Hindawi
Journal of Advanced Transportation
Volume 2020, Article ID 3271608, 12 pages
https://doi.org/10.1155/2020/3271608

Research Article

An Efficient Solving Method to Vehicle and Passenger Matching Problem for Sharing Autonomous Vehicle System

Ming Li,[1] **Nan Zheng**,[2,3] **Xinkai Wu**,[1,3] **Weihua Li**,[4] **and Jianhua Wu**[4]

[1]*School of Transportation Science and Engineering, Beihang University, Beijing 100191, China*
[2]*Institute of Transport Studies, Department of Civil Engineering, Monash University, Australia*
[3]*Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100191, China*
[4]*Institute of Rail Transportation of Jinan University, Electrical and Information College of Jinan University, Zhuhai 519070, China*

Correspondence should be addressed to Xinkai Wu; xinkaiwu@buaa.edu.cn

Received 20 May 2019; Accepted 15 October 2019; Published 10 February 2020

Academic Editor: Dongjoo Park

With the potential of increasing mobility and reducing cost, shared mobility of autonomous vehicles (AVs) is going to gain solid growth in the coming decade. The major issue for the shared use of AVs is how to project serving routes in an efficiently way. From another perspective, this issue could be understood as to segment maximum number of passengers into groups. Therefore, this paper intends to investigate passengers' similarity instead of directly matching AVs and passengers. The goal is to determine the minimum number of groups and assign each group with an AV. To this end, a cluster-based algorithm is proposed to classify passengers. Numerical experiments with both small-size and large-size demands are performed to present the validity of the proposed algorithm. Results indicate that the cluster-based algorithm could bring benefit to minimizing the number of vehicles and total travel distance. At last, sensitivity analysis of key parameters shows that vehicle capacity will have little impact when the number of seats exceeds four, and time windows could make continuous influence on gathering passengers.

1. Introduction

Autonomous vehicles (AVs) are regarded as a promising mode of transportation that provides increased mobility, enhanced customer satisfaction, and reduced infrastructure costs (e.g., [1–3]). For the autonomous transportation system, using shared AVs (SAVs) is considered as one of the most efficient way to provide on-demand service. According to the number of passengers carried at one time, existing researches related with SAVs could be classified into two categories, i.e., car-sharing and ride-sharing.

In the car-sharing mode, each SAV only takes one passenger at one time. Different from traditional car-sharing system (e.g., Boyaci et al. [4], Huang et al. [5]), the major concern for SAV system is how to assign SAVs to provide point-to-point service instead of determining depot locations and relocate AVs among stations for balancing supply and demand (e.g., Ma et al. [6], Levin et al. [7], Chen et al. [2], Miao et al. [8]). In the ride-sharing mode, multiple passengers are permitted to ride in one car simultaneously. If the system control platform could match efficiently passengers and vehicles, then the overall vehicle occupancy will be increased and traffic congestion will be released through cutting down on-road vehicles. Therefore, this paper will focus on how to build a control platform to make passengers sharing a car.

For modeling the ride-sharing service, most researches refer to the classical Dial-a-Ride Problem (DARP) model proposed by Cordeau [9]. The essential idea of DARP is to extract the routing problem into the arc-based flow conservation model. In this way, vehicle routes could be decoded by linking visited arcs. Although this method could illustrate the routing problem with a mixed integer model, its drawback is inevitable to generate valid arcs for identical requests. Meanwhile the number of variables for this model would be raised exponentially with increased passengers. To solve this model in an efficient way, efforts have been devoted on exploring optimal solutions. One of the prevailing algorithms is the branch-and-cut method, which aims at eliminating invalid arcs and cutting redundant domains, e.g., Liu et al. [10], Bongiovanni et al. [11]. Except for it, other

methods are also tried to find optimal routes. For example, Hosni et al. [12] adopted the Lagrangian decomposition method to segment the multivehicles routing problem into several single vehicle. Mahmoudi and Zhou [13] combined the forward dynamic programming and Lagrangian relaxation approach to search feasible solutions. Cordeau and Laporte [14] applied a Tabu search heuristic to solve the DARP and proposed a framework of neighborhood evaluation. Häme and Hakula [15] proposed a heuristic algorithm that maximized the number of customers served by a single vehicle. Diana and Dessouky [16] proposed an insertion heuristic to handle the computational complexity and investigated the relationship between solution quality and computational costs.

However, most existing methods try to explore detailed vehicle trajectories. This poses difficulties in searching feasible paths for increased demands. Vazifeh, et al. [17] transferred the dispatching of vehicles as a minimum path cover problem on directed graphs and tested graph algorithms with massive taxi data of New York City. Based on this method, it might be powerful to solve the vehicle and passenger matching problem by investigating the similarity of passenger trip trajectories and classifying passengers into groups. Our idea herein is to simplify the researched problem by directly exploring the relationship between passengers. In this way, passengers can be classified into several groups. Then by checking the connection between groups and assigning each group with one vehicle, the number of vehicle needed can be derived. With this thought, we will propose an algorithm for tackling various passengers. In summary, the main objective of this paper is: (i) simplify the researched problem by linking passengers instead of directly matching vehicles and passengers; and (ii) design a cluster-based algorithm to find solutions for high traffic demands with high efficiency.

The rest of the paper is organized as follows. In Section 2, the vehicle dispatching model for SAV-based transportation systems is proposed Section 3 presents an efficient heuristic algorithm developed for the proposed model. In Section 4, comparative results of various scenarios are discussed. Finally, Section 5 presents some conclusions with remarks.

2. The Dispatching Problem of an AV-Based Transportation Service System

This paper mainly discusses how to plan AVs serving routes and determine the number of dispatched vehicles for the SAV transportation system. An optimization model is developed for matching vehicles and passengers. The details of the proposed model are descripted in this section.

2.1. Problem Statement. For the SAV system, a major problem is to optimize vehicle serving paths with the consideration of saving the number of vehicles. In general, passengers are characterized by origin-destination (OD) pairs and time windows. In detail, let U be the set of all requests where $o(u)$ and $d(u)$, $\forall u \in U$, respectively represent requested pickup (origin) and delivery (destination) nodes. For these requests, the earliest and latest pickup time indexes are $t^e_{in}(u)$ and $t^l_{in}(u)$, $\forall u \in U$ and the earliest and latest drop-off time are $t^e_{out}(u)$ and $t^l_{out}(u)$, $\forall u \in U$. Based on passenger requests, a directed graph $G = (A, E)$ could be depicted, where A is the set of OD nodes and E is the set of edges associating travel time among nodes. Let TT represents the set of paths travel time in the road network, values of weighted edges in the graph are regarded as travel time, i.e., $TT\ (o(u), d(u)) \cup TT\ (o(u'), d(u)) \cup TT\ (o(u),\ d(u')) \cup TT\ (o(u'),\ d(u')) \in E, \forall u,\ u' \in U$. It should be noted that travel time is estimated as fixed values for simplification. For vehicles, V is the set of available AVs. We assume that AV fleet is homogeneous with respect to capacity which is equal to a preset value C_w. Under this background, the goal of optimizing the AV transportation system is set to minimize the system cost, i.e., the number of used vehicles.

Based on existing researches, an extended DARP model could be modified to express the objective of minimizing vehicle number instead of minimizing the total travel distance. Since the DARP model is a node-flow model, the solution domain is denoted by $4 \times \{V\} \otimes \{U\} \otimes \{U\}$. When available vehicles and user demands are large, numerous time will be exhausted on searching optimal vehicle paths. To reduce the complexity in mathematical modeling and to present it in a simple way, we intend to build the optimization model by only delivering the relationship of passengers, i.e., $\lambda(u, w)$, 1 if passenger u and w are served by the same vehicle, and 0 otherwise. In this way, passengers are divided into groups and the space is reduced to $0.5 \times \{U\} \otimes \{U\}$. Furthermore, detailed vehicle paths could be obtained through translating passenger relationships. For example, given four passengers ($u = 1, 2, 3, 4, \forall u \in U$) waiting for service. If $\lambda(1, 2) = 1$ and $\lambda(1, 4) = 1$, passenger 1, 2, and 4 are carried by the same vehicle and passenger 3 is assigned to another vehicle. This implies that two vehicles in total are required to satisfy these four passengers. Therefore, the whole framework of this researched problem could be concluded as shown in Figure 1. Table 1 lists some important parameters and variables.

2.2. The Optimization Model of Passenger–Passenger Matching. Before formulating the optimization model, some characteristics of this system are predefined as follows:

(i) All passenger requests should be served;
(ii) All passengers are assumed to have the willingness of sharing rides with others;
(iii) All passengers' pick-up and delivery times are within their expected time windows;
(iv) Each passenger is only served by one single vehicle;
(v) Vehicle capacity should not be exceeded when carrying multiple passengers; and
(vi) AVs are controlled by a central control system.

A vehicle carrying more passengers will increase the probability of saving vehicle needed, which means that more passengers are associated. That is to say maximizing the number of associated passengers could take an effect in minimizing the total number of vehicles. In this way, the objective function is formulated by Equation (1)

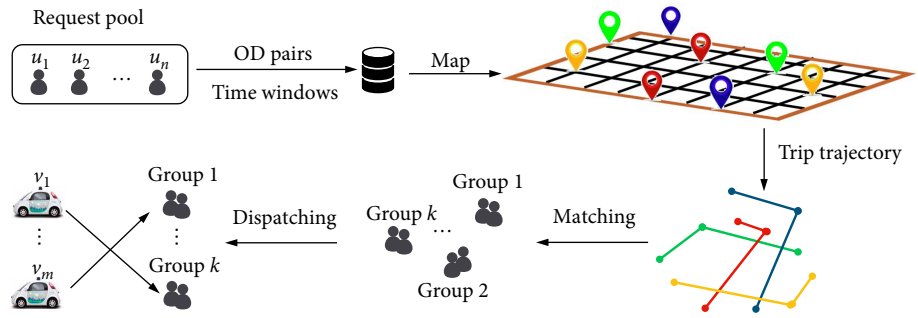


Figure 1: The framework of the AV transportation system.

Table 1: Summary of notations.

Parameters and sets	
A	Set of nodes
E	Set of links
V	Set of AVs
U	Set of passengers
TT	Set of minimum travel time between any two nodes
$t^e_{on}(u)$	The earliest pick-up time
$t^l_{on}(u)$	The latest pick-up time
$t^e_{off}(u)$	The earliest drop-off time
$t^l_{off}(u)$	The latest drop-off time
$o(u)$	Pick-up location of request u
$d(u)$	Drop-off location of request u
C_w	Vehicle capacity
Variables	
$\lambda(u,w)$	Binary variable, 1 if passenger u and w are served by the same vehicle
$p(u)$	The moment when passenger u is picked from the original node
$g(u)$	The moment when passenger u is delivered at the destination

$$F = \max \sum_{u,w \in U} \lambda(u,w), \tag{1}$$

with the following constraints:

(1) The transitivity relation of passengers. To determine the correlation between multiple passengers, the transitivity of passenger relationships is denoted in Equation (2). It means that passengers u' and u'' are connected if they are both linked by a passenger u.

$$2 - \lambda(u,u') - \lambda(u,u'') = \begin{cases} 0 & \lambda(u',u'') = 1, \\ 1 & \lambda(u',u'') = 0 \text{ or } 1, \\ 2 & \lambda(u',u'') = 0. \end{cases} \tag{2}$$

To provide the convenience for solving the model, Equation (2) is transformed with an equivalent expression, as shown in Equation (3):

$$\begin{aligned} &2 - \lambda(u,u') - \lambda(u,u'') \geq 1 - \lambda(u',u'') \\ &\forall u,u',u'' \in U, \quad u \neq u' \neq u''. \end{aligned} \tag{3}$$

(2) Service time constraint. Since the desired pick-up and drop-off times are collected in advance, request service time should be within these time limitations, as reflected in Equations (4) and (5). For request u and w, if they are assigned to the same vehicle ($\lambda(u,w) = 1$), time differences in spatial dimension also need to be added to determine variables $p(u)$ and $g(u)$, which are expressed in Equations (6)–(8). M is a very large number. Equation (6) implies that time difference of taking request u and u' for vehicle v should not be less than the minimum travel time between original node $o(u)$ and $o(w)$. Similarly, Equation (7) illustrates that time difference of dropping off requests u and w need to be larger than the minimum travel time between destination nodes $d(u)$ and $d(w)$. In addition, the time constraint for picking up request u and dropping off w is also considered in Equation (8).

$$t^e_{\text{in}}(u) \leq p(u) \leq t^l_{\text{in}}(u) \;\; \forall u \in U, \tag{4}$$

$$t^e_{\text{out}}(u) \leq g(u) \leq t^l_{\text{out}}(u) \;\; \forall u \in U, \tag{5}$$

$$\left|p(u) - p(w)\right| + (1 - \lambda(u,w)) \cdot M \geq TT(o(u),o(w)) \quad \forall u,w \in U,\; u \neq w, \tag{6}$$

$$\left|g(u) - g(w)\right| + (1 - \lambda(u,w)) \cdot M \geq TT(d(u),d(w)) \quad \forall u,\mu \in U,\; u \neq w, \tag{7}$$

$$\left|p(u) - g(w)\right| + (1 - \lambda(u,w)) \cdot M \geq TT(o(u),d(w)) \quad \forall u,w \in U,\; u \neq w. \tag{8}$$

These nonlinear constraints (Equations (6)–(8)) add increased complexity to the optimization problem. A transformation method proposed by Sexton and Bodin [18] is introduced to cope with the computational issue. The basic idea of this work is to model service time priority by a 0-1 matrix. Hence a binary variable $\overline{\omega}_0(u,w)$ is set to denote whether pick-up time of request u is earlier than request w. In this way, we can determine $p(u) - p(w) \geq TT(o(u),o(w))$ or $p(u) - p(w) \leq -TT(o(u),o(w))$. $\overline{\omega}_0(u,w)$ will be equal to 1 if request w ranks behind request u in the service order, i.e.,

$p(u) \le p(w)$. A linear approximations of Equation (6) are shown in Equations (9) and (10). For example, Equation (7) could be simplified as $p(u) - p(w) \le -TT(o(u), o(w))$, if $\overline{\omega}_0(u,w) = 1$ and $p(u) - p(w) \ge TT(o(u), o(w))$ for Equation (10) when $\overline{\omega}_0(u,w)$=0. In such a way, Equations (7) and (8) deliver exactly the same function as does by Equation (6).

$$p(u) - p(w) + \left(1 - \overline{\omega}_0(u,w)\right) \cdot M \le -TT\left(o(u), o(u')\right) \quad \forall u, w \in U,\ u \ne w, \tag{9}$$

$$p(u) - p(w) + \overline{\omega}_0(u,w) \cdot M \ge TT\left(o(u), o(u')\right) \quad \forall u, w \in U,\ u \ne w. \tag{10}$$

Following the same methodology, we reconstruct Equations (7) and (8) by Equations (11)–(13), where $\overline{\omega}_1(u,w)$ denotes the precedence relation of dropping off request u and w, and $\overline{\omega}_2(u,w)$ compares picking up request u and dropping off request w.

$$g(u) - g(w) + \left(1 - \overline{\omega}_1(u,w)\right) \cdot M \le -TT(d(u), d(w)) \quad \forall u, w \in U,\ u \ne w, \tag{11}$$

$$g(u) - g(w) + \overline{\omega}_1(u,w) \cdot M \ge TT(d(u), d(w)) \quad \forall u, w \in U,\ u \ne w, \tag{12}$$

$$p(u) - g(w) + \left(1 - \overline{\omega}_2(u,w)\right) \cdot M \le -TT(o(u), d(w)) \quad \forall u, w \in U,\ u \ne w, \tag{13}$$

$$p(u) - g(w) + \overline{\omega}_2(u,w) \cdot M \ge TT(o(u), d(w)) \quad \forall u, w \in U,\ u \ne w. \tag{14}$$

(3) Vehicle capacity constraint. To express the status of onboard passengers at a certain time, we discretize the simulation time into uniform intervals and an auxiliary variable $z_1(u,t)$ is used to present the states of requests. If $z_1(u,t) = 1$, it means that request u is being served at current time t. Otherwise, request u is waiting to be picked up or has arrived at its destination. Combining this binary factor $z_1(u,t)$, we can count the total number of onboard passengers at any time t, which is formulated in Equation (15):

$$\sum_{w \in U} \lambda(u,w) \cdot z_1(w,t) - \left(1 - z_1(u,t)\right) \cdot M \le C_w \ \forall u \in U, \forall t \in T, \tag{15}$$

where the state variable $z_1(u,t)$ is an indirect representation on the relation between actual pick-up time $p(u)$ and drop-off time $g(u)$, as shown in Equation (16):

$$z_1(u,t) = \begin{cases} 1 & p(u) \le t \le g(u) \\ 0 & else \end{cases} \quad \forall u \in U,\ \ \forall t \in T. \tag{16}$$

The presence of the nonlinear constraint of (16) makes it difficult to be implemented directly in commercial solvers. For the sake of computation complexity, we conduct linear transformations as by Equations (17)–(23). To accomplish the process, we introduce another auxiliary variable $z_2(u,t)$ associated with request state. Given the pick-up and drop-off times, a request state could be divided into three phases. Firstly, a waiting phase, the request is waiting to be served and pick-up time has not been reached, in which time duration of this type should be earlier than actual pick-up time $p(u)$. Equation (17) should always hold in this situation, $z_2(u,t)$ and $z_1(u,t)$ limited by Equation (22) will be approximated to 1 and 0 respectively to meet Equations (18)–(20), i.e., waiting state when $z_1(u,t) = 0$ and $z_2(u,t) = 1$. In the second onboard phase, a request has been transported to its destination but has not yet arrived. Time duration of this phase should be governed by the pick-up time and ended at drop-off time, i.e., $p(u) \le t \le g(u)$. During the off-board phase, Equations (17) and (20) will hold only if $z_1(u,t) = 1$ and $z_2(u,t) = 0$. The last phase will occur when a request has been completed and time is later than drop-off time $g(u)$. Hence, Equation (20) should always hold for off-line situations. Under this constraint, it can be proved that $z_1(u,t) = 0$ and $z_2(u,t) = 0$ are the only solution to make Equations (17) and (19) satisfied. In this way, it is obvious that Equations (17)–(23) are equivalent constraints to Equation (16), which could be added in linear forms.

$$t \le p(u) - 1 + \left(1 - z_2(u,t)\right) \cdot M \ \ \forall u \in U,\ \ \forall t \in T, \tag{17}$$

$$t \ge p(u) - z_2(u,t) \cdot M \ \ \forall u \in U,\ \ \forall t \in T, \tag{18}$$

$$t \le g(u) + \left(1 - z_1(u,t)\right) \cdot M \ \ \forall u \in U,\ \ \forall t \in T, \tag{19}$$

$$t \ge g(u) + 1 - \left(z_1(u,t) + z_2(u,t)\right) \cdot M \ \ \forall u \in U,\ \ \forall t \in T, \tag{20}$$

$$1 \le t \le T, \tag{21}$$

$$z_1(u,t) + z_2(u,t) \le 1 \ \ \forall u \in U,\ \ \forall t \in T, \tag{22}$$

$$z_1(u,t) \in \{0,1\},\ z_2(u,t) \in \{0,1\} \ \ \forall u \in U,\ \ \forall t \in T. \tag{23}$$

3. Solution Method

A mixed integer optimization problem is formulated to solve the matching of request and vehicle dispatching. Current practice to obtain optimal solutions of such problems is through commercial software such as Gurobi and CPLEX. However, these solvers are known to have limitation in computation time once travel demand increases significantly.

To solve large-scale instances within an acceptable time, we intend to discover potential shared trips to reduce the computational complexity by exploring passengers' characteristics. Sharing trips denote trips that have overlapping schedules in both time and space dimensions. This is similar to the cluster analysis (e.g., [19]) to find relative elements in a cluster and divide a set into several independent clusters. Thus minimizing

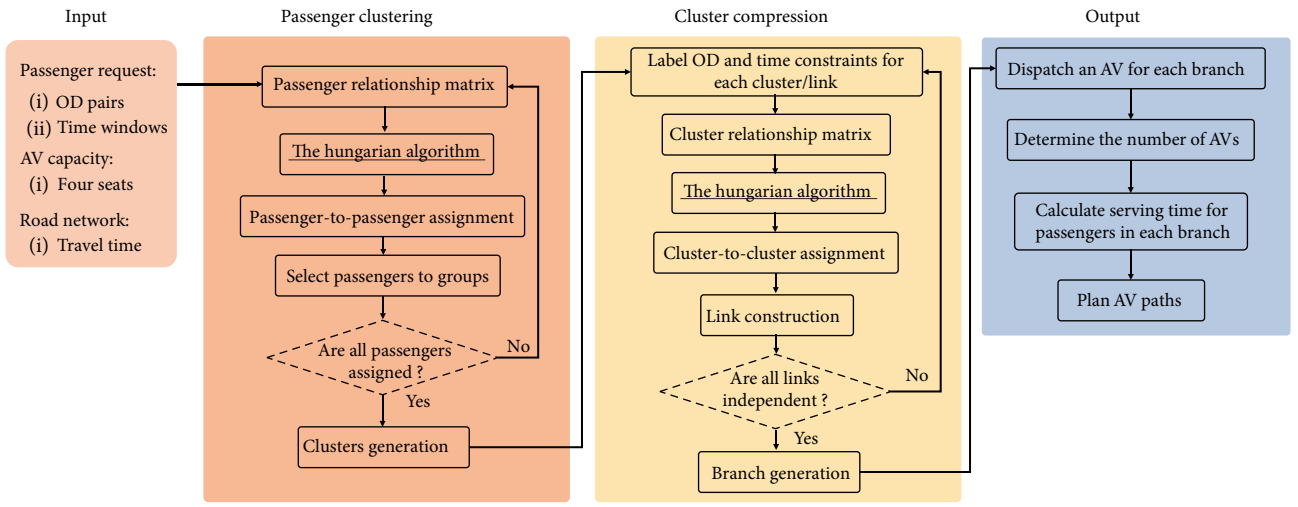


FIGURE 2: The flow chart of the cluster-based algorithm.

$$\Pi = \begin{bmatrix} \pi_{11} & \pi_{12} & \cdots & \pi_{1n} \\ \vdots & \pi_{22} & \ddots & \vdots \\ \pi_{u1} & & \pi_{uu} & \pi_{un} \\ \vdots & & \ddots & \vdots \\ \pi_{n1} & \cdots & \pi_{nu} & \cdots \ \pi_{nn} \end{bmatrix} \quad \text{where } \pi_{u,k} = R(u) + R(k) + (k) - R(s)$$

FIGURE 3: The relationship matrix of passengers.

the number of vehicle could be regarded as minimizing independent clusters. The minimization implies that each cluster should include as many passengers as possible.

In this way, the vehicle dispatching problem is converted to segmenting passenger set. If each cluster is taken as a passenger with specific OD pair and time restriction, traffic demand will be reduced to the number of clusters in this cluster-based method. Under this simplified network, passengers no longer have intersection in both time and space dimensions, i.e., multiple passengers sharing one vehicle at the same time is impossible. It can be concluded that the problem is approximately translated to a multiple travelling salesman problem (mTSP) with time windows. To solve this model, we also adopt the cluster-based method to classify passengers and derive the number of required vehicles. The cluster-based method has shown to have significant impact in simplifying the optimization model of this paper. The whole process of the proposed algorithm could be summarized into two parts, i.e., *passenger clustering* and *cluster compression*, as shown in Figure 2. The detailed description of this algorithm is stated as follows:

3.1. Part I: Passenger Clustering. To divide passengers into groups, the first step is to describe the sharing ability of passengers for selecting sharing pairs. It should be noted that a sharing pair means two passengers carried by one AV at the same time. Based on passengers' OD pairs and service time restrictions, we can roughly evaluate the sharing ability of any two passengers and generate a relationship matrix Π, as shown in Figure 3, where the element $\pi_{u,k}$ means the weight for a passenger pair. The detour distance, overlapping trip and waiting time are all feasible parameters to reflect their weights. To give an intuitive evaluation on the service system, we take the saving travel distance as the weighting value. In this way, the weight value is formulated as $\pi_{u,k} = R(u) + R(k) - R(s)$, where $R(s)$ denotes the travel distance for a passenger pair (u, k). It should be pointed out that a passenger pair will not be a valid pair if two passengers are not matched with time and space constraints. Under this situation, we set $R(s)$ as a very large number for invalid pairs. In addition, diagonal elements are also invalid pairs, which are set to 0 in this relationship matrix.

With this matrix, the task of passenger clustering could be considered as finding sharing pairs with the maximum total saved travel distance. This process is similar with job assignment problems, which is stated as follows:

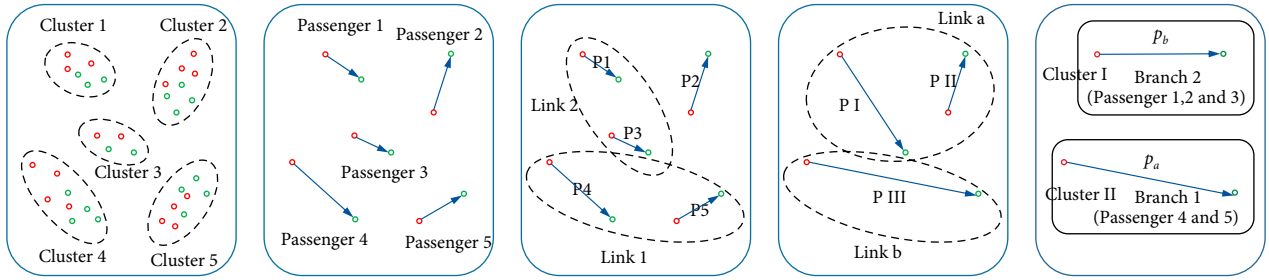
$$\max \sum_{u,k \in U} \pi_{u,k} \cdot \overline{\omega}_{u,k}, \tag{24a}$$

s.t.

$$\pi_{u,k} = R(u) + R(k) - R(s) \ \forall u, \quad k \in U, \tag{24b}$$

$$\sum_{k \in U} \overline{\omega}_{u,k} = 1 \ \forall u \in U, \tag{24c}$$

$$\sum_{u \in U} \overline{\omega}_{u,k} = 1 \ \forall k \in U, \tag{24d}$$



(a) Cluster segmentation (b) Network simplication (c) Link construction (d) Link extension (e) Branch generation

FIGURE 4: A paradigm of cluster compression.

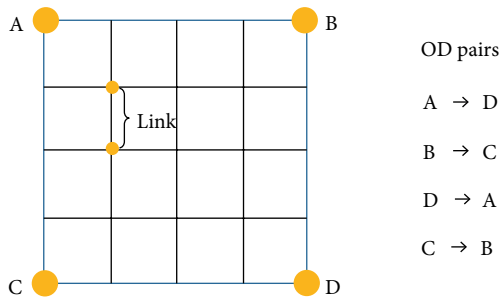


FIGURE 5: Sketch of a 5 × 5 size network.

where $\overline{\omega}_{u,k}$ is a binary variable, 1 if passenger u and k are combined and 0 otherwise. Equation (24a) denotes the objective of maximizing the total saved travel cost. Equation (24b) is the weight value expression for passengers. Equations (24c) and (24d) represent each passenger should be matched with a passenger to share a ride.

Results of this model might contain invalid pairs, which should be refined to obtain valid sharing pairs. In the relationship matrix, diagonal pairs might be invalid but are contained in the results. Hence, the pair (u, u) will be deleted if $\overline{\omega}_{u,u} = 1$. Repeated pairs are another issue for this model. For example, if $\overline{\omega}_{u,k}$ and $\overline{\omega}_{k,u}$ are simultaneously equal to 1, they both denotes passenger u and $k$will become a group. Thus, pair (u, k) or (k, u) needs to be deleted for simplifying sharing pairs.

The aforementioned model and refining process are the first step to determine passenger pairs, which could not be directly used to find minimum passenger groups. This is because more pairs might be found between passenger pairs and single passengers. To minimize the number of passenger groups, an iteration process is conducted to explore new passenger pairs. The whole procedure of passenger clustering could be concluded as follows:

Step 1: Preliminary. Collect passenger OD pairs and expected service times. Label them and mark them as original passengers. Then calculate the relationship matrix based on Equation (24b).

Step 2: Sharing pair determination. Solve Equations (24a)–(24d) with the Hungarian algorithm and obtain sharing pairs by deleting invalid pairs and repeated pairs. Then take each sharing pair as a group. If a passenger is simultaneously appeared in two groups, integrate these two groups as a group, until groups are independent. Unmatched passengers are respectively regarded as a group. If the set of sharing pairs is empty, stop the iteration and output passenger groups; otherwise go to step 3.

Step 3: Passenger number redefinition. Take each group as a new passenger, and relabel them.

Step 4: The relationship matrix recalculation. Since a new passenger represents a passenger group, the total travel distance $R(s)$ for a passenger pair should be calculated with a heuristic method or dynamic programming. For simplify, we will adopt the insertion algorithm to find a feasible visiting sequence to evaluate the total travel distance. Then generate the relationship matrix and go back to step 2.

3.2. Part II: Cluster Compression. This part aims to lower the upper bound and move it towards the optimal solution. If each cluster is served by one vehicle, the number of clusters gives an upper bound of the vehicle dispatching problem. To reduce the number of used vehicles, the method of recombining clusters is introduced, as shown in Figure 4. It should be noted that a vehicle could not serve multiple clusters at the same time for clusters output from Part I. In this way, if each cluster is considered as a passenger (Figure 4(b)), the problem is similar to mTSP with time windows. For mTSP, the objective is to minimize the total travel distance. Thus the goal of this part is to dispatch the minimum number of vehicles to visit these passengers with least travel distance. The basic idea is to prejudge any two passengers that could be served by a vehicle. For passenger u and u', assuming that they are served by the same vehicle, possible serving sequences are $o(u) \to d(u) \to o(u') \to d(u')$ and $o(u') \to d(u') \to o(u) \to d(u)$. If either of them satisfies time restrictions of picking up and dropping off, they will be marked as a link, as shown in Figure 4(c). It should be noted that the weighting value of a link is the travel distance of drop-off point $d(u)$ to pick-up point $o(u')$ or drop-off point $d(u')$ to pick-up point $o(u)$. To choose appropriate links, we also use the Hungarian algorithm to make decisions. A link will be considered as a new passenger in this network, as shown in Figure 4(d). Then the same process is applied again to link passengers till no link could be established. At the end, independent passengers (e.g., p_a and p_b in Figure 4(e)) are defined as branches, where a branch implies a vehicle is required and passengers could be derived from associating links. The whole iterative process is illustrated as below:

Step 1: Preliminary. Based on information submitted by passengers, recalculate OD and time requirement for clusters and label

Table 2: Model performance comparison (uniform distribution).

Requests	Our model			Cordeau's DARP model		
	Number of vehicles	Time (s)	Gap	Number of vehicles	Time (s)	Gap
24	6	1.01	0%	7	1.01	14.3%
32	8	3.36	0%	8	1.71	0%
40	10	3.03	0%	11	2.13	9.09%
48	12	5.63	0%	12	5.05	0%

Table 3: Model performance comparison (random generation).

Requests	Our model			Cordeau's DARP model		
	Number of vehicles	Time (s)	Gap	Number of vehicles	Time (s)	Gap
24	9	0.46	0%	10	1001	23.3%
28	10	1.52	0%	12	709	22.9%
32	12	1.84	0%	13	5512	25.6%
36	12	2.07	0%	15	4458	33.2%
40	12	5.92	0%	16	5055	41%
44	15	8.22	0%	16	6032	43%
48	14	15.27	0%	16	6124	43.5%

them with new passenger indexes $\eta \in U^*$. Then generate a 0-1 matrix to express the connectivity of these recreate passengers and mark gap distances as weight values for links and a very large number for nonlinks.

Step 2: Link selection. Apply the Hungarian algorithm to find optimal links with the minimum accumulated gap distance. Then filter these selected links to produce independent links, where this process is similar with Equation (23).

Step 3: Stop criterion. If there is no feasible links available, stop calculation and output branches; otherwise go to step 4.

Step 4: Passenger set updating. Recalculate OD and time limitation for each independent links and label passengers indexes again. Then go back to step 2.

4. Numerical Experiments

In this section, a set of cases are generated to examine the validity of our model and performance of the proposed cluster-based algorithm. For providing abundant comparisons, computational results of a typical DARP model and insertion algorithm are calculated. From these results, the application scope of our model and the proposed algorithm are concluded. In addition, the effect of sensitive parameters on determining the minimum number of vehicles are analyzed for the SAV system.

4.1. Results of Small Scale Problem. For the small scale problem test, request size is set to be ranging from 24 to 48 defined on a simple network of 5×5 (Figure 5). Travel time of each link between adjacent nodes is assumed to be 60 seconds. Two simple scenarios are employed: fixed and random OD pairs.

In the first scenario, we intend to explain our model with predetermined sharing paths. The OD pairs are designed to be in diagonal directions (A → D, B → C, D → A, and C → B). Each OD pair is allocated with the same amount of requests. The time windows of the requests for the same OD pair are identical and they only allow two other sharing paths A → B → D → C and D → C → A → B. In this uniformly distributed scenario, minimum vehicle number for the 24-request case could be estimated in the following way. First we consider ride sharing among the same OD pair, i.e., assigning one vehicle to each OD pair. In this situation, four passengers could be served and two passengers are left for each OD pair. Since path A → B → D → C and D → C → A → B are sharing rides, dispatching one vehicle for each path will completely serve the remaining passengers. In this way, 6 vehicles are needed to satisfy 24 passengers. Similarly, the minimum number for the other instances are 8, 10, and 12 vehicles. Our model and a typical DARP model [9] are solved for this case using Gurobi solver. Their results are listed in Table 2. For the four cases, our model could obtain the correct minimum vehicle number as aforementioned. However, the DARP model could not find the correct minimum number for the 24 and 40 cases within a short time.

In the randomly generated scenario, any two nodes on the network could form an OD pair with stochastically given time windows. For randomly generated case, one passenger might share rides with more passengers but only within time windows restriction. As is presented in Table 3, our model could still give optimal values. Computation times of our model are not drastically exhausted for exploring the best passenger combination. It implies that our model is more adaptive to solving the vehicle dispatching problem.

For these instances, the proposed cluster-based algorithm and a heuristic algorithm (insertion algorithm [20]) are also used to find optimal dispatching plans. Results of these algorithms are listed in Tables 4 and 5. In fixed OD pair case, the minimum vehicle number of the two algorithms are equal to the optimal values and computational times are shorter than those of Gurobi solvers. For the random case, most results of

Table 4: Algorithm performance comparison (uniform distribution).

Requests	Cluster-based algorithm			Insertion algorithm			Gurobi solver	
	Number of vehicles	Time (s)	Gap	Number of vehicles	Time (s)	Gap	Number of vehicles	Time (s)
24	6	0.23	0%	6	0.31	0%	6	1.01
32	8	0.42	0%	8	0.29	0%	8	3.36
40	10	0.77	0%	10	0.38	0%	10	3.03
48	12	1.28	0%	12	0.35	0%	12	5.63

Table 5: Algorithm performance comparison (random generation).

Requests	Cluster-based algorithm			Insertion algorithm			Gurobi solver	
	Number of vehicles	Time (s)	Gap	Number of vehicles	Time (s)	Gap	Number of vehicles	Time (s)
24	9	0.3	0%	11	0.38	18.2%	9	0.46
28	11	0.29	9%	12	0.35	16.7%	10	1.52
32	12	0.35	0%	13	0.33	7.7%	12	1.84
36	12	0.49	0%	14	0.42	14.3%	12	2.07
40	13	0.68	8%	15	0.53	20%	12	5.92
44	15	0.79	0%	17	0.62	11.8%	15	8.22
48	14	0.83	0%	16	0.67	2.5%	14	15.27

Table 6: Computation time comparison.

Computation time (s)	Requests					
	50	60	70	80	90	100
Gurobi solver	35.43	89.16	191	307.6	927.4	2835.6
Cluster-based algorithm	0.95	0.88	1.16	1.34	3.01	5.21

cluster-based algorithm find the best values while the heuristic algorithm could not give any optimal values. For request 28 and 40, although dispatching plan derived from cluster-based algorithm needs one more vehicle, they are less than those by the insertion algorithm. This shows that the cluster-based algorithm has a better chance of finding the optimal or near-optimal values. Nevertheless, compared with the insertion algorithm, the cluster-based algorithm might need a little longer time to obtain superior results.

To further analyze our model's adaptability, we conduct a series of tests to observe computation time variation with increased requests on the same road network, as shown in Table 6. In addition, the minimum number of vehicles of cluster-based algorithm are 8, 9, 8, 13, 12, and 13; and the optimal values of our model are 8, 8, 8, 11, 12, and 11. Although cluster-based algorithm could not search the optimal values for all cases, it has an overwhelming advantage in terms of computation time. It can be seen that the computation time of Gurobi is more than 1000 seconds when request reaches 100, which might need several hours to find the optimal value for larger requests. Therefore cluster-based algorithm will become a better choice when near-optimal solutions are demanded in a short time with hundreds of requests.

4.2. Comparison Test for Large-Scale Case. In addition to the 5×5 road network, a larger 20×20 grid representing city downtown is adopted with randomly generated requests. The length of each link is set to 2640 ft and free flow speed is 30 mph [7]. Assuming a maximum vehicle speed, travel time on each link is approximately 60 seconds. The time interval between earliest and latest time of pick-up or drop-off nodes is still set to be 120 seconds.

In this section, random requests ranging from 500 to 4000 are tested. The average distance of passengers' trips are 8.59 miles, 8.69 miles, 8.7 miles, 8.65 miles, 8.65 miles, 8.67 miles, 8.63 miles, and 8.66 miles, respectively. Total vehicle number and total travel distance solved by the cluster-based algorithm are listed and compared with the insertion algorithm (Table 7). It is obvious that the proposed algorithm could find a ride-sharing pattern with fewer number of vehicles and shorter travel distance. In terms of vehicle number, the proposed cluster-based algorithm saves 12–42 vehicles as compared to the insertion algorithm. As for travel distance, a reduction of 100–600 miles could be obtained.

The average occupation rate (average passenger number per vehicle) obtained with the cluster-based algorithm ranges from 3.6 to 5.32, which is higher than the insertion algorithm's occupation rate from 3.31 to 5.12 (Figure 6). Detailed passenger number distributions are presented in Figure 7. As could be noticed, the passenger number distributions have similar form of a normal distribution with peak frequency in the middle. The cluster-based algorithm's distributions always have a larger median value. For example, the mean value of cluster-based algorithm is 4 for the 500 request case, which is one passenger more than the insertion algorithm. This means that the cluster-based algorithm is able to apply more sharing rides

Table 7: Comparison results for large-scale instances.

Scenario	The number of vehicles			Total travel distance (mile)		
	Cluster-based algorithm	Insertion algorithm	Percentage (%)	Cluster-based algorithm	Insertion algorithm	Percentage (%)
500	139	151	7.9	1564.5	1635.75	4.3
1000	254	277	8.3	2880.35	3065.35	6.1
1500	340	360	4.1	3966.15	4110.15	3.5
2000	421	445	5.4	5043.75	5260	4.1
2500	511	544	6.1	6121.87	6402.5	4.4
3000	600	636	5.6	6984	7254.25	3.7
3500	663	705	5.9	8077	8304	2.7
4000	752	781	3.7	9070.75	9224.24	1.7

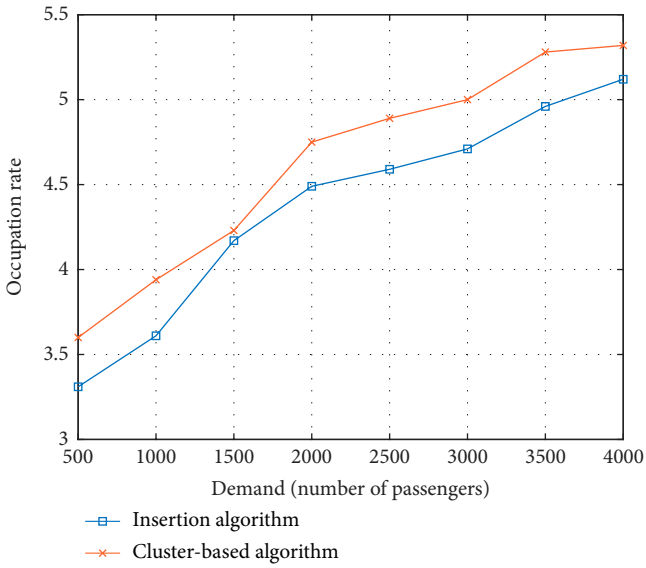


Figure 6: The occupation rate comparison.

than the insertion method since its number of highly loaded vehicle is larger. For example, for 4000 requests case, 52 vehicles are assigned with 8 passengers by the cluster-based algorithm while only 7 vehicles could be assigned with the same number of passengers for the insertion algorithm.

4.3. Sensitivity Analysis. In this section, we conduct a sensitivity analysis to examine how the performance of the proposed algorithm is affected by key input parameters. Figure 8 presents results of demanded vehicle number under various vehicle capacities. It should be noted that capacity equaling 1 means only one passenger is served at a time by a vehicle with four seats where ride-sharing is not included. For capacity of 7 case, seven passengers are allowed to be carried at the same time. For simplicity, we mark vehicle capacity of 1, 4, and 7 as condition I, II, III respectively. For condition I, the number of required vehicles are much higher than the other two conditions, while results of condition II and III are of little difference. If condition II is considered as the benchmark, it seems that vehicles with higher capacity makes little contribution to reducing total vehicle number. This implies that the SAV system might not gain more benefits by introducing larger vehicles with more seats under this condition. This is mainly because the probability of numerous passengers with very similar OD pair and the time windows are relatively low. The average number of multiple passengers served by a vehicle at the same time will not be a large value, especially when demand is lower than 3000. Thus a vehicle with four seats is enough under this condition.

The influence of the time windows (the difference between earliest and latest expected time of picked up or dropped off) on system's total vehicle number is shown in Figure 8. With increasing time interval, total vehicle number gradually reduces from 139 to 93 as time windows increase from 2 minutes to 16 minutes (Figure 9(a)), which results in shortened total travel distance (Figure 9(b)). It indicates that the system succeeds in finding sharing rides for more passengers with extended time restriction. Since the time interval is a key parameter that reflects the willingness of waiting time, passengers might be picked later compared with the earliest expected times if it is set to be longer. If this parameter is very large, it is intuitive that they will have more opportunities to share a ride with others. Under this condition, the number of vehicles will become smaller. For more trips which are integrated as a trip, the total travel distance will also be reduced.

5. Conclusions

In this paper, we formulate the vehicle dispatching problem of SAV transportation system into a 0-1 integer programming model. Unlike existing vehicle routing optimization models, our model focuses on exploring the similarity of passengers' demand in time and space dimensions to classify passengers into groups, in which the number of required vehicles is derived indirectly. To solve this model, the cluster-based algorithm is proposed for classifying passengers. The whole process consists of two parts: (1) the Hungarian algorithm is introduced to select appropriate passengers sharing trips and determine an upper bound for required vehicles; (2) a reunion process by linking sharing trips is conducted to lower the upper bound. Since the Hungarian algorithm only needs a polynomial time, the computational complexity of the

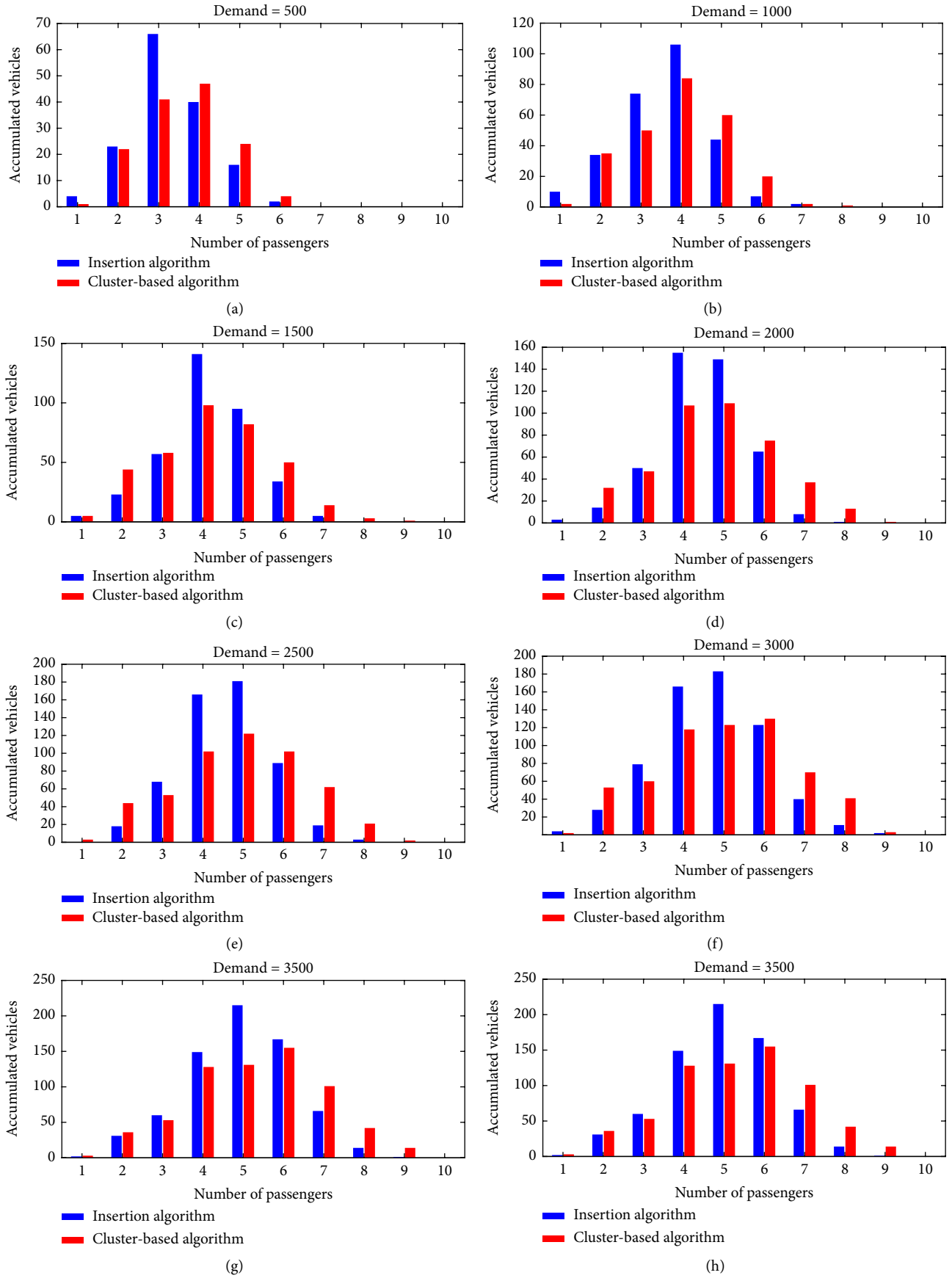


Figure 7: The distribution of served passengers of each vehicle.

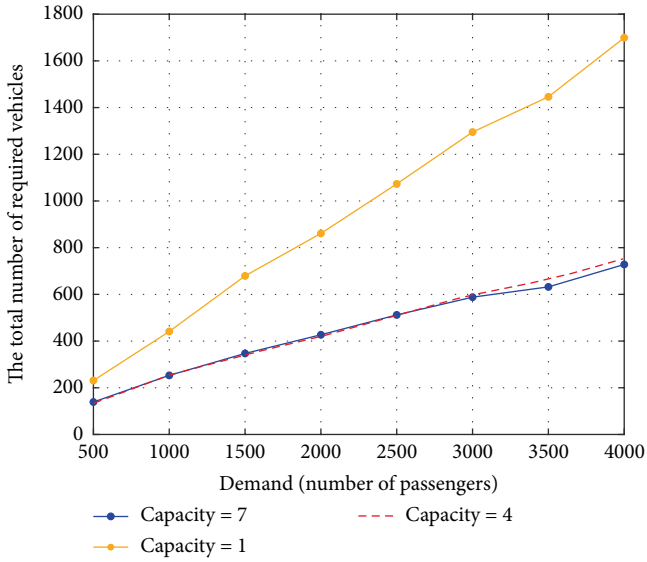


Figure 8: Sensitivity analysis of vehicle capacity.

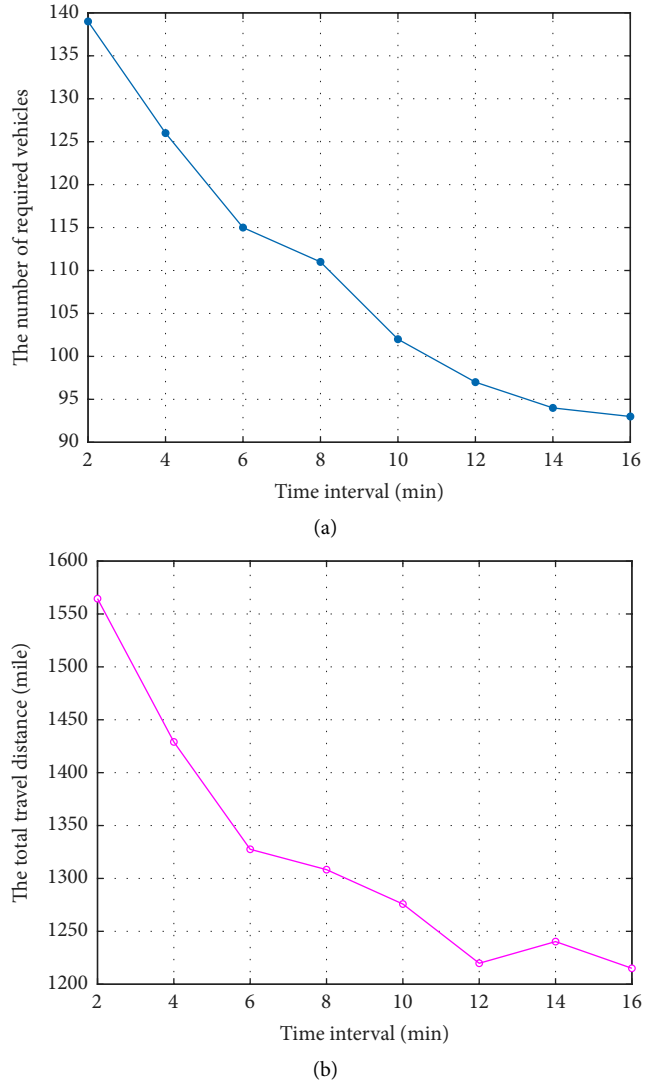


Figure 9: Sensitivity analysis of the time windows. (a) The number of vehicles and (b) the total travel distance.

proposed algorithm could be greatly reduced, which makes it applicable for solving large-scale cases.

The validity and efficiency of our vehicle dispatching model and the proposed cluster-based algorithm are presented by conducting a series of tests. First the model and algorithm are applied for small-size passenger requests. Results show that the proposed algorithm could always find optimal or near-optimal solutions when comparing the optimal values with obtained from the optimization solver. We also list results of a typical DARP model [9] and insertion algorithm for further analysis. By comparing computation time and solution gaps, it indicates that the proposed algorithm has an advantage in gathering passengers sharing a vehicle and making the objective function towards the best value. For large-size cases, the new algorithm still expresses a better performance than the insertion algorithm in minimizing the number of vehicles and total travel distance. At last, the effect of key input parameters on the number of vehicles are discussed. It is concluded that enlarging vehicle capacity will not reduce used vehicles in consequence when it exceeds four and extending waiting time will make a positive feedback on decreasing the number of vehicles.

Through the whole study, we mainly investigate how to minimize used vehicles for the SAV system with given demands, which is a static dispatching method. To enrich the application scope of the cluster-based algorithm, the dynamic or on-line planning will be regarded as an interesting research direction. Furthermore, we assume and all passengers have the willingness of accepting ride sharing, which only consider the dispatching problem from the system view. We will extend our model by introducing customized passengers demand for the further research. In addition, charging price is a key factor in passengers' decisions, which might bring a trade-off in passengers cost and the system revenue. In this way, price optimization will be also considered as our future research to enrich our model.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is partially supported by the National Science Foundation of China under Grant #61773040, and #U1811463, and the National Key Research and Development Program of China (2016YFB0100902).

References

[1] A. Talebpoura and H. S. Mahmassani, "Influence of connected and autonomous vehicles on traffic flow stability

and throughput," *Transportation Research Part C: Emerging Technologies*, vol. 71, pp. 143–163, 2016.

[2] T. D. Chen, K. M. Kockelman, and J. P. Hanna, "Operations of a shared, autonomous, electric vehicle fleet: implications of vehicle & charging infrastructure decisions," *Transportation Research Part C*, vol. 94, pp. 243–254, 2016.

[3] Z. Chen, F. He, and Y. Yin, "Optimal deployment of charging lanes for electric vehicles in transportation networks," *Transportation Research Part B: Methodological*, vol. 91, pp. 344–365, 2016.

[4] B. Boyaci, K. G. Zografos, and N. Geroliminis, "An optimization framework for the development of efficient one-way car-sharing systems," *European Journal of Operational Research*, vol. 240, no. 3, pp. 718–733, 2015.

[5] K. Huang, G. H. de A. Correia, and K. An, "Solving the station-based one-way carsharing network planning problem with relocations and non-linear demand," *Transportation Research Part C: Emerging Technologies*, vol. 90, pp. 1–17, 2018.

[6] J. Ma, X. Li, F. Zhou, and W. Hao, "Designing optimal autonomous vehicle sharing and reservation systems: a linear programming approach," *Transportation Research Part C: Emerging Technologies*, vol. 84, pp. 124–141, 2017.

[7] M. W. Levin, K. M. Kockelman, S. D. Boyles, and T. Li, "A general framework for modeling shared autonomous vehicles with dynamic network-loading and dynamic ride-sharing application," *Computers, Environment and Urban Systems*, vol. 64, pp. 373–383, 2017.

[8] H. Miao, H. Jia, J. Li, and T. Z. Qiu, "Autonomous connected electric vehicle (ACEV)-based car-sharing system modeling and optimal planning: a unified two-stage multi-objective optimization methodology," *Energy*, vol. 169, pp. 797–818, 2019.

[9] J.-F. Cordeau, "A branch-and-cut algorithm for the dial-a-ride problem," *Operation Research*, vol. 54, no. 3, pp. 573–586, 2006.

[10] M. Liu, Z. Luo, and A. Lim, "A branch-and-cut algorithm for a realistic dial-a-ride problem," *Transportation Research Part B: Methodological*, vol. 81, pp. 267–288, 2015.

[11] C. Bongiovanni, M. Kaspi, and N. Geroliminis, "The electric autonomous dial-a-ride problem," *Transportation Research Part B: Methodological*, vol. 122, pp. 436–456, 2019.

[12] H. Hosni, J. Naoum-Sawaya, and H. Artail, "The shared-taxi problem: formulation and solution methods," *Transportation Research Part B: Methodological*, vol. 70, pp. 303–318, 2014.

[13] M. Mahmoudi and X. Zhou, "Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: a dynamic programming approach based on state–space–time network representations," *Transportation Research Part B: Methodological*, vol. 89, pp. 19–42, 2016.

[14] J.-F. Cordeau and G. Laporte, "A Tabu search heuristic for the static multi-vehicle dial-a-ride problem," *Transportation Research Part B: Methodological*, vol. 37, no. 6, pp. 579–594, 2003.

[15] L. Häme and H. Hakula, "A maximum cluster algorithm for checking the feasibility of dial-a-ride instances," *Transportation Science*, vol. 49, no. 2, pp. 295–310, 2015.

[16] M. Diana and M. M. Dessouky, "A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows," *Transportation Research Part B: Methodological*, vol. 38, no. 6, pp. 539–557, 2004.

[17] M. M. Vazifeh, P. Santi, G. Resta, S. H. Strogatz, and C. Ratti, "Addressing the minimum fleet problem in on-demand urban mobility," *Nature*, vol. 557, no. 7706, pp. 534–538, 2018.

[18] T. R. Sexton and L. D. Bodin, "Optimizing single vehicle many-to-many operations with desired delivery times: I. Scheduling," *Transportation Science*, vol. 19, no. 4, pp. 378–410, 1985.

[19] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297, University of California Press, Berkeley, CA, 1967.

[20] J. Jaw, A. Odoni, H. Psaraftis, and N. Wilson, "A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows," *Transportation Research B*, vol. 20, pp. 243–257, 1986.