming solution under practical constraints.

### A. Public Vehicle

Conceptually, PV system keeps the flexibility and speed of private cars or taxis with lower cost at the expense of a little inconvenience (picking or dropping others). The scenario of PV system is shown by Fig. 1: in the traffic networks, some users need trips with pickup points (origins) and dropoff points (destinations). A user sends requests through a smart phone to a central server. The server assigns some PVs to pick up these users, and schedule optimized paths for PVs. Then PVs move according to updated paths. If the paths of PVs are not well designed, persons may spend more time on waiting or traveling on roads. Thus, how to devise the paths to reduce the whole travel distance and meet the demands of users are a topic we are focusing on, which is named as the PV Path (PVP) problem. We suppose there are no transfers between PVs. For the PVs already having persons inside, PVs have to arrive at the destinations of these persons. At the same time, they may pickup new persons and change their paths dynamically following the commands of server.

PV system is one new type transportation system, which is different from traditional carpool or carpooling. Carpool is a personal decision based on plenty of factors, *e.g.*, trip length, travel time, number of participants. To start a carpool, a driver needs to communicate with other potential participants and negotiate with them to get an agreement. PV system is based on the decisions of the central server, which make decisions for all the PVs and persons who need service, and devise new paths for PVs dynamically.

### B. Problem Formulation

Assume at time $t_s$, there are $N_p$ PVs, denoted by set $P$, and total $m$ requests, denoted by $R = R_1 \bigcup R_2$, where $R_1$ is a set of requests currently being served by PVs and $R_2$ is a set of requests to be assigned to PVs. Let $m_1 = |R_1|$, and $m_2 = |R_2|$. Temporarily, assume each request is corresponding to one passenger. Let $r$ denote a request $r \in R$, and $p$ denote a PV $p \in P$. For $R_1$, since requests are currently being served, their origins are not meaningful any more but their destinations need to be reached. Thus, let $V_d^1$ to be a set of the destinations of requests existing in $R_1$. For $R_2$, both origins and destinations of requests need to be considered. Thus, for requests in $R_2$, let $V_o^2$ and $V_d^2$ be a set of their origins and destinations, respectively. Consider a weighted complete graph, $G = (V, E), V = V_p \bigcup V_o^2 \bigcup V_d$, and $E$ is a set of edges between two vertices of $V$. $V_p$ is the set of locations of PVs. $V_d = V_d^1 \bigcup V_d^2$. The weight of edge between any pair of vertices $i$ and $j$, is $d_{i,j}$, the traveling distance based on the shortest path between them. To describe our problem, the other variables are summarized in Table I.

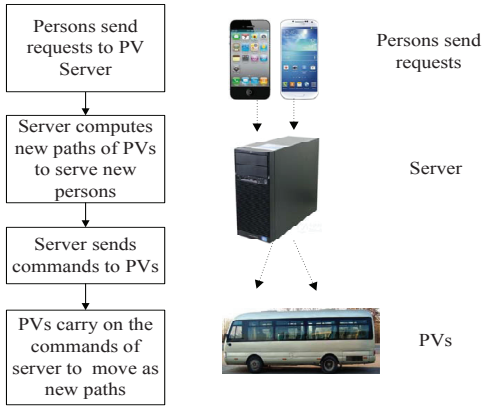


Fig. 1. PV frame.

TABLE I
Denotations in PV system

| | |
|---|---|
| $K_{p,i,j}$ | is 1, if both $i$ and $j$ are on the path of p, and $i$ precedes (not necessarily immediately) $j$, otherwise, 0. |
| $I_p$ | the set of locations p traverses (current location of p is not included). |
| $I_{p,i}$ | is 1, if $i$ is on path of p, otherwise, 0. |
| $P$ | set of PVs. |
| $c_p$ | capacity of PVs. Assume $N_p * c_p \geq m$. |
| $a_p$ | current location of p. |
| $b_p$ | last location of p traverses. |
| $r_e$ | the earliest start time of r |
| $\mu_p$ | a set of requests currently being served by p. |
| $e_p$ | the number of requests currently being served by p. $e_p = \|\mu_p\|$, $\sum_{p=1}^{N_p} e_p = m_1$. |
| $\tau_{p,i,j}$ | travel time on path of p from vertex $i$ to $j$. |
| $Q_{p,r}$ | is 1, if p serves r, otherwise, 0. |
| $x_{p,i,j}$ | is 1, if p traverses edge $(i,j) \in E$, and $i$ precedes $j$ immediately, otherwise, 0. |
| $f_{p,i}^+$ | number of picked requests of p before vertex $i$. |
| $f_{p,i}^-$ | number of dropped requests of p before vertex $i$. |
| $g_{p,i}^+$ | is 1 if p picks one request at vertex $i$, otherwise, 0. |
| $g_{p,i}^-$ | is 1 if p drops one request at vertex $i$, otherwise, 0. |

The formulation of PVP is shown by Eqns (1-14), where Eqn (1) is the objective function, and Eqns (2-13) are constraints. Eqn (1) means the sum of travel distance of each PV. Eqns (2-3) detail the matching between PVs and requests: Eqn (2) ensures that, if r is currently being served by p, p

has to complete serving r by traversing through its destination $r_d$. Eqn (3) ensures that there are exactly $m$ one-to-one assignments among PVs and requests. Eqn (4) implies that, if r is yet to be served, and it has been assigned to one PV p, p has to traverse through its origin and later on its destination. Eqn (5) denotes the total number of picked or dropped passengers when p traverses the edge $\{i, j\}$. Eqns (6)(7) imply that, if p traverses the origin of one request it will pick him up, and if the destination, it will drop him off. Eqns (8-12) are similar to linear programming formulation of PCTSP (precedence constrained traveling salesman problem). Eqn (8) implies that, any location on the path of p except the last one ($b_p$) has only one successor. Eqn (9) implies that, any location on the path of p except the first one ($a_p$, current location of p) has only one precursor. Eqn (10) points out the relationship between $K_{p,i,j}$ and $x_{p,i,j}$: $K_{p,i,j}$ is not less than $x_{p,i,j}$, which can be inferred from our definitions. Eqn (11) details two cases: If neither edge $\{i, j\}$ nor $\{j, i\}$ is not on path of p, $K_{p,i,j} = K_{p,j,i} = 0$. If edge $\{i, j\}$ or $\{j, i\}$ is on the path of p, only either one of $K_{p,i,j}$ and $K_{p,j,i}$ is 1, and the other is 0. Eqn (12) prevents the occurrence of subtours. Eqn (13) implies the constraint of capacity of PVs. Eqn (14) implies the constraint of the earliest start time.

$$\text{Objective: } \min \sum_{p\in P} \sum_{\{i,j\}\in E} d_{i,j} x_{p,i,j} \tag{1}$$

Subject to:

$$r \in \mu_p \Rightarrow Q_{p,r} = 1, I_{p,r_d} = 1 \tag{2}$$

$$r \in R, \sum_{p\in P} Q_{p,r} = 1 \tag{3}$$

$$r \in R_2, Q_{p,r} = 1 \Rightarrow I_{p,r_o} = 1, I_{p,r_d} = 1 \\ K_{p,r_o,r_d} = 1 \tag{4}$$

$$x_{p,i,j} = 1 \Rightarrow f^+_{p,i} + g^+_{p,i} = f^+_{p,j}, f^-_{p,i} + g^-_{p,i} = f^-_{p,j} \tag{5}$$

$$i \in \{I_p \bigcap V_o^2\} \Rightarrow g^+_{p,i} = 1 \tag{6}$$

$$i \in \{I_p \bigcap V_d\} \Rightarrow g^-_{p,i} = 1 \tag{7}$$

$$i \in \{I_p \bigcup \{a_p\} \backslash \{b_p\}\} \Rightarrow \sum_{j\in I_p, j\neq i} x_{p,i,j} = 1 \tag{8}$$

$$j \in I_p \Rightarrow \sum_{i\in I_p \bigcup \{a_p\}, i\neq j} x_{p,i,j} = 1 \tag{9}$$

$$K_{p,i,j} \geq x_{p,i,j} \tag{10}$$

$$K_{p,i,j} + K_{p,j,i} \leq 1 \tag{11}$$

$$K_{p,i,j} + K_{p,j,j'} + K_{p,j',i} \leq 2 \tag{12}$$

$$i \in I_p \Rightarrow e_p + f^+_{p,i} - f^-_{p,i} + g^+_{p,i} - g^-_{p,i} \leq c_p \tag{13}$$

$$r \in R_2, Q_{p,r} = 1 \Rightarrow t_s + \tau_{p,a_p,r_o} \geq r_e \tag{14}$$

$$f^+_{p,i}, f^-_{p,i} \geq 0. \;\; x_{p,i,j}, K_{p,i,j}, K_{p,j,j'}, K_{p,j',i}, \\ Q_{p,r}, g^+_{p,i}, g^-_{p,i}, I_{p,r_o}, I_{p,r_d} \in \{0, 1\}$$

## III. NP-Completeness

In this section, we prove the NP-Completeness of the PVP problem by reducation from the TSP problem.

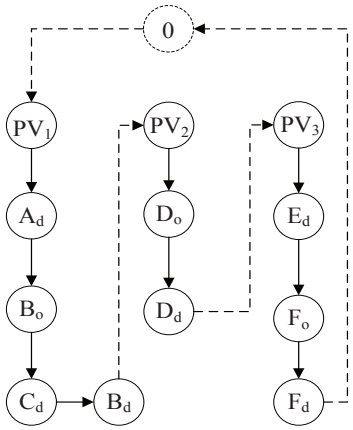


Fig. 2. Transformed graph of PVP.

**Theorem III.1.** *The PV Path (PVP) problem is NP-Complete, as it can be reduced from Traveling Salesman Problem (TSP).*

*Proof:* First, we show that PV Path (PVP) problem is NP. Given any instance of PVP, we use as a certificate the paths and tasks of all PVs. Here, tasks include picking/dropping locations of any request. The verification algorithm checks that any request $r \in R$ is served by $p \in P$. We check whether all the requests can arrive at their destinations through the paths and service tasks of PVs. Clearly, this process can be done in polynomial time. Therefore, PVP belongs to NP.

Second, we prove that PVP is NP-Hard by showing that it can be reduced from the Traveling Salesman Problem (TSP), which is NP-Hard. We construct a new graph $\tilde{G}$, which is shown by Fig. 2. First, we introduce a virtual vertex (denoted by 0) and connect it to all the PVs (denoted by $PV_1, PV_2, PV_3$) corresponding to a salesman with zero-cost edges. Points in TSP are mapped to the locations of PVs, origins and destinations of requests. Assume that a virtual salesman first goes to the location of $PV_1$. For requests $\mu_1 = \{A, C\}$ already in $PV_1$, the salesman has to reach the corresponding destinations $A_d$ and $C_d$. In addition, the salesman will take more requests and assign $PV_1$ to serve them. In Fig. 2, B is selected with its origin and destination $B_o$ and $B_d$, resulting in an updated path $\{PV_1, A_d, B_o, C_d, B_d\}$. Then the salesman moves to another PV $PV_2$, and so on.... This process repeats until all the requests are served.

At last, the salesman moves back to his origin (denoted by 0), and the corresponding cost is zero. In Fig. 2, all the edges with zero costs are denoted by dash lines. At last, the edges (dash lines) with zero costs are eliminated to reduce to the PVP problem. Therefore, for any instance of TSP, a corresponding instance in PVP can be mapped, and the reduction is polynomial. The solution to the instance in PVP is also the solution to TSP. Thus, PVP is NP-Complete. ■

## IV. Heuristics

In this section, we propose our heuristic algorithm and analyze its approximation ratio.

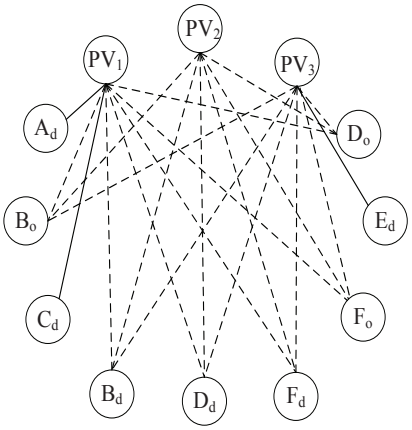


Fig. 3. Assignment graph constituted by PVs, origins and destinations of requests.

### A. Origin-Destination Pair Insertion

If we design paths for any request which is not yet being served by PVs, the precedence between origin and destination both should be considered. First, we introduce an assignment graph (Fig. 3) which consists of PVs, origins and destinations of requests, where $R_1 = \{A, C, E\}$ and $R_2 = \{B, D, F\}$. More specifically, $\mu_1 = \{A, C\}$, $\mu_3 = \{F\}$, and origins of requests in $R_1$ are excluded since they are not in a range of interesting. Requests in $R_2$ can be served by any of PVs, $PV_1$, $PV_2$, or $PV_3$. In general, for the requests in set $\mu_p$ of p (by definition, they exist in $R_1$), there is no precedence among their destination locations. On the other hand, for a new request $r \in R_2$, its origin $r_o$ has to be visited before its destination $r_d$. Therefore, if p decides to take request $r \in A_2$, both $r_o$ and $r_d$ need to be inserted into the current path of p with the precedence constraint being satisfied.

**Definition IV.1.** *The insertion cost (IC) at location of (i,j) $\pi_{r,p,i,j}$. Assume r is taken by p. On the current path of p, insert one origin-destination pair $(r_o, r_d)$ of r, where $r_o$ precedes $r_d$, leading to a new path. The cost $\pi_{r,p,i,j}$ is the difference between the distance of the two paths.*

**Definition IV.2.** *The least insertion cost (LIC) $\pi_{r,p} = \pi_{r,p,i',j'}$ will be the least cost for all possible insertion locations of $i$ and $j$ of p for request r.*

**Definition IV.3.** *The minimum insertion cost (MIC) $\pi_r = \pi_{r,p'} = \pi_{r,p',i',j'}$ will be the minimum cost of request r for all possible PVs in $P$.*

For p, let $list_p$ be the list of requests either currently being served or will be served by it. Now, let the current path $\{\theta_0, \theta_1, \ldots, \theta_{L_p}\}$, consisting of its current location $\theta_0 = a_p$. In order to insert new request r, select one location $\theta_i$ $(1 \le i \le L_p + 1)$ of path of p to insert $r_o$ in front of $\theta_i$. Then from the locations after $r_o$ select another location $\theta_j$ $(i + 1 \le j \le L_p + 2)$ to insert $r_d$. Thus, there are $(L_p - i + 2)$ locations to select $r_d$ for every selected $r_o$. The insertion complexity is $O(L_p^2)$.

**Algorithm 1** Algorithm of PCPI

**Input:**
$R_1, R_2, P$.
**Output:**
Paths of PVs, $\chi = \{\chi_p | p \in P\}$.
1: **for** $p \in P$ **do**
2: $list_p \leftarrow \mu_p$.
3: Calculate the initial path $\chi_p$ through $\mu_p$ using Christofides's algorithm [13], then remove duplicate points.
4: **end for**
5: **for** $r \in R_2$ **do**
6: **for** $p \in P$ **do**
7: Use Eqn (15) to calculate IC $\pi_{r,p,i,j}$ at all possible locations $\{i, j\}$ on the path $\chi_p$ of p. Then obtain LIC $\pi_{r,p} = min(\pi_{r,p,i',j'})$.
8: **end for**
9: From the LIC of $i$ in every PV calculate MIC $\pi_r = \pi_{r,p'} = min\{\pi_{r,1}, \pi_{r,2}, \ldots, \pi_{r,N_p}\}$.
10: **if** $\pi_r \neq \infty$ **then**
11: Insert origin $r_o$ and destination $r_d$ of r at the locations of $i'$ and $j'$ on the path $\chi_{p'}$ of p'.
12: $list_{p'} \leftarrow list_{p'} \bigcup \{r\}$. // Schedule request r in the list of p'.
13: **end if**
14: **end for**

Let $d\{\theta_{i-1}, \theta_i\}$ denote the shortest distance between $\theta_{i-1}$ and $\theta_i$ on road networks. Let $d\{\theta_i, \ldots, \theta_j\} = d\{\theta_i, \theta_{i+1}\} + \ldots, +d\{\theta_{j-1}, \theta_j\}$ denote the distance of shortest path from $\theta_i$ to $\theta_{i+1}$,... and to $\theta_j$.

$$\pi_{r,p,i,j} = \begin{cases} d\{\theta_{i-1}, r_o, r_d, \theta_i\} - d\{\theta_{i-1}, \theta_i\}, \\ \qquad \text{if } 1 \le i \le L_p, j = i + 1. \\ d\{\theta_{L_p}, r_o, r_d\}, \\ \qquad \text{if } i = L_p + 1, j = i + 1. \\ d\{\theta_{i-1}, r_o, \theta_i\} + d\{\theta_{L_p}, r_d\} - d\{\theta_{i-1}, \theta_i\}, \\ \qquad \text{if } i \le L_p, j = L_p + 2. \\ d\{\theta_{i-1}, r_o, \theta_i\} + d\{\theta_{j-1}, r_d, \theta_j\} \\ -d\{\theta_{i-1}, \theta_i\} - d\{\theta_{j-1}, \theta_j\}, \\ \qquad \text{if } i \le L_p, j \le L_p + 1, j \neq i + 1. \end{cases} \quad (15)$$

### B. Algorithm

Now, we consider an ideal scenario without PV capacity and passengers' detour constraints, which is named as $S_I$. And $S_{II}$ is the scenario with the two constraints. First, we discuss solution of $S_I$.

PCPI (Precedence Constrained origin-destination Pair Insertion) Algorithm is shown in Algorithm 1. For p, initial path $\chi_p$ denotes the path constituted by its current location and the destinations of requests in $\mu_p$. Lines (1-4) compute the initial path of each PV. Lines (6-8) calculate the least cost of inserting r of by p'. Lines (5-14) get the minimum cost of

serving r among all the PVs, and then insert the origins and destinations of r at appropriate locations. Line (10) means that, if the cost of serving r is infinite, r is denied.

The operation of inserting origins and destinations is carried out by line (11) and the corresponding path should be updated afterwards. We know that the complexity of insert one origin-destination pair on path of p is $O(L_p^2)$. From lines (1-4), the complexity is $nc_p^3$. From lines (5-14), the complexity is $O(m_2 N_p c_p^2) = O(mN_p c_p^2)$. Generally, $m \geq c_p$. Finally, the complexity of PCPI is $O(mN_p c_p^2)$.

**Theorem IV.1.** *Approximation ratio of PCPI is 3.5.*

*Proof:* Let $d'_p$ denote the distance of initial path for p calculated by line (3) in Algorithm 1, and $d' = \sum_j d'_p$. Let $d''_p$ denote the total MIC when p serves additional requests outside ($d''_p = \sum_{r \in R_2} \pi_{r,p}$), and $d'' = \sum_p d''_p$. Let $d_{OPT}$ denote the distance of optimal solution. The approximation ratio of Christofides's algorithm is 1.5, and clearly, $d' \leq 1.5 d_{OPT}$.

Now, we discuss $d''$. D. Rosenkrantz *et al.* have proved approximation ratio of cheapest insertion and nearest merger methods [14] is 2. In fact, we can name PVP as a multiple-cycle TSP: we add a virtual point to denote the root, and the origin and destination of any request consist of a cycle. Set the weight of the arc returning to virtual point zero. Bockenhauer *et al.* point out that, the approximation ratio of Algorithm 4 in [15] is not worse than 2. We can get $d'' \leq 2 d_{OPT}$. Let $d_T$ denote the finally total distance of PV paths. We get $d_T = d' + d'' \leq 3.5 d_{OPT}$. ∎

Then we discuss the solution for the other scenario $S_{II}$. With respect to r, the travel distance $d_r$ is not less than $d_r^s$, the shortest distance from its origin to destination. $Detour(r)$ (detour ratio of r) is the percentage of additional distance compared with $d_r^s$, *i.e.*, $Detour(r) = \frac{d_r - d_r^s}{d_r^s}$. Let $Detour_{avg}$ denote the average detour ratio of all requests: $Detour_{avg} = \frac{\sum_r d_r - \sum_r d_r^s}{\sum_r d_r^s}$. Make sure that the detour ratio of any request does not exceed a threshold $DetourTh$: $Detour(r) \leq DetourTh$.

Let $n_{r,p,i,j}^{max}$ denote the maximum number of passengers among all locations if p picks request r at location $i$ and drops it at $j$. Let $\pi'_{r,p,i,j}$ of $S_{II}$ replace $\pi_{r,p,i,j}$ in $S_I$, and the relationship between them is shown by Eqn (16). If p serves request r, there will be no seat for him, or there would appear large detour for other requests in schedule list of this PV. This request would not be selected by p.

$$\pi'_{r,p,i,j} = \begin{cases} \infty, \text{ if } n_{r,p,i,j}^{max} > c_p \\ \quad \text{or } Detour(r) > DetourTh \\ \pi_{r,p,i,j}, \text{ otherwise.} \end{cases} \quad (16)$$

## V. Performance Evaluation

In this section, we evaluate the performance of the proposed algorithm compared with taxis.

Our simulation is based on the road networks of Shanghai in China, particularly, in the downtown area of about 50 $km^2$. Within in a 24-hour time window, there are 100,000 trip requests generated and geographically distributed in that area. We use Shanghai daily traffic characteristics [16] to configure our peak and nonpeak traffic patterns. Fig. 4 shows the distribution of trip requests by time of the day (24 hours). For simplicity, we assume there is a single passenger for every trip request. The simulations run until all the trip requests complete, therefore, will extend certain minutes beyond 24 hours.

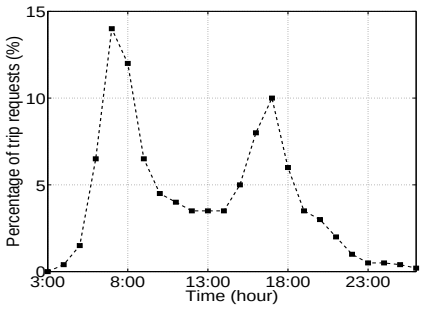


Fig. 4. Distribution of trip requests by time of the day.

A taxi also follows the shortest path to pick the next assigned request (as unloaded) and then to reach its destination (as occupied). With respect to the shortest path algorithm, we use Dijkstra Algorithm. Up on arrival of destination, a taxi will stop there, remaining as its on-call status, until the next trip request being scheduled. Thus, taxis have three statuses: occupied, on-call, and unloaded. Similarly, a PV also has three types of statuses as a taxi does, but will use our PCPI algorithm to determine its paths. $c_p$ is set to 15. Considering of the comfort of passengers, $DetourTh$ is set to 0.2. Table II lists the variables and definitions used in simulations. Here, we do not consider traffic congestion, and PVs or taxis travel as the speed we set all the time. Finally, we put 500-1,500 PVs to the system to compare the performance with taxis.

TABLE II
VARIABLES USED IN SIMULATIONS

| Variables | Definition | Values |
|---|---|---|
| $s_t$ | speed of taxis. Unit: km/h. | 40 |
| $s_p$ | speed of PVs. Unit: km/h. | 40 |
| $N_t$ | total number of taxis. | 2,300 |
| $c_t$ | capacity of taxis. | 4 |
| $t_p$ | time spent in picking one request. Unit: second. | 6 |
| $t_d$ | time spent in dropping one request. Unit: second. | 6 |
| $n_{sh}$ | number of passengers sharing one request. | 1∼4 |
| $d_{lb}$ | lower bound of distance from origin to destination. Unit: km | 5 |

### A. Results

For time being, let number of passengers per request, $n_{sh}$, be 1, and leave discussion of its general values later.

In Fig. 5, the black bars denote the waiting time, and white bars denote the travel time. Here, take $N_t = 2,300$ as a benchmark, and the total time is about 20 minutes. We see that, if more PVs join to serve passengers, the average waiting

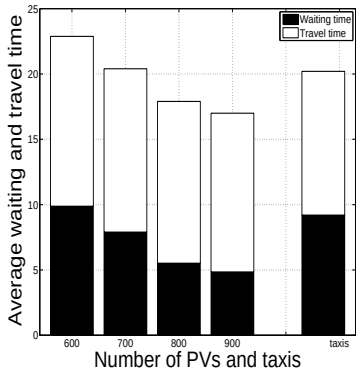


Fig. 5. Waiting and travel time for PVs and taxis.

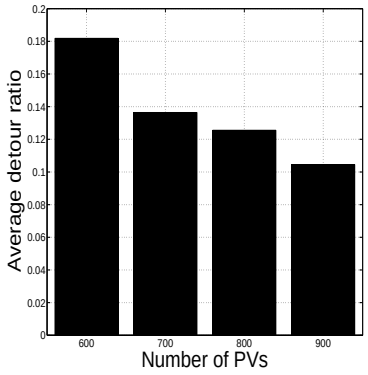


Fig. 6. Average detour ratio of requests.

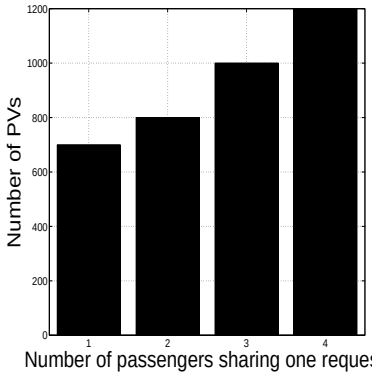


Fig. 7. Required number of PVs when $n_{sh}$ varies.

and travel time decrease. The total distance of taxis is $8.2\times10^5$ km, and distance of 700 PVs is about $3.3 \times 10^5$ km, reduced by 60%. Fig. 6 denotes the average detour ratio of passengers. If more PVs join the transportation, passengers can enjoy less detour distance. In our algorithm, considering of comfort of passengers, the detour ratio of any request should not exceed the threshold we set.

At last, we discuss the scenario when more passengers share one request. When $n_{sh}$ = 2, 3, and 4, we put 500-1,500 PVs to the system and compare their performances with taxis. The result is shown by Fig. 7. If more passengers share one request, more PVs should join to achieve corresponding performance (total time, waiting time plus travel time). When 4 passengers share one request, the performance of PVs is not as well as the one with 2 or 3 passengers. For $N_t = 2,300$, the number of vehicles in PV system is reduced by 47%-69% compared with taxis when $n_{sh}$ varies from 1 to 4.

## VI. Conclusion

In this paper, we propose one public vehicle system to improve efficiency of conventional carpool and taxi system. To reduce the total travel distance, we introduce a solution through linear programming method. Then we propose an algorithm to design paths for PVs to serve user requests. Through large simulations, we find the number of vehicles can be greatly reduced using PV system compared with taxis. Therefore, heavy traffic congestions in modern cities can be mitigated. The proposed algorithm is efficient, real time, and can be practical in the future traffic systems. In this paper, congestion is not considered, and in future, we would build better simulations showing the effects of traffic congestion, which would be more actual than this one, and better compare the performance between PV system and other transportation systems.

## Acknowledgment

This work was supported by the Natural Science Foundation of China (NSFC) projects (Nos. 61373155, 91438121, 61303202 and 61373156), the Key Basic Research Project (No. 12JC1405400), China Postdoctoral Science Foundation (2014M560334 and 2015T80433) and the Shanghai Pujiang Program (No. 13PJ1404600) of the Shanghai Municipality.

## References

[1] S. Bitam and A. Mellouk, "Its-cloud: Cloud computing for intelligent transportation system," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 2054–2059, 2012.

[2] "Roadtesting google's new driverless car." http://www.telegraph.co.uk/motoring/11382073/Roadtesting-Googles-new-driverless-car.html.

[3] M. Zhu, X.Y. Liu, L. Kong, R. Shen, W. Shu, and M.Y. Wu, "The charging-scheduling problem for electric vehicle networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, p-p. 3178–3183, 2014.

[4] https://www.uber.com/.

[5] M. Furuhata, M. Dessouky, F. Ordóñez, M.-E. Brunet, X. Wang, and S. Koenig, "Ridesharing: The state-of-the-art and future directions," *Transportation Research Part B: Methodological*, vol. 57, pp. 28–46, 2013.

[6] G. Correia and J. M. Viegas, "Carpooling and carpool clubs: Clarifying concepts and assessing value enhancement possibilities through a stated preference web survey in lisbon, portugal," *Transportation Research Part A: Policy and Practice*, vol. 45, no. 2, pp. 81–90, 2011.

[7] D. Zhang, Y. Li, F. Zhang, M. Lu, Y. Liu, and T. He, "coride: carpool service with a win-win fare model for large-scale taxicab networks," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, p. 9, 2013.

[8] A. Guasch, J. Figueras, C. Montañola-Sales, J. Casanovas-Garcia, *et al.*, "Simulation analysis of a dynamic ridesharing model," in *IEEE Proceedings of the Winter Simulation Conference*, pp. 1965–1976, 2014.

[9] P. Dutta, E. Arnaud, E. Prados, and M. Saujot, "Calibration of an integrated land-use and transportation model using maximum-likelihood estimation," *IEEE Transactions on Computers*, vol. 63, no. 1, pp. 167–178, 2014.

[10] M. Firat and G. J. Woeginger, "Analysis of the dial-a-ride problem of hunsaker and savelsbergh," *Operations Research Letters*, vol. 39, no. 1, pp. 32–35, 2011.

[11] S. Ma, Y. Zheng, and O. Wolfson, "T-share: A large-scale dynamic taxi ridesharing service," in *IEEE International Conference on Data Engineering (ICDE)*, pp. 410–421, 2013.

[12] F. Drews and D. Luxen, "Multi-hop ride sharing," in *Sixth Annual Symposium on Combinatorial Search*, 2013.

[13] N. Christofides and S. Eilon, "Algorithms for large-scale travelling salesman problems," *Operational Research Quarterly*, pp. 511–518, 1972.

[14] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, II, "An analysis of several heuristics for the traveling salesman problem," *SIAM journal on computing*, vol. 6, no. 3, pp. 563–581, 1977.

[15] H.J. Böckenhauer, J. Hromkovič, J. Kneis, and J. Kupke, "On the approximation hardness of some generalizations of tsp," in *Algorithm Theory–SWAT*, pp. 184–195, Springer, 2006.

[16] http://sh.eastday.com/qtmt/20110309/u1a863059.html.