Autonomous Vehicle Public Transportation System: Scheduling and Admission Control

Albert Y.S. Lam, Yiu-Wing Leung, and Xiaowen Chu

***Abstract*—Technology of autonomous vehicles (AVs) is getting mature and many AVs will appear on the roads in the near future. AVs become connected with the support of various vehicular communication technologies and they possess high degree of control to respond to instantaneous situations cooperatively with high efficiency and flexibility. In this paper, we propose a new public transportation system based on AVs. It manages a fleet of AVs to accommodate transportation requests, offering point-to-point services with ride sharing. We focus on the two major problems of the system: scheduling and admission control. The former is to configure the most economical schedules and routes for the AVs to satisfy the admissible requests while the latter is to determine the set of admissible requests among all requests to produce maximum profit. The scheduling problem is formulated as a mixed-integer linear program and the admission control problem is cast as a bilevel optimization, which embeds the scheduling problem as the major constraint. By utilizing the analytical properties of the problem, we develop an effective genetic-algorithm-based method to tackle the admission control problem. We validate the performance of the algorithm with real-world transportation service data.**

***Keywords*—Autonomous vehicle, admission control, bilevel optimization, car sharing, smart city.**

I. Introduction

HUMAN mobility is largely supported by public transport. Many people rely on public transport to move from one place to another when the destinations of their journeys are not within walkable distances. To transform a city with limited room for large-scale infrastructure into a smart city, its public transportation system may need to be further upgraded mainly from the existing road networks. Representatives of road-based public transport are buses and taxis, each type of which has its pros and cons. In general, buses follow fixed routes offering shared ride so that more passengers can be served on each single journey. On the other hand, taxis offer private services and run on flexible dedicated routes based on the passengers' requests. Nevertheless, no single one type can support high throughput and flexibility at the same time. The efficiency and capacity of the whole public transportation system may be enhanced if there exists a new public transport which can accommodate many people in a short period of time and concur high mobility. It may maintain flexibility by offering point-to-point services while enhancing efficiency by supporting shared ride. Such kind of public transport requires several characteristics which may not be possessed by a typical public transport. To develop such a public transport, the vehicles need to cooperate to take up customers' requests instead of cruising around the city for random offers. To enhance the efficiency and cooperativeness, a control center can be employed to coordinate all the vehicles, manage all the service requests, and assign the vehicles to serve the requests. Moreover, the vehicles should follow the routes and carry out the travel plans instructed so as to achieve system-wise objectives. Recently, autonomous vehicles (AVs) have been undergone active research and we can expect many AVs running on the roads in the near future. The AV is a good candidate possessing most of the requirements mentioned above. Hence AVs can be adopted to construct a new smart public transportation system with high efficiency and flexibility.

In this paper, we introduce an intelligent AV-based public transportation system. It manages a fleet of AVs to accommodate transportation requests, offering point-to-point services with ride sharing. We focus on two important problems in the system: *scheduling* and *admission control*. The former is about how to assign the designated vehicles to the admissible transportation requests, and when and where the vehicles should reach to provide services with the lowest cost. The latter is to determine the set of admissible requests among all requests to achieve maximum revenue. As a whole, the contributions of this paper include:

- proposing the AV public transportation system;
- improving the model for scheduling proposed in [1], such that the formulation developed in this paper can now support both directed and undirected graphs;
- developing distributed scheduling;
- formulating the admission control problem;
- introducing the concept of admissibility and the related analytical results;
- designing an effective method to solve the admission control problem; and
- validating the performance of the solution method with real-world transportation service data.

The rest of this paper is organized as follows. Related work is given in Section II and we present various system components and their operations in Section III. The scheduling problem is discussed in Section IV. In Section V, we formulate the admission control problem and provide the related analytical results. We propose a genetic-algorithm-based solution method for admission control and develop distributed scheduling in

A preliminary version of this paper was presented in [1].

A.Y.S. Lam is with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam, Hong Kong (e-mail: ayslam@eee.hku.hk).

Y.-W Leung and X. Chu are with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong (e-mail: {ywleung, chxw}@comp.hkbu.edu.hk).

Section VI. Section VII evaluates the system performance with real-world transportation service data. Finally we conclude this paper in Section VIII.

II. Related Work

The concept of AVs was raised in the 1920's and the research thereof has started for more than thirty years. An AV is equipped with many sensors, which provide the vehicle with full sensing ability so as to adapt to the neighborhood environment and realize fully automated control. In 2007, the DARPA Urban Challenge boosted the awareness of AVs capable of being driven in traffic and performing complex maneuvers [2]. In 2010, VisLab carried out the experiment that several driverless vehicles successfully traveled 13,000 km from Italy to China [3]. Google demonstrated an AV prototype in 2011 [4]. By the end of 2013, several states in the United States, including Nevada, Florida, California, and Michigan, had passed the law to allow AVs running on public roads [5]. The first self-driving shuttle on sale was from NAVIA [6]. Other automotive manufacturers, like Mercedes-Benz [7], BMW, and Audi [8], have invested in self-driving technologies and include AVs in their production plans.

Most research work on AVs mainly focused on the control and communication aspects. Mladenovic and Abbas [9] proposed a self-organizing and cooperative control framework for distributed vehicle intelligence. Hu *et al.* [10] studied lane assignment strategies for connected AVs and proposed a lane changing maneuver to balance the tradeoff between efficiency and safety. Petrov and Nashashibi [11] developed a feedback controller for autonomous overtaking without utilizing roadway marking and inter-vehicle communication. Li *et al.* [12] presented a multi-level fusion-based road detection system for driverless vehicle navigation to ensure safety in various road conditions. All these show that AV is a promising technology with the support from governments, high-tech companies, and car manufacturers.

Vehicles can communicate with each other and fixed infrastructure via various vehicular wireless communication techniques [13]. Nowadays vehicular communications are mostly deployed over satellite, cellular networks, and vehicular ad-hoc networks (VANETs) [13]. VANET is a mobile ad-hoc network where vehicles act as the mobile nodes [14] and it can improve the communication capacity and organization of AVs constituting an intelligent transportation system. Furda *et al.* [15] introduced a wireless communication framework for driverless vehicles. It facilitated vehicle-to-vehicle and vehicle-to-infrastructure communications and improved the safety and efficiency of vehicles. Alsabaan *et al.* [16] made use of traffic light signals and vehicle-to-vehicle (V2V) communications to help vehicles adapt their speeds and avoid unnecessary stop, acceleration, and excessive speed. Gomes *et al.* [17] designed a driver-assistance system which allowed a vehicle to collect real-time camera images from other vehicles in the neighborhood over V2V communications. In this way, AVs become connected and can communicate with the control center.

Shareability of taxi services has been studied recently. Santi *et al.* [18] investigated the tradeoff between passenger inconvenience and collective benefits of sharing and concluded that a small increase in discomfort could induce the significant benefits of less congestion, less running costs, less split fares, less polluted, and cleaner environment. Ma *et al.* proposed a taxi ridesharing system called T-Share in [19], where the dynamic taxi ridesharing problem was studied. For a dataset of taxi services in Beijing, it showed that 25% additional taxi users could be served with saving of 13% of total travel distance. These studies confirmed that ridesharing is beneficial but they mostly focused on taxi services. In this paper, we focus on AVs, which have a key intrinsic property hardly found in the standard taxis: the direct control of vehicles does not involve any human factors. In other words, AVs can completely follow the instructions from the control center in the sense that they neither undertake any unassigned requests nor reject any assigned requests. We can see that AVs can fully cooperate to achieve the system objective but it may not be the case for human-driving taxis.

The AV public transportation system is uniquely designed and it can help improve the capacity and flexibility of the future transportation system. It is extended to a multi-tenant system in [20], in which new service types are introduced and the pricing problem is addressed. To further demonstrate its feasibility, we show how the existing work discussed above

TABLE I. Contributions to the system.

Technology/ feature	Example	Contributions	Ref.
Hardware	VisLab	Demonstrate the feasibility of AVs	[3]
	Google	Show the confidence of the industry in AVs	[4]
	Mercedes-Benz, BMW, Audi, NAVIA	Guarantee supply of AVs for the system	[7], [8], [6]
Software	Mladenovic & Abbas	Enhance self-organizing and cooperative control of AVs	[9]
	Hu *et al.*	Balance the efficiency and safety of AVs	[10]
	Petrov & Nashashibi	Enhance self-control of AVs	[11]
	Li *et al.*	Improve safety of AVs	[12]
Law	Nevada, Florida, California, and Michigan	Demonstrate the support of governments	[5]
Communications	Cottingham	Introduce the vehicular wireless communications available to be used in the system	[13]
	Dahiya & Chauhan	Improve the communication capacity and organization of AVs	[14]
	Furda *et al.*	Enhance the communications between AVs and the control center	[15]
	Alsabaan *et al.*	Improve the comfort of AVs	[16]
	Gomes *et al.*	Collect data for the system to estimate traffic conditions	[17]
Ridesharing	Santi *et al.*	Confirm the ridesharing functionality of the system	[18]
	Ma *et al.*		[19]
AV public transportation system	Lam *et al.*	Provide a proof of concept	[1]
	Lam *et al.*	Investigate the scheduling and admission control problems	This work

may contribute to the system in Table I.

The scheduling problem has been introduced in [1] and it can be considered as a variant of the Dial-A-Ride Problem (DARP) [21]. However, in our AV scheduling problem, we allow modifying the previously assigned but not yet served requests at desirable times to achieve system-wise performance goal. When the system evolves, the AVs appear at different locations at different time instants. It may happen that a particular request can be better served by a different AV at different times. Consider an example with two AVs, I and II. At a paricular time, AV-I is in the neighborhood of a location while AV-II is not. A request originated from this location may be better served by AV-I. After some time, AV-I may have gone away but AV-II may have come into the neighborhood. Then the request may be better served by AV-II instead. As the AVs are connected through appropriate vehicular communication technologies, the schedules of AVs can be revised from time to time. We consider this in our formulation making our scheduling problem different from DARP. As the system involves a number of AVs, determining their schedules in a distributed manner can undoubtedly speed up the process. Distributed scheduling has been advanced in many engineering disciplines, e.g., communication networks [22], [23]. As a new system, we will dedicatedly design a distributed methodology for the scheduling thereof.

Admission control generally refers to a validation process in communication systems for quality-of-service assurance. It determines which new connection or service request can be granted with resources for subsequent operations. For example, [24] designed an admission control mechanism to add or drop session requests in 4G wireless networks and [25] discussed various admission control algorithms for multi-service IP networks. We adopt this idea in the transportation system and design an admission control mechanism to differentiate the transportation service requests for maximizing the total profit. There are many methods to facilitate admission control. Genetic Algorithm (GA) is one of them and it has been successfully utilized to design admission control mechanisms, e.g., [26] and [27]. Based on the special formulation of the admission control problem (to be discussed in Section V), we will also adopt GA to solve the problem.

III. System Model

In this section, we design the architecture for the system which can manage a fleet of AVs to serve customers for transportation services. In the following, we first introduce the system components and then describe the operations characterizing their interactions.

A. System Components

1) Network Structure: A graph is employed to model the region being served by the system. It characterizes the locations and the road connections necessarily to describe movements of the AVs, origins and destinations of the service requests, and other required facilities. It is a directed graph denoted by $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of locations and $\mathcal{E}$ refers to the road segments connecting the locations so that we can completely describe the routes of AVs with G. For $i, j \in \mathcal{V}$, each edge $(i, j) \in \mathcal{E}$ is associated with an operational cost c_{ij} and a travel time t_{ij}, which is an estimation of time for an AV to traverse from i to j based on historical data. Depended on the system objective, c_{ij} typically represents the distance of the road segment (i, j) as the operational cost of AVs is usually measured by the fuel consumption which is in turn characterized by the travel distance. If the system aims to optimize the total service duration, we can set $c_{ij} = t_{ij}$ for all (i, j)'s. We allow $c_{ij} \neq c_{ji}$ and $t_{ij} \neq t_{ji}$ to account for the asymmetry of road segments. Moreover, refuel stations are located in some locations specified by $\tilde{\mathcal{V}} \subset \mathcal{V}$ and each AV ends its journey at any one of these refuel stations (reasons explained in Section III-B1). Based on the nature of the AVs, $\tilde{\mathcal{V}} \subset \mathcal{V}$ will be the locations of charging (gas) stations if the AVs are electric (conventional) vehicles. For the case of electric vehicles, $\tilde{\mathcal{V}} \subset \mathcal{V}$ can be determined based on the charging demand and the connectivity of the charging station network according to [28].

2) Transportation Requests: Customers request services in the form of transportation requests, which are collectively denoted by $\mathcal{R}$. Each $r \in \mathcal{R}$ is represented by the 5-tuple $\langle s_r, d_r, T_r, [e_r, l_r], q_r \rangle$. $s_r \in \mathcal{V}$ and $d_r \in \mathcal{V}$ represent the customer pickup and dropoff locations, respectively. T_r is the maximum ride time, an exceedance of which will lead to customer dissatisfaction. $[e_r, l_r]$ refers to the service starting time window, where e_r and l_r are the earliest and latest service starting times, respectively. q_r stands for the number of seats needed in the request r.

3) Vehicles: The system coordinates a fleet of AVs denoted by $\mathcal{K}$. Each $k \in \mathcal{K}$ is represented by the 5-tuple $\langle a_k, t_k^0, \tilde{T}_k, Q_k, \mathcal{R}_k \rangle$. $a_k \in \mathcal{V}$ is the first location where k will visit from the current position of k while t_k^0 is the time required to reach a_k from its current position. It is possible that, at the time of scheduling, the AV is in the middle of a road segment heading to a_k. a_k and t_k^0 can be easily estimated by submitting its current position to the system. $\tilde{T}_k$ denotes the maximum remaining operation time that k can continue to provide services without refueling.[1] Q_k is the passenger capacity that k can accommodate simultaneously. $\mathcal{R}_k = \tilde{\mathcal{R}}_k \cup \overline{\mathcal{R}}_k \in \mathcal{R}$ is the set of requests previously assigned to k. $\mathcal{R}_k$ can be further categorized into two types; $\tilde{\mathcal{R}}_k$ contains those currently being served by k while $\overline{\mathcal{R}}_k$ was assigned to k at a previous schedule but the services have not been implemented yet. For the former, some seats have already been taken by the customers from $\tilde{\mathcal{R}}_k$. On the contrary, seats have only been reserved but no actual seats have been taken from $\overline{\mathcal{R}}_k$. We will handle $\tilde{\mathcal{R}}_k$ and $\overline{\mathcal{R}}_k$ differently when performing scheduling in Section IV.

Without loss of generality, we assume that the number of seats required in any request is no larger than the capacity of any vehicle, i.e.,

$$q_r \leq Q_k, \forall r \in \mathcal{R}, k \in \mathcal{K}. \tag{1}$$

[1]The maximum remaining operation time of k can be converted from its corresponding remaining fuel level.

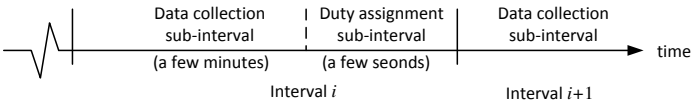


Fig. 1. Operating intervals in the system.

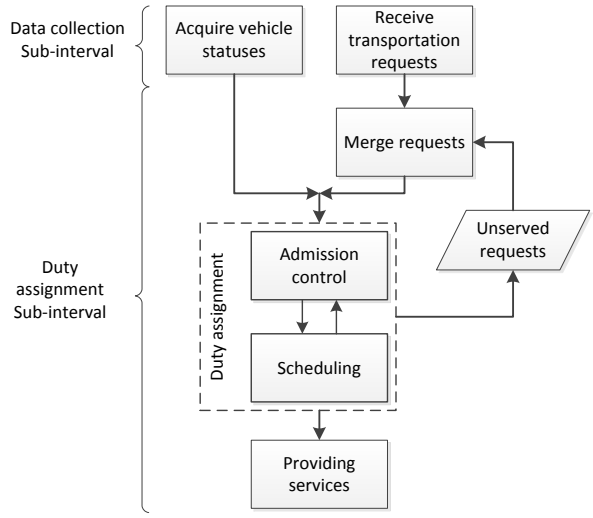


Fig. 2. Operation flow of the system.

We can always split those requests violating (1) into multiple requests so that this condition always holds.

B. Operations

The system is managed and operated by a control center whose main duties are to collect all the required information and assign the AVs to serve the transportation requests. The system operates in a fixed time interval basis and each time interval is divided into data collection and duty assignment sub-intervals (see Fig. 1). In each interval, the control center first collects transportation requests and vehicle statuses in the data collection sub-interval. Then the AVs are assigned to serve the transportation requests in the duty assignment sub-interval. On one hand, the duration of each interval should be long enough such that the communication delays will not result in any data missing from the customers and vehicles for scheduling. On the other hand, it should be short enough such that the collected data can reflect the current situation happening in that interval. In practice, the data collection sub-interval is longer than the duty assignment one. The former may last for a few minutes while the latter may takes a few seconds.

Fig. 2 illustrates the operation flow of the system with respect to an operating interval. As powered by various wireless vehicular communication technologies, all AVs are connected and can communicate with the control center instantaneously. In this way, the control center can collect the necessary vehicle statuses, e.g., current locations of AVs, confirmation of serving requests, traffic congestion information, etc., in the data collection sub-interval. Customers can also submit their requests to the control center by any appropriate means, e.g., phone calls, mobile apps, etc. After the data collection sub-interval, all the data required to perform duty assignment are ready at the control center.

In the duty assignment sub-interval, the control center processes the collected data and computes the duty assignment. There may exist some unattended requests incurred from some previous intervals because of their unsuitability in the previous system conditions. They are merged with the newly submitted requests and then all these requests are considered *en masse*. The duty assignment further consists of two processes: admission control and scheduling. Admission control checks all the outstanding requests and determines which requests are going to be admitted in the current interval. The unadmitted requests will be reserved for consideration in the next interval again. Any invalid or inappropriate requests are also permanently excluded in the admission control process. We compute the travel schedules of the AVs to serve the admitted requests in the scheduling process. If a vehicle is assigned with a request, its schedule settled by the control center needs to satisfy the following requirements:

1) Complete route specification: Since the vehicle is unmanned, we need to specify the *exact* route so that the vehicle can follow the route to pick the passengers of the assigned requests up and to drop them off at the required destinations. Moreover, the route should be short enough so that it has sufficient fuel to complete the route. The vehicle should end up at a refuel station to avoid breaking down in the middle of any road segments. This can guarantee that the vehicle must be able to refuel after completing all the assigned services.

2) Time constraints: The vehicle should be able to pick the passengers up at a time within the service starting time window specified in the request. Moreover, the actual ride time should be no longer than the maximum value stated in the request.

3) Capacity constraints: When the vehicle arrives at the pickup location, there should always be enough free seats available to accommodate all the passengers of the request.

Admission control and scheduling are inter-related and we will discuss their details in the subsequent sections. After determining the result, the control center then distributes the assignments to the corresponding AVs, which provide services to the customers.

IV. Scheduling

Scheduling involves determining the following:

- the assignment of AVs to the requests;
- the routes of AVs to accomplish the assigned requests; and
- the times by which the AVs should reach particular locations.

Here we assume that all requests being scheduled are admissible, where the admittability of a request is handled by admission control. Thus all requests will be served by appropriate vehicles after scheduling. When discussing admission control in Section V, we will explain the relationship between admission control and scheduling.

To facilitate scheduling, we assume that all vehicles are connected and can communicate with the control center with reasonably short delays. This ensures that no apparent changes in positions happen to the AVs in each interval given in Fig. 1. With the support of modern advanced communication technologies, this assumption can go through. In our model, we require that the computation of scheduling can be done in a short period of time. This ensures the validity of the traffic data when the vehicles traverse along their assigned routes. There are basically two types of traffic data: the distances and travel times of road segments. The former is time-invariant while the latter usually changes gradually. In other words, significant changes in travel times only take place in a timespan much longer than the time interval.

A. Preprocessing

We schedule the AVs to accomplish the transportation requests to achieve the minimum total operational cost in terms of fuel costs, which are in turn measured by the total distance traveled. The distance between any pair of locations is invariant and we transform $G(\mathcal{V},\mathcal{E})$ to $G'(\mathcal{V}',\mathcal{E}')$ with any shortest path algorithm, e.g., Dijkstra's algorithm [29], where $\mathcal{V}' \subset \mathcal{V}$ is the set of locations at which we need to determine the arrival times of the assigned AVs in order to configure their travel schedules. $\mathcal{V}'$ includes the first locations visited by all the vehicles (i.e., a_k's), the sources and destinations of the requests (i.e., s_r's and d_r's, respectively), and the locations of the refuel stations (i.e., $i \in \tilde{\mathcal{V}}$). $\mathcal{E}'$ is defined as $\{(i,j)|i,j \in \mathcal{V}'\}$ such that there exists a shortest path from $i \in \mathcal{V}$ to $j \in \mathcal{V}$ in G. For $(i,j) \in \mathcal{E}'$, the associated c_{ij} and t_{ij} are the sums of costs and times, respectively, of all the edges constituting the corresponding shortest path in G. In the subsequent computation, we focus on $G'(\mathcal{V}',\mathcal{E}')$ instead of $G(\mathcal{V},\mathcal{E})$. The reasons why we adopt this transformation are two-fold: First, the number of variables needed in the formulation can be dramatically reduced. The set $\mathcal{V}\backslash\mathcal{V}'$ are not important as all conditions confining to the locations specified by the vehicles and requests are restricted to $\mathcal{V}'$ only. In this way, the efficiency of solving the scheduling problem can be improved significantly. Second, this can improve the flexibility of the schedules. Consider that AV k goes from vertices 1 to 4 and there exist two paths connecting them as, Path 1: $1 \to 2 \to 4$, and Path 2: $1 \to 3 \to 4$. Suppose that vertices 1 and 4 belong to $\mathcal{V}'$ but vertices 2 and 3 do not. To satisfy the requirements imposed on k, we need to determine the times by which k should arrive at vertices 1 and 4 only, i.e., t_1^k and t_4^k. If vertices 2 and 3 are also included in the formulation and Path 1 is finally chosen, t_2^k will be specified by solving the scheduling problem and thus k needs to arrive at the vertices by t_1^k, t_2^k, and t_4^k, respectively. If not, only t_1^k and t_4^k are specified and we can give flexibility to k of arriving at vertex 2. t_2^k can be any time between t_1^k and t_4^k as long as the required travel times spent on $(1,2)$ and $(2,4)$ have been considered. This flexibility gives room for k to respond to any instantaneous traffic incidents which may disturb its original travel plan. This also allows k to change to Path 2, if needed, without altering the original travel plan.

Note that the preprocessing step can be skipped if the scheduling problem constructed directly from $G(\mathcal{V},\mathcal{E})$ can be solved efficiently. However, if the preprocessing is required to simplify the scheduling problem, it can be considered as a number of result lookups. As c_{ij}'s generally refer to the travel distances which are invariant, the results of the shortest path computations are also invariant. In fact, before the system operates, we can first compute the shortest path for every pair of locations in $\mathcal{V}$. When the preprocessing is triggered in an interval, we just need to look up the pre-computed shortest path results. Hence, the time cost of preprocessing can be considered negligibly small.

B. Problem Formulation

We formulate the scheduling problem based on $G'(\mathcal{V}',\mathcal{E}')$. The given data for the problem parameters include the graph $G'(\mathcal{V}',\mathcal{E}')$ with costs c_{ij}'s and travel times t_{ij}'s, the set of transportation requests $\mathcal{R}$, and the set of AVs $\mathcal{K}$. We define several variables for the problem. Binary variables x_{ij}^k's are used to indicate which connections will be traversed by the vehicles, as

$$x_{ij}^k = \begin{cases} 1 & \text{if vehicle } k \text{ traverses } (i,j), \\ 0 & \text{otherwise.} \end{cases}$$

We define binary variables y_r^k's for the assignment of the vehicles to the requests, as

$$y_r^k = \begin{cases} 1 & \text{if vehicle } k \text{ is assigned to request } r, \\ 0 & \text{otherwise.} \end{cases}$$

For $i \in \tilde{\mathcal{V}}$, binary variables g_i^k's are utilized to indicate the refuel stations at which the vehicles end their routes, as

$$g_i^k = \begin{cases} 1 & \text{if vehicle } k \text{ ends its route at vertex } i \in \tilde{\mathcal{V}}, \\ 0 & \text{otherwise.} \end{cases}$$

We need to specify the times and occupancy conditions at various locations along the routes. Let t_i^k be the time by which k should arrive at vertex i and f_i^k be the number of passengers in k right before it leaves i.

We aim to construct economical schedules for the AVs and thus we minimize the total operational cost with the objective function as

$$\sum_{i,j\in\mathcal{V},k\in\mathcal{K}} c_{ij}x_{ij}^k. \tag{2}$$

We define a set of constraints to confine the scope of the variables so that the requirements discussed in Section III-B are satisfied. Each transportation request can only be served once and thus we have

$$\sum_{k\in\mathcal{K}} y_r^k = 1, \forall r \in \mathcal{R}. \tag{3}$$

Each AV will end at one of the refuel stations if it is assigned to a request. This is specified by

$$\sum_{i\in\tilde{\mathcal{V}}} g_i^k \leq 1, \forall k \in \mathcal{K}. \tag{4}$$

If AV k is not assigned to any request, we do not need to determine a path for k so as the final stopping refuel station for k. Thus it is possible to have $\sum_{i\in\tilde{\mathcal{V}}} g_i^k = 0$ for some k.

Let $\mathcal{N}^+(i)$ and $\mathcal{N}^-(i)$ be the sets of incoming and outgoing neighbors of vertex i, i.e., $\mathcal{N}^+(i) = \{j \in \mathcal{V}' | (j,i) \in \mathcal{E}'\}$ and $\mathcal{N}^-(i) = \{j \in \mathcal{V}' | (i,j) \in \mathcal{E}'\}$. We model a path with a network flow model. A path starting at a_k and ending at $i \in \tilde{\mathcal{V}}$ can be defined with the following:

$$0 \leq \sum_{i\in\mathcal{N}^-(a_k)} x_{a_k i}^k - \sum_{i\in\mathcal{N}^+(a_k)} x_{i a_k}^k \leq \sum_r y_r^k, \forall k \in \mathcal{K}, \quad (5)$$

$$0 \leq \sum_{j\in\mathcal{N}^+(i)} x_{ji}^k - \sum_{j\in\mathcal{N}^-(i)} x_{ij}^k \leq g_i^k, \forall i \in \tilde{\mathcal{V}}, k \in \mathcal{K}, \quad (6)$$

$$\sum_{j\in\mathcal{N}^+(i)} x_{ji}^k = \sum_{j\in\mathcal{N}^-(i)} x_{ij}^k, \forall i \in \mathcal{V}' \setminus \tilde{\mathcal{V}} \cup \{a_k | k \in \mathcal{K}\}. \quad (7)$$

Eq. (5) defines for the starting vertex of k, where a starting vertex has one unit of net outgoing flow. $\sum_r y_r^k$ specifies if a path needs to be defined for k. If there are no requests assigned to k, $\sum_r y_r^k$ becomes zero and a_k is not the starting vertex of any paths for k. Similarly, (6) defines for the destination vertex of k and the exact vertex i ended by k is indicated by g_i^k. If k ends at $i \in \tilde{\mathcal{V}}$, (6) will allow i to have one unit of net incoming flow for k. For other vertices, (7) sets the conversation of flow by equalizing the corresponding incoming and outgoing flows.

If request r is assigned to vehicle k, k needs to pass through the pickup location s_r of r. It is equivalent to having positive outgoing flow for k at s_r as

$$\sum_{i\in\mathcal{N}^-(s_r)} x_{s_r i}^k \geq y_r^k, \forall r \in \mathcal{R}, k \in \mathcal{K}. \quad (8)$$

Similarly, k needs to pass through the dropoff point d_r of request r when r is served by k. This requires positive incoming flow for k at d_r as

$$\sum_{i\in\mathcal{N}^+(d_r)} x_{i d_r}^k \geq y_r^k, \forall r \in \mathcal{R}, k \in \mathcal{K}. \quad (9)$$

Note that specifying incoming flow for s_r is not sufficient as it is possible to have zero incoming flow when k begins its path at s_r exactly. Similarly, it is not sufficient to specify outgoing flow for d_r as it is possible to have zero outgoing flow when k ends its path at d_r.

No matter where vehicle k goes, it cannot travel continuously longer than its operational time limit specified by $\tilde{T}_k$. Moreover, it needs to take at least t_k^0 in order to reach the initial vertex of its path. Hence we have

$$t_k^0 \leq t_i^k \leq \tilde{T}_k, \forall i \in \mathcal{V}', k \in \mathcal{K}. \quad (10)$$

Let M be a sufficiently large positive number. When vehicle k traverses edge (i,j), the time at j should be larger than or equal to the time at i together with the travel time on (i,j), i.e., t_{ij}. This can be specified by

$$t_j^k \geq t_i^k + t_{ij} - M(1 - x_{ij}^k), \forall k \in \mathcal{K}, i,j \in \mathcal{V}'. \quad (11)$$

When vehicle k is assigned to request r, the actual ride time to reach d_r from s_r should be no larger than the maximum ride time T_r specified by r, i.e.,

$$t_{d_r}^k - t_{s_r}^k \leq T_r + M(1 - y_r^k), \forall r \in \mathcal{R}, k \in \mathcal{K}. \quad (12)$$

If request r is served by vehicle k, k should arrive at s_r within the service starting time window $[e_r, l_r]$ specified by r. This can be expressed as

$$e_r - M(1 - y_r^k) \leq t_{s_r}^k \leq l_r + M(1 - y_r^k), \forall r \in \mathcal{R}, k \in \mathcal{K}. \quad (13)$$

Passengers being served occupy seats and the capacity limits of all vehicles should be satisfied at all times. So we have

$$0 \leq f_i^k \leq Q_k, \forall i \in \mathcal{V}', k \in \mathcal{K}. \quad (14)$$

At a_k, some passengers induced from $\tilde{\mathcal{R}}_k$ may get off k and new passengers may get on k from other requests. The occupancy conditions of the AVs at their initial vertices a_k's are given by

$$f_{a_k}^k \geq \sum_{r|s_r=a_k} q_r y_r^k - \sum_{r|d_r=a_k} q_r y_r^k, \forall k \in \mathcal{K}. \quad (15)$$

When k traverses from i to j along (i,j), vertex j may be the pickup locations of some requests and dropoff locations of some other requests. The relationship between the occupancy conditions of AV k at i and j can be specified as

$$f_j^k \geq f_i^k - M(1 - x_{ij}^k) + \sum_{r|s_r=a_k} q_r y_r^k - \sum_{r|d_r=a_k} q_r y_r^k, \quad (16)$$
$$\forall i,j \in \mathcal{V}', k \in \mathcal{K}.$$

When an AV reaches a refuel station, all requests assigned to it should have been settled and no passenger should be accompanied to the end of the route. This is described by

$$f_i^k \leq M(1 - g_i^k), \forall i \in \tilde{V}, k \in \mathcal{K}. \quad (17)$$

Recall that there are two kinds of requests which have already been assigned to the AVs before the current scheduling interval, i.e., $\mathcal{R}_k = \tilde{\mathcal{R}}_k \cup \overline{\mathcal{R}}_k$. As a (nearly) real-time application, with updated information, we may further improve the system performance by revising the already assigned requests. For those requests currently being served, e.g., $r \in \tilde{\mathcal{R}}_k$ with the passengers sitting in k, we can consider those r's as "new" requests starting the service at the the starting node a_k by setting $s_r = a_k$ and affirming $y_r^k = 1$. As k has been serving r by following a previously determined schedule, we can update its T_r by shortening the elapsed time. The service starting time window is no longer important and thus we set $e_r = -\infty$ and $l_r = +\infty$. There is no change to q_r. For those requests $\overline{\mathcal{R}}_k$'s which have been previously assigned to k but not yet been served, we may reschedule $r \in \overline{\mathcal{R}}_k$ with other AVs if it can result in lower cost. As the passengers do not concern about which vehicle would eventually provide the service, it may be more efficient to re-allocate those r's in $\overline{\mathcal{R}}_k$ to other more appropriate vehicles with lower operational cost. This enhances the flexibility of the system. As a whole, the scheduling problem is defined as

Problem 1 (Scheduling):

$$\begin{aligned} \text{minimize} \quad & (2) \\ \text{subject to} \quad & (3)-(17) \\ \text{over} \quad & x_{ij}^k \in \{0,1\}, y_r^k \in \{0,1\}, g_l^k \in \{0,1\}, t_i^k \in \mathbb{R}^+, \\ & f_i^k \in \mathbb{Z}^+, \forall i,j \in \mathcal{V}', l \in \tilde{\mathcal{V}}, r \in \mathcal{R}, k \in \mathcal{K}. \end{aligned}$$

Problem 1 has a linear objective function and linear equality and inequality constraints. Some of its variables are binary while the rest are real. Thus the scheduling problem is a mixed-integer linear program (MILP). Although the preprocessing step discussed in Section IV-A helps simplify the problem, the numbers of variables and constraints also grow with the sizes of $\mathcal{R}$ and $\mathcal{K}$. As those invalid requests have been removed by admission control (discussed in Section V), this MILP is always feasible and all requests must be served. As long as all c_{ij}'s are positive, the solution of Problem 1 does not result in zero cost and the schedule without serving any requests will never be a solution.

C. Complete Schedule Construction

Since the vehicles are unmanned, we need to provide complete instructions about the paths and schedules so that they know when and where they should go in order to provide services to the customers. Solving the MILP gives the solutions for x_{ij}^k's, y_r^k's, b_k's , t_i^k's, and f_i^k's. As being binary variables, the results of x_{ij}^k's and y_r^k's are unambiguous. The latter tells which vehicles are assigned to the requests. The former explains the route of each k in G' starting at a_k and ending at one of the refuel stations. The paths determined in G' in turn infer the corresponding complete routes in G. Recall that we have determined the shortest path from i to j in G corresponding to the edge $(i,j) \in \mathcal{E}'$. By inserting the shortest paths for every pair of adjacent vertices along the paths based on G', the complete routes in G can be derived accordingly.

Note that (10)–(13) define the scope of t_i^k's in the form of inequality. The resulting t_i^k's make feasible time schedules but may not be specific enough leading to ambiguity. For example, if the arrival of k at location i at any moment in $[t', t'']$ is feasible, a reasonable way is to set $t_i^k = t'$ and this enhances the flexibility for the later scheduling intervals. To construct the schedule of k, we examine the path computed from x_{ij}^k's. For the first vertex, we set $t_{a_k}^k = t_k^0$. For any subsequent vertices, says from i to j, we can add the travel time on edge (i,j) to the settled time at i to obtain the settled time at j, i.e., $t_j^k = t_i^k + t_{ij}$. If vertex j induces a request, we need to fulfill its service starting time window and thus we have $t_j^k = \max\{t_i^k + t_{ij}, e_r\}$.

Similarly, (14)–(17) also confine the occupancies of the vehicles at various locations with inequalities. The exact seat conditions cannot be told from the resulting f_i^k's. Usually, we only concern about the seat conditions at the customer pickup and dropoff points, i.e., s_r's and d_r's. We can examine the route computed from x_{ij}^k's again and determine the occupancy conditions. For example, k goes from i to j on (i,j). If j is the service starting location of request r, we add the number of seats required for r to the occupancy of k at i to get its occupancy at j, i.e., $f_j^k = f_i^k + q_r$. If j is a service destination location instead, we subtract the seats taken by r from the occupancy of k at i to get its occupancy at j, as $f_j^k = f_i^k - q_r$. In this way, the complete schedules of the vehicles with duty assigned can be determined and the vehicles just need to follow the schedules to accomplish the services.

V. Admission Control

Recall that, in Section IV, all requests submitted for scheduling are assumed to be admissible and need to be served. In this section, we investigate the admission control problem. We first formulate the problem and then study the variations in the presence of traffic congestion and no-show of passengers.

A. Problem Formulation

Admission control is responsible for determining a set of requests suitable for scheduling. In other words, after admission control, we will produce a subset $\check{\mathcal{R}} \subset \mathcal{R}$ for subsequent scheduling, where $\mathcal{R}$ is the set of all available requests and $\check{\mathcal{R}}$ will be settled by appropriate AVs in scheduling. However, to judge if a particular request r is admissible, we need to check not only its feasibility but also its profitability, i.e., whether serving r will induce a positive net profit. Determining the net profit from r involves its induced cost, which is regulated through scheduling. Hence there is no clear precedence relationship between scheduling and admission control and these two processes should be considered simultaneously.

We can interpret the requests and AVs as the demand and supply of transportation services, respectively, and then the constraints of Problem 1 define the scope of matching between the demand and supply. The constraints can be satisfied more easily with larger $\mathcal{K}$ and smaller $\mathcal{R}$. Practically, the size of $\mathcal{K}$ is generally fixed as the system would not suddenly employ more AVs into the fleet or many AVs become out of service all of a sudden. However, the requests submitted are absolutely external from the system; the system can neither forbid the customers from submitting requests nor modify the attributes in the requests to match the conditions of AVs. In fact, just a single inappropriate request (e.g., a request with very short tolerable ride time) can make Problem 1 infeasible and the scheduling collapse. To avoid this, the system should perform admission control by screening out any inappropriate requests before undergoing the scheduling (see Fig. 2). Consider that entertaining a request results in revenue. Although the system cannot modify the submitted requests, it has the right to dismissing any requests by sacrificing the corresponding revenue. Admission control manipulates $\mathcal{R}$ with the following objectives: 1) Produce a subset of requests $\check{\mathcal{R}} \subset \mathcal{R}$ so that the scheduling process can be performed, i.e., Problem 1 is made feasible with $\check{\mathcal{R}}$; 2) Maximize the profit incurred.

Consider that we admit $\check{\mathcal{R}}$ for scheduling with Problem 1, which can be re-written as

$$\begin{aligned} \text{minimize} \quad & \phi(\alpha) && \text{(18a)} \\ \text{subject to} \quad & \alpha \in \mathcal{Z}(\check{\mathcal{R}}), && \text{(18b)} \end{aligned}$$

where $\alpha \triangleq \{x_{ij}^k\} \cup \{y_r^k\} \cup \{g_l^k\} \cup \{t_i^k\} \cup \{f_i^k\}$, $\phi(\alpha) \triangleq \sum_{i,j \in \mathcal{V}, k \in \mathcal{K}} c_{ij} x_{ij}^k$, and let $\mathcal{Z}(\check{\mathcal{R}})$ be the feasible region of

Problem 1 with respect to $\check{\mathcal{R}}$. Let ρ_r be the revenue made when admitting $r \in \mathcal{R}$ and define

$$z_r = \begin{cases} 1 & \text{if we admit } r \in \mathcal{R} \text{ for scheduling,} \\ 0 & \text{otherwise.} \end{cases}$$

We also define the admission function $\sigma(\mathcal{R}, [z_r]_{r\in\mathcal{R}})$ which returns $\check{\mathcal{R}} \subset \mathcal{R}$ based on z_r such that $r \in \check{\mathcal{R}}$ if $z_r = 1$. The total profit is the difference between the total revenue and total cost, i.e., $\sum_{r\in\mathcal{R}} \rho_r z_r - \phi(\alpha)$. Then we formulate the admission control problem as

Problem 2 (Admission Control):

$$\text{maximize} \quad \Phi(\mathcal{R}, [z_r]_{r\in\mathcal{R}}) = \sum_{r\in\mathcal{R}} \rho_r z_r - \phi(\alpha) \tag{19a}$$

$$\text{subject to} \quad \check{\mathcal{R}} = \sigma(\mathcal{R}, [z_r]_{r\in\mathcal{R}}), \tag{19b}$$

$$z_r = 1, \forall r \in \mathcal{R}_k, k \in \mathcal{K}, \tag{19c}$$

$$\alpha \in \arg\min\{\phi(\alpha) : \alpha \in \mathcal{Z}(\check{\mathcal{R}})\}, \tag{19d}$$

$$\text{over} \quad \alpha, \check{\mathcal{R}} \in \mathcal{R}, z_r \in \{0, 1\}, \forall r \in \mathcal{R}, \tag{19e}$$

where (19c) ensures that those requests admitted in the previous operating intervals will still be admitted in the current interval. We cast admission control as a bilevel optimization problem, which consists of an upper- and a lower-level optimization. Φ is the upper-level objective function with upper-level variables $\check{\mathcal{R}}$ and z_r's. ϕ represents the lower-level objective function with lower-level variable α. Eq. (19d) is in fact (18), and thus, we cast Problem 1 as a constraint of Problem 2. The upper-level optimization is to manipulate the whole set of requests $\mathcal{R}$ and determine $\check{\mathcal{R}}$ such that $\check{\mathcal{R}}$ can maximize the total profit. The lower-level optimization is to schedule the AVs to serve the set of admissable requests $\check{\mathcal{R}}$ so that the retained cost is the lowest. The two levels of optimization are inter-related; the upper level requires the result of the lower level, i.e., α, in order to get $\check{\mathcal{R}}$, while the lower level needs the result from the upper level, i.e., $\check{\mathcal{R}}$, in order to output α. Note that if the upper level produces $\check{\mathcal{R}}$ which makes $\mathcal{Z}$ infeasible, the resulting α will return $+\infty$ for the objective function of (19d), which will in turn make the objective function (19a) retain $-\infty$.

Bilevel optimization is in general difficult to solve. A bilevel problem with a linear objective function and linear constraints is NP-hard [30]. As seen from (19), we are manipulating discrete variables in the problem. As classical methods for bilevel optimization usually assume smoothness or convexity [31], those classical methods are not applicable to Problem 2. As inspired by [32], [33], we decide to tackle the problem with an evolutionary heuristic approach. Evolutionary approaches are commonly applied to bilevel optimization problems in transport science. For example, in [34], Differential Evolution (DE) is employed to address the optimal toll problem, which is about setting polls to control congestion, and the road network design problem, which determines the capacity enhancements of network facilities. In [35], GA is applied to the transit road space priority problem, which optimizes the system by reallocating the road space between private car and transit modes. We will design a GA-based algorithm to solve Problem 2. Before discussing the details of the algorithm, we define admissibility and give some analytical results for Problem 2, which can help design the algorithm in the next section.

Definition 1 (Admissibility): A set of requests $\check{\mathcal{R}}$ is admissible if $[z_r]_{r\in\mathcal{R}}$ produces $\check{\mathcal{R}}$, which results in finite profit, i.e., $\Phi(\mathcal{R}, [z_r]_{r\in\mathcal{R}}) > -\infty$.

Theorem 1: We have the following results for admissibility:

1) Consider that a subset of requests $\check{\mathcal{R}} \subset \mathcal{R}$ are admissible. Let $\mathcal{P}(\check{\mathcal{R}})$ be the power set of $\check{\mathcal{R}}$. Any $\check{\mathcal{R}}' \in \mathcal{P}(\check{\mathcal{R}})$ is also admissible.
2) For any singleton $\{r\} \subset \mathcal{R}$, if $\{r\}$ is not admissible, any superset $\check{\mathcal{R}} \supset \{r\}$ are also non-admissible.
3) Consider subsets of requests, $\check{\mathcal{R}}_1, \check{\mathcal{R}}_2 \subset \mathcal{R}$, and subsets of vehicles $\check{\mathcal{K}}_1, \check{\mathcal{K}}_2 \subset \mathcal{K}$. Suppose $\check{\mathcal{K}}_1 \cap \check{\mathcal{K}}_2 = \emptyset$. If $\check{\mathcal{R}}_1$ and $\check{\mathcal{R}}_2$ are admissible by $\check{\mathcal{K}}_1$ and $\check{\mathcal{K}}_2$, respectively, then $\check{\mathcal{R}}_1 \cup \check{\mathcal{R}}_2$ are also admissible.

Proof: For Statement 1, Constraint (19b) defines $\check{\mathcal{R}}$, which is an input of Constraint (19d). It is sufficient to show that the removal of any $r \in \check{\mathcal{R}}$ will not make (19d) infeasible if the participating AVs can serve all the requests in $\check{\mathcal{R}}$. Suppose that r is removed from $\check{\mathcal{R}}$ and AV k would have assigned to serve r if r had been admitted. k can still follow the path as if r is present. Hence (19d) is still feasible for $\check{\mathcal{R}} \setminus r$.

For Statement 2, a non-admissible r means that it is impossible to arrange an AV to entertain r. We will never be able to provide services to a set of requests containing r as its component r can never be served.

For Statement 3, we can represent $\check{\mathcal{R}}_1 \cup \check{\mathcal{R}}_2$ by three non-overlapping sets $\check{\mathcal{R}}_1 \setminus (\check{\mathcal{R}}_1 \cap \check{\mathcal{R}}_2)$, $\check{\mathcal{R}}_2 \setminus (\check{\mathcal{R}}_1 \cap \check{\mathcal{R}}_2)$, and $\check{\mathcal{R}}_1 \cap \check{\mathcal{R}}_2$. Since $\check{\mathcal{K}}_1$ and $\check{\mathcal{K}}_2$ are mutually exclusive, $\check{\mathcal{R}}_1 \setminus (\check{\mathcal{R}}_1 \cap \check{\mathcal{R}}_2)$ and $\check{\mathcal{R}}_2 \setminus (\check{\mathcal{R}}_1 \cap \check{\mathcal{R}}_2)$ can be served by $\check{\mathcal{K}}_1$ and $\check{\mathcal{K}}_2$ simultaneously. Each $r \in \check{\mathcal{R}}_1 \cap \check{\mathcal{R}}_2$ can be admitted by either $k \in \check{\mathcal{K}}_1$ or $k \in \check{\mathcal{K}}_2$. ■

Lemma 1: The system will not make negative profit. That is, for any $\mathcal{R}$, Problem 2 must have at least one feasible solution whose objective function value is non-negative.

Proof: We separate $\mathcal{R}$ into the previously admitted and newly received requests, i.e., $\{\mathcal{R}_k\}$ and $\mathcal{R} \setminus \{\mathcal{R}_k\}$.

For the newly received requests, we can always set $z_r = 0, \forall r \in \mathcal{R} \setminus \{\mathcal{R}_k\}$. Then (19b) gives $\check{\mathcal{R}} = \emptyset$. Eq. (19d) returns α with $\phi(\alpha) = 0$ as no requests need to be served and thus no AVs have been used to provide service. Hence we have $\sum_{r\in\mathcal{R}\setminus\{\mathcal{R}_k\}} \rho_r z_r - \phi(\alpha) = 0$.

For the previously admitted requests, since they are admitted in some previous admission control processes, they must incur non-negative profit when they were admitted as new requests before. Otherwise, we would not have admitted them at the first place. ■

Lemma 1 implies that Problem 2 must be feasible.

Theorem 2: Consider two subsets of requests $\check{\mathcal{R}}$ and $\check{\mathcal{R}}'$ with $\check{\mathcal{R}} \subset \check{\mathcal{R}}' \subset \mathcal{R}$. If both $\check{\mathcal{R}}$ and $\check{\mathcal{R}}'$ are admissible, then $\check{\mathcal{R}}'$ will not be less profitable than $\check{\mathcal{R}}$, i.e., $\sup \Phi(\check{\mathcal{R}}, \{z_r | r \in \check{\mathcal{R}}\}) \leq \sup \Phi(\check{\mathcal{R}}', \{z_r | r \in \check{\mathcal{R}}'\})$.

Proof: Suppose $\sup \Phi(\check{\mathcal{R}}, \{z_r | r \in \check{\mathcal{R}}\}) > \sup \Phi(\check{\mathcal{R}}', \{z_r | r \in \check{\mathcal{R}}'\})$. We write $\check{\mathcal{R}}' = \check{\mathcal{R}} \cup (\check{\mathcal{R}}' \setminus \check{\mathcal{R}})$. Then we have

$$\sup \Phi(\check{\mathcal{R}}', \{z_r | r \in \check{\mathcal{R}}'\})$$

$$= \sup \Phi(\check{\mathcal{R}}, \{z_r | r \in \check{\mathcal{R}}\}) + \sup \Phi(\check{\mathcal{R}}' \setminus \check{\mathcal{R}}, \{z_r | r \in \check{\mathcal{R}}' \setminus \check{\mathcal{R}}\}).$$

By Lemma 1, $\sup \Phi(\check{\mathcal{R}}' \setminus \check{\mathcal{R}}, \{z_r | r \in \check{\mathcal{R}}' \setminus \check{\mathcal{R}}\})$ has a value larger than or equal to zero. This induces a contradiction. ■ Theorem 2 implies that entertaining more requests will not reduce the amount of profit made.

B. Variations

Here we investigate how traffic congestion and no-show of paasengers impact on admission control (and scheduling). Basically, we will see that under these circumstances, the proposed admission control and scheduling mechansims can still be applied but we may need some additional minor arrangements to handle various situations.

1) Traffic Congestion: Traffic congestion has direct impact on the travel time t_{ij} for some $(i, j) \in \mathcal{E}$ and subsequently affects the admissibility of requests. Recall that the system operates in a fixed-interval basis and each interval generally lasts for a few minutes (see Section III-B). We basically assume that, within an interval, the parameters, including the travel times, are constant or with very small changes such that the results of admission control completed for that interval are still valid. If the travel times are relatively fast changing, we need to shorten the duration of the intervals to make the assumption valid. On the other hand, if the travel times are slowly varying, we may lengthen the durations to reduce the computation burden. Hence, the duration of the operating intervals depends on the traffic conditions of the deployed service area.

Now consider that t_{ij} in the current interval has been updated such that its value is different from that used in the previous interval. There are three cases for the possible influence: (i) t_{ij} does not involve in $\mathcal{R}_k$ for all k; (ii) t_{ij} involves in $\overline{\mathcal{R}}_k$ for some k; and (iii) t_{ij} involves in $\tilde{\mathcal{R}}_k$ for some k. For Case (i), since t_{ij} has not been used to serve any requests, its change does not affect the schedules of any AVs. Hence, nothing needs to be done solely based on t_{ij}. For Case (ii), although t_{ij} has been used to determine the schedules of some AVs, the involved requests have not been served yet. We can simply consider these requests as newly submitted requests and perform admission control and scheduling with them again. For Case (iii), t_{ij} affects those schedules which are being implemented by some AVs. In the subsequent intervals, the scheduling process will see if the road segment (i, j) can be avoided by determining other shortest paths. If not, as the passengers are being served, it may not be appropriate to ask them to shift to other vehicles for their journeys and nothing can be done further operationally. However, we may compensate the passengers in the marketing perspective, e.g., by issuing cash coupons for future rides.

2) No-show of Passengers: No-show refers to the situation that some or all passengers of a paricular request are absent at the scheduled pickup time. If a passenger cannot arrive at the pickup location on time, this will be considered as no-show. If some but not all passengers are absent, the schdule of the designated AV is unaffected but fewer seats are required. These unused seats can be released to serve other appropriate requests in the later intervals. If all passengers are absent, the "resources" allocated to the request can be released in the subsequent intervals right after its original pickup time. This gives the AV more flexibility in time and occupancy to serve future requests. In the business perspective, there may exist some penalty policies to discourage such activities.

VI. Genetic-Algorithm-Based Solution Method

In this section, we propose a solution method to tackle Problem 2. We adopt a GA-based framework to structure the method. Some of its components are designed based on the analytical results discussed in Section V.

A. Working Principle of Evolutionary Algorithms

Evolutionary Algorithms (EAs) refer to a class of optimization algorithms, whose designs are inspired by various natural phenomena. Examples include GA [36], DE [37], and Chemical Reaction Optimization (CRO) [38]. Different EAs generally have similar working principles: An EA samples the solution space of the problem iteratively and tries to locate a global optimum after examining a limited number of candidate solutions in the solution space. In each iteration, with some operators, it generates a population of candidate solutions based on those obtained from the previous iterations and their corresponding objective function values. It tends to converge to the global optimum along the iterations and it terminates when a stopping criterion is matched. Different EAs have different designs of their operators. For example, GA is designed based on the ideas of natural selection in genetics while CRO mimics the nature of chemical reaction processes. Unlike most of the traditional optimization approaches, EAs require the problem to be neither convex nor differentiable. In each algorithm run, they only need to sample a number of candidate solutions and evaluate their solution qualities with the objective function. Hence, a search with an EA usually incurs many objective function calls. As discussed, EAs have been shown effective in solving bilevel optimization problems in transport science. We are going to adopt the well-established GA framework to facilitate the design of a method which can return good solutions for Problem 2 in a practical sense.

B. Distributed Scheduling

When an EA is employed to address Problem 2, many candidate solutions will be generated. To evaluate the quality of a particular candidate solution, we need to compute (19a) once, which also needs to examine (19d) one time. In other words, a single run of EA requires to solve Problem 1 many times. When the lower-level optimization is simple, the computational burden of solving it many times may still be acceptable. However, this is not the case for Problem 1, where the required numbers of variables and constraints grow exponentially with the quantities of transportation requests and serving AVs. This implies that we need to a more effective way to solve Problem 1, in order to tackle Problem 2.

Consider that $\check{\mathcal{R}}_k \subset \mathcal{R}$ is the subset of requests assigned to vehicle k. Suppose that we know the distribution of the

requests to the vehicles, i.e., $\check{\mathcal{R}}_k$ for all k. Since each request is only served by one vehicle, we have $\check{\mathcal{R}}_k \cap \check{\mathcal{R}}_l = \emptyset$, for any $k, l \in \mathcal{K}$, $k \neq l$, and $\bigcup_{k\in\mathcal{K}} \check{\mathcal{R}}_k = \mathcal{R}$. When given $\check{\mathcal{R}}_k$, we consider the following problem:

Problem 3 (Scheduling Subproblem for vehicle k):

$$\text{maximize} \quad \sum_{i,j\in\mathcal{V}'} c_{ij}\mathring{x}^k_{ij} \tag{20a}$$

$$\text{subject to} \quad \sum_{i\in\tilde{\mathcal{V}}} \mathring{g}^k_i \leq 1, \tag{20b}$$

$$0 \leq \sum_{i\in\mathcal{N}^-(a_k)} \mathring{x}^k_{a_k i} - \sum_{i\in\mathcal{N}^+(a_k)} \mathring{x}^k_{i a_k} \leq \sum_r \mathring{y}^k_r, \tag{20c}$$

$$0 \leq \sum_{j\in\mathcal{N}^+(i)} \mathring{x}^k_{ji} - \sum_{j\in\mathcal{N}^-(i)} \mathring{x}^k_{ij} \leq \mathring{g}^k_i, \forall i \in \tilde{\mathcal{V}}, \tag{20d}$$

$$\sum_{j\in\mathcal{N}^+(i)} \mathring{x}^k_{ji} = \sum_{j\in\mathcal{N}^-(i)} \mathring{x}^k_{ij}, \forall i \in \mathcal{V}' \setminus \tilde{\mathcal{V}} \cup \{a_k\} \tag{20e}$$

$$\sum_{i\in\mathcal{N}^-(s_r)} \mathring{x}^k_{s_r i} \geq \mathring{y}^k_r, \forall r \in \check{\mathcal{R}}_k, \tag{20f}$$

$$\sum_{i\in\mathcal{N}^+(d_r)} \mathring{x}^k_{i d_r} \geq \mathring{y}^k_r, \forall r \in \check{\mathcal{R}}_k, \tag{20g}$$

$$\mathring{t}^0_k \leq \mathring{t}^k_i \leq \tilde{T}_k, \forall i \in \mathcal{V}', \tag{20h}$$

$$\mathring{t}^k_j \geq \mathring{t}^k_i + \mathring{t}_{ij} - M(1 - \mathring{x}^k_{ij}), \forall i, j \in \mathcal{V}' \tag{20i}$$

$$\mathring{t}^k_{d_r} - \mathring{t}^k_{s_r} \leq T_r + M(1 - \mathring{y}^k_r), \forall r \in \check{\mathcal{R}}_k, \tag{20j}$$

$$e_r - M(1 - \mathring{y}^k_r) \leq \mathring{t}^k_{s_r} \leq l_r + M(1 - \mathring{y}^k_r), \forall r \in \check{\mathcal{R}}_k, \tag{20k}$$

$$0 \leq \mathring{f}^k_i \leq Q_k, \forall i \in \mathcal{V}', \tag{20l}$$

$$\mathring{f}^k_{a_k} \geq \sum_{r|s_r=a_k} q_r \mathring{y}^k_r - \sum_{r|d_r=a_k} q_r \mathring{y}^k_r, \tag{20m}$$

$$\mathring{f}^k_j \geq \mathring{f}^k_i - M(1 - \mathring{x}^k_{ij}) + \sum_{r|s_r=a_k, r\in\check{\mathcal{R}}_k} q_r \mathring{y}^k_r - \sum_{r|d_r=a_k, r\in\check{\mathcal{R}}_k} q_r \mathring{y}^k_r, \forall i, j \in \mathcal{V}', \tag{20n}$$

$$\mathring{f}^k_i \leq M(1 - \mathring{g}^k_i), \forall i \in \tilde{V}, \tag{20o}$$

$$\text{over} \quad \mathring{x}^k_{ij} \in \{0,1\}, \mathring{y}^k_r \in \{0,1\}, \mathring{g}^k_l \in \{0,1\}, \mathring{t}^k_i \in \mathbb{R}^+, \mathring{f}^k_i \in \mathbb{Z}^+, \forall i, j \in \mathcal{V}', l \in \tilde{\mathcal{V}}, r \in \check{\mathcal{R}}_k. \tag{20p}$$

Solving Problem 3 only allows us to obtain the serving path, the schedule to reach various locations along the path, and the capacity conditions of vehicle k for serving the requests indicated by $\check{\mathcal{R}}_k$. Problem 3 looks similar to Problem 1 but indeed much simpler. It does not contain (3) and it manipulates fewer variables as those related to vehicles other than k are not included. It also possesses fewer constraints because of fewer variables.

For simplicity, similar to (18), we also write the solution, objective function, and the solution space of Problem 3 as $\mathring{\alpha}_k$, $\phi_k(\mathring{\alpha}_k)$ and $\mathcal{Z}_k$, respectively.

Theorem 3: When given $\check{\mathcal{R}}_k \subset \mathcal{R}, \forall k \in \mathcal{K}$, such that $\check{\mathcal{R}}_k \cap \check{\mathcal{R}}_l = \emptyset$, for any $k, l \in \mathcal{K}$, $k \neq l$, and $\bigcup_{k\in\mathcal{K}} \check{\mathcal{R}}_k = \mathcal{R}$, solving Problem 3 for all $k \in \mathcal{K}$ is equivalent to solving Problem 1, i.e.,

$$\inf_{\alpha\in\mathcal{Z}} \phi(\alpha) = \sum_{k\in\mathcal{K}} \inf_{\mathring{\alpha}_k\in\mathcal{Z}_k} \phi_k(\mathring{\alpha}_k),$$

and $x^k_{ij} = \mathring{x}^k_{ij}$, $y^k_r = \mathring{y}^k_r$, $g^k_l = \mathring{g}^k_l$, $t^k_i = \mathring{t}^k_i$, and $f^k_i = \mathring{f}^k_i$, $\forall i, j \in \mathcal{V}', l \in \tilde{\mathcal{V}}, r \in \check{\mathcal{R}}_k, k \in \mathcal{K}$.

Proof: When given such $\check{\mathcal{R}}_k \subset \mathcal{R}, \forall k \in \mathcal{K}$, we can construct $y^k_r, \forall r \in \mathcal{R}, k \in \mathcal{K}$, such that (3) holds. In this way, we can remove (3) from Problem 1. Without Constraint (3), the objective function and the rest of the constraints of Problem 1 become separable in terms of k: (2) gives the sum of costs spent on the vehicles; Eqs. (4)–(9) specify the paths traversed by the vehicles, each of which are independent; Eqs. (10)–(13) confine the time requirements at various locations along the vehicular paths; Eqs. (14)–(17) limit the passenger capacity conditions along the vehicular paths. If we group the terms of (2) and the constraints (4)–(17) for each k, we will have $|\mathcal{K}|$ problems, each of which is given by (20). ■

Theorem 3 states that when the assignment of requests to the vehicles is known, solving the $|\mathcal{K}|$ individual scheduling subproblems distributedly can retain the solution of the original scheduling problem. Note that this result is dedicatedly developed based on some characteristics of the problem formulations and it generally cannot be applied to the other scheduling problems. Unlike general distributed optimization [39], [40], our result here does not require techniques like message-passing. As a result, the $|\mathcal{K}|$ subproblems can be solved by $|\mathcal{K}|$ computing units distributedly. Assuming that the vehicles are connected through advanced vehicular communication technologies at all times, an obvious option of the computing unit is the AV. Thus, by Theorem 3, if we can assign each vehicle with the requests it needs to serve, each vehicle can determine a feasible path *per se* to serve the assigned requests with the lowest cost by solving Problem 3 concurrently. However, when the communications between a particular AV and the control center are interrupted, the corresponding subproblem can be delegated to an unoccupied computing unit at the control center or even to the cloud instead. The computed scheduling result can be returned to the AV when its communications have been resumed.

C. Algorithmic Components

Since GA is one of the most popular EAs, we adopt a GA-based design to address the admission control problem. GA generates a sequence of candidate solutions using operations inspired by natural evolution, e.g., inheritance, selection, crossover, and mutation. Here we introduce various algorithmic components before discussing the overall algorithmic design:

1) Chromosome: A chromosome specifies a candidate solution of Problem 2. While the lower-level optimization is handled by a standard MILP method, our GA approach is mainly used to handle the upper-level optimization. A chromosome is represented by a $1 \times |\mathcal{R}|$ binary vector $z =$

$[z_1, \ldots, z_r, \ldots, z_{|\mathcal{R}|}]$, together with a vehicle assignment vector $\kappa = [\kappa_1, \ldots, \kappa_r, \ldots, \kappa_{|\mathcal{R}|}]$, where κ_r represents the vehicle assigned to r if z_r is of unity. Note that the introduction of κ is the trick to carry out distributed scheduling discussed in Section VI-B. Although κ can be determined in (19d) if we apply the original formulation of scheduling (18), there is no harm in manipulating κ together with z in the chromosome level. This makes distributed scheduling feasible and the benefit of computation time saving will be clear in Section VII-A. During the course of search, we maintain a population of N_{pop} chromosomes.

2) Fitness Evaluation: We evaluate the fitness of each chromosome in a distributed manner. The fitness evaluation process is illustrated in Fig. 3 and it consists of five steps:

(1) Grouping requests in $\check{\mathcal{R}}_k$: Each chromosome i contains z^i and κ^i. For those r's with $z_r^i = 1$, based on κ^i, at the control center, we can divide $\mathcal{R}$ into $|\mathcal{K}|$ groups, i.e., $\check{\mathcal{R}}_k, \forall k \in \mathcal{K}$.
(2) Request information distribution: For each k, the control center transmits $\check{\mathcal{R}}_k$ to AV k, e.g., via VANET.
(3) Distributed scheduling: Modern vehicles are generally equipped with computers and thus each k can solve the individual Problem 3 simultaneously with other vehicles. Those AVs with empty $\check{\mathcal{R}}_k$ assigned can skip the computation.
(4) Individual cost return: The individual vehicles transmit the computed costs of scheduling to the control center, e.g., via VANET.
(5) Fitness computation: Based on Theorem 3, the cost associated to the chromosome is the sum of the objective function values of Problem 3 determined by the individual AVs, i.e., $\phi(\alpha) = \sum_{k\in\mathcal{K}} \phi_k(\mathring{\alpha}_k|\kappa_r = k)$. Then the fitness of the chromosome can be computed as $\Phi(z,\kappa) = \sum_{r\in\mathcal{R}} \rho_r z_r - \sum_{k\in\mathcal{K}} \phi_k(\mathring{\alpha}_k|\kappa_r = k)$.[2]

Note that $\Phi(z,\kappa)$ becomes $-\infty$ if and only if any request r with $z_r = 1$ is non-admissible. The advantages of undergoing the above process are three-fold:

(i) The computation time can be dramatically reduced. Among all the computation components in the algorithm, scheduling is the most computationally demanding. If each vehicle can compute their own schedules, all the individual scheduling subproblems can be solved simultaneously.
(ii) All entities need to manage the necessary data only. ρ_r is the result of the deal between the customer and the control center. With distributed scheduling, the usage of ρ_r is restricted to the control center and no vehicles are involved. Moreover, after a vehicle solves its scheduling subproblem, its computed schedule is stored in that vehicle only, but not the control center nor any other vehicles.
(iii) The amount of communications keeps minimal. In each evaluation, the only data needed to be communicated between the control center and the vehicles are the requests assigned to the individual vehicles (in Step 2) and the computed scheduling costs (in Step 4). The system does not require a sophisticated communication system to satisfy the communication requirements.

[2]By abuse of notation, we write $\Phi(z,\kappa) = \Phi(\mathcal{R}, [z_r])$ to emphasize the structure of the chromosome.

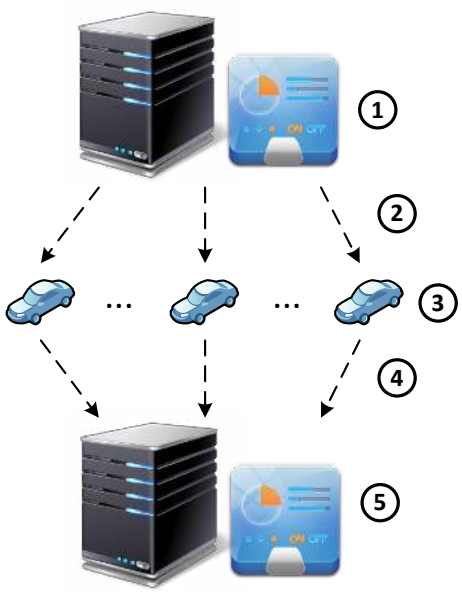


Fig. 3. Fitness evaluation process.

3) Tabu List: We construct a tabu list τ_r for each request r to reduce the size of the search space. τ_r contains those vehicles k which cannot serve r. As implied by Theorem 1, if a request r is not admissible by k, any set of requests containing r will also not be admissible by k. In other words, we will never need to consider those k in τ_r when configuring κ_r. Unlike Tabu Search [41], we do not need to update the tabu lists during the course of search.[3] τ_r's are only constructed in the initialization phase of the algorithm and utilized in both initial population generation and mutation.

4) Selection: In each generation, a fraction X_{rate} of N_{pop} survives and the rest of $(1 - X_{rate})$ will be replaced by the children bled in the processes of crossover. We apply weighted random pairing [42] to select the survived chromosomes to perform crossover.

5) Crossover: Crossover is an operator in GA to achieve intensification. In each operation, it manipulates two parent chromosomes to breed two offspring. The offspring inherit the merits from their parents and thus they tend to have better fitness values, i.e., higher objective function values of (19a). By Theorem 2, a larger set of requests will improve the fitness. Also based on Statement 3 of Theorem 1, we manipulate the chromosomes with crossover as follows. Parents i and j reproduce offspring i' and j'. i' admits all those r's as i does with the same set of vehicles. If there is any k which is adopted in j but not in i, we randomly adopt one such k

[3]As discussed in Section III-B, admission control is completed in the duty assignment sub-interval once in each operating interval. Such sub-interval is short so that it is unlikely to have great changes to the positions of the AVs. Thus the tabu lists can be assumed to be static throughout the admission control process happened in each interval. However, the tabu lists may need to be updated in the next interval as the vehicles may have moved to other positions.

in i on those r's which are not admitted in its parent i. We produce an offspring j' dominantly inherited by the parent j similarly. In this way, the offspring are likely to admit more requests resulting in higher fitness.

6) Mutation: Mutation exhibits diversification to prevent the algorithm from getting stuck in local optimums and we basically follow [42] to design mutation. We control the amount of mutation with a mutation rate $\mu \in [0, 1]$. We apply elitism to the chromosome with highest fitness in the population and only the rest undergo mutation. A mutation occurs on bit z_r^i of chromosome i and the number of mutations taken place in each generation is $\mu \times (N_{pop} - 1) \times |\mathcal{R}|$. If we perform mutation on z_r^i, we toggle z_r^i. If z_r^i is changed from 0 to 1, we randomly assign κ_r a k which is not in the tabu list τ_r. If z_r^i is changed from 1 to 0, we set $\kappa_r = 0$. To further enhance diversification, besides the elite chromosome, each chromosome has a probability of γ to be replaced by a random chromosome.

D. Algorithmic Design

We basically follow [42] to design the algorithm, which consists of three stages: initialization, iterations, and the final stage. The flow chart of the algorithm is given in Fig. 4. We maintain the chromosomes with feasible candidate solutions during the whole course of search.

1) Initialization: In initiation, we define all the system parameters, e.g., N_{pop} and X_{rate}, and construct the tabu list τ_r for each r. Then we create the initial population of chromosomes, each of which is assigned with one random request r associated with a vehicle not in its tabu list τ_r. This can ensure all chromosomes are initially feasible. We evaluate the fitness of the initial chromosomes before the iterations start.

2) Iterations: In each iteration (or called generation), we manipulate the candidate solutions held by the chromosomes. Before any modification, we back up the feasible candidate solutions stemmed from the previous generation. Then we perform selection, crossover, and mutation to manipulate the chromosomes, followed by fitness evaluations. If any chromosome possesses an infeasible solution, we retain its original feasible one from the backup. We check the stopping criteria to see if we continue with the next iteration or proceed to the final stage. One commonly used stopping criterion is termination after undergoing a certain number of generations.

3) Final Stage: We output the best solution found in this stage.

In general, the solution method is implemented in a central manner at the control center. When evaluating the fitness of the chromosomes, the scheduling tasks are distributed to the vehicles based on distributed scheduling.

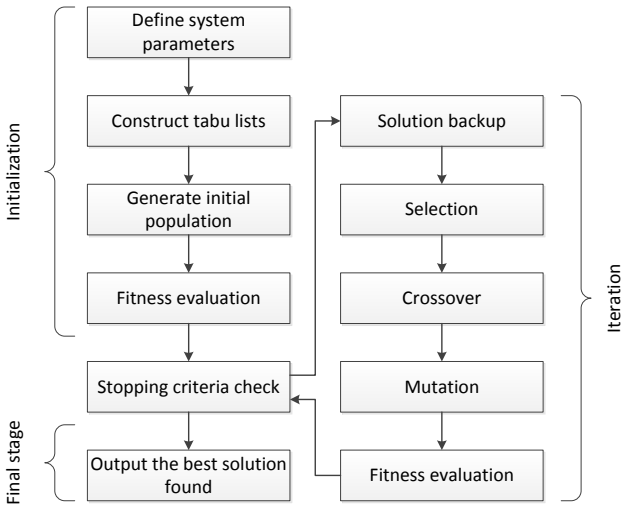


Fig. 4. Flow chart of the algorithm.

VII. Performance Evaluation

We perform a series of simulations to evaluate different aspects of the algorithm. We consider a set of real taxi service data from [43], containing the pickup and dropoff times, and pickup and dropoff locations of a number of taxi trips served in the City of Boston. We sample 100 trip data whose pickup times happened within a period of 30 minutes in a day of 2012 as the transportation request pool. Since no existing transport can offer flexible shared-ride services as our system does, we adopt the data for our system as follows: the earliest service starting time as the pickup time of the data, the latest service starting time as the pickup time plus 15 minutes, the maximum ride time as the actual trip time times 1.5, random seat occupancy in the range of $[1, 5]$, and 50% of the actual taxi fare as the charges. The driving distance and travel time between any two locations are determined through the Google Maps API. Based on [44], we assume that the fuel cost is 16 cents per mile. We select five gas stations in Boston as the refuel stations for AVs. Each vehicle is assumed to be equipped with five seats and we randomly place the vehicles in the city.

We perform the simulations on a computer with Intel Core i7-2600 CPU at 3.40 GHz and 32 GB of RAM. They are conducted in the MATLAB environment, where the scheduling problem is addressed with YALMIP [45] and CPLEX [46]. We follow [42] to set the GA parameters: $N_{pop} = 16$, $X_{rate} = 0.5$, and $\mu = 0.15$, and we set $\gamma = 0.5$. Recall that, to operate the system for a period of time, we need to do admission control for each operating interval within the period. To perform admission control for an interval, we need to undergo a number of scheduling processes. We try to evaluate the performance of the algorithm incrementally from the smallest module. First we evaluate the computation time for scheduling. In the second test, we evaluate the performance of the algorithm on solving the admission control problem. At last, we examine the profits made when the system operates continuously for a period of time.

A. Computation Time for Scheduling

As Problem 1 is an MILP, we assume that CPLEX can return the optimal solution if the problem is tractable. So we focus on the computation time. When we look at Problem 1, the numbers of variables and constraints grow exponentially with the problem size in terms of the quantities of transportation

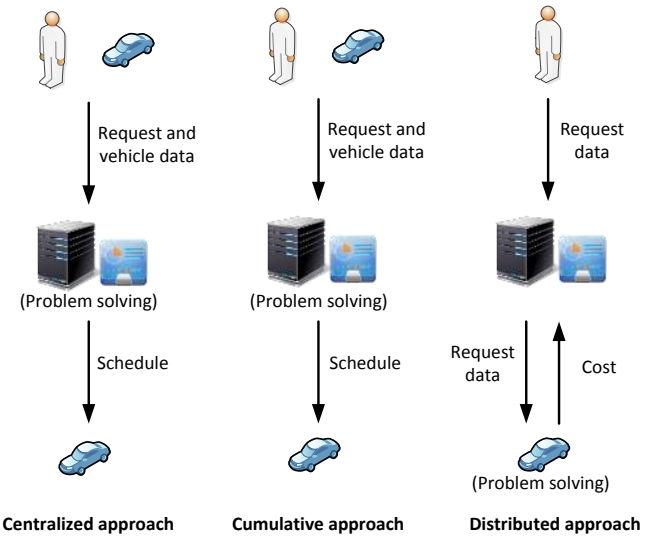
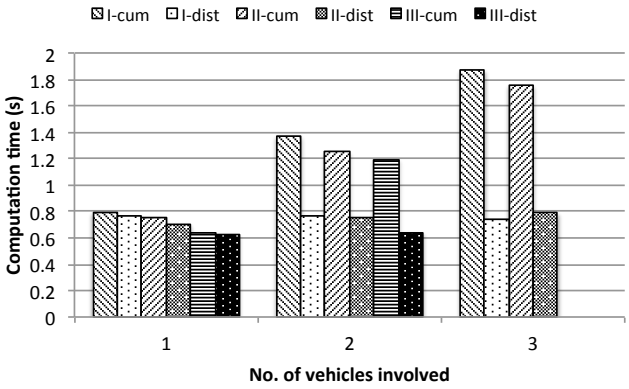
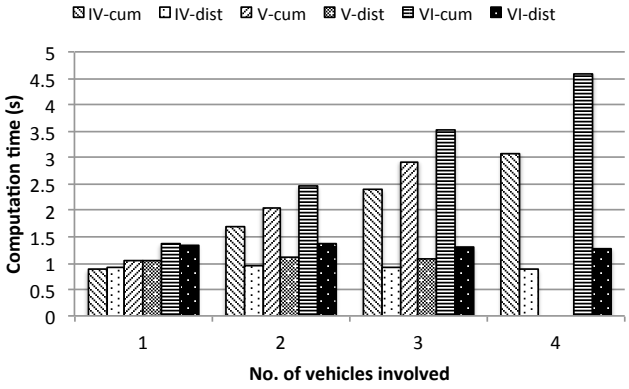


Fig. 5. Data processing, communications, and computation of the three approaches in scheduling.

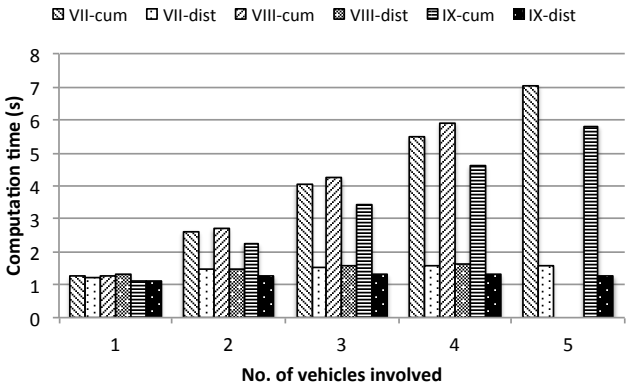
requests and vehicles. Hence the computation time for scheduling grows very fast with the problem size. For demonstrative purposes, we focus on small problem instances. We randomly generate 9 cases from the Boston dataset: three cases with three requests, three with four requests, and three with five requests. All the cases are served with five vehicles. Recall that we have two main ways to address the scheduling problem: (1) by solving Problem 1 as a whole and (2) by solving a number of Problem 3 collectively. For the latter, we can further arrange the subproblems to be solved (2.1) *en masse* at the control center or (2.2) separately at the individual vehicles. Thus, there are three approaches in total and we call (1), (2.1), and (2.2) the centralized, cumulative, and distributed approaches, respectively. The data processing, communications, and computation of the three approaches are depicted in Fig. 5. For the centralized and cumulative approaches, all data need to be collected and gathered at the control center from the passengers and vehicles for processing. After scheduling, the computed schedules will be distributed to the corresponding vehicles. For the distributed approach, the vehicular data are only maintained at the problem solving agents, i.e., that vehicles *per se*, before and after the corresponding subproblems being solved. After scheduling, the resulting costs are transmitted back to the control center for the subsequent scheduling. When different numbers of vehicles are involved, the computation time can be noticeably different. To see this, for each of Cases I-IX, we examine all possible combinations of z and κ (i.e., candidate solutions for chromosomes) and check their computation times for scheduling. We consider the time spent on communications negligible as it is usually much smaller when compared with the computation time. Fig. 6 shows the average computation times for feasible schedules with different numbers of vehicles involved in each case. Since the computation time of the centralized approach grows too fast (e.g., 8.30 s, 69.25 s, and 6.72×10^3 s for 3–5 requests, respectively), the time



(a) 3 requests



(b) 4 requests



(c) 5 requests

Fig. 6. Computation times for scheduling.

changes for the cumulative and distributed approaches would have become indistinguishable if the centralized data had also been displayed. For clearer representation, we skip the results for the centralized approach in Fig. 6. In Fig. 6, some bars

are missing because no feasible schedule can be computed with particular numbers of vehicles involved. For example, one request in Case III cannot be scheduled with any vehicle, and thus, no results are shown for three vehicles for Case III. Generally, for the cumulative approach, the computation time grows linearly with the number of vehicles involved as more subproblems with similar size need to be solved. For the distributed approach, the computation times with different vehicle sizes are more or less similar because the involved subproblems can be handled at different vehicles simultaneously. While the computation time of the centralized approach grows exponentially with the number of requests, that of the cumulative approach increases at a much slower rate and that of the distributed approach is approximately steady. Hence, it is not feasible to adopt the centralized approach. If the vehicles have sufficient communication and computation capabilities, we suggest the distributed approach. Otherwise, we can only endorse the cumulative approach for scheduling.

B. Admission Control in an Operating Interval

Next we investigate the performance of the algorithm to address admission control for an operating interval. Each fitness evaluation involves solving the scheduling problem once and the computation time for each fitness evaluation is dominated by that for scheduling. Moreover, the computation time of the algorithm depends on the number of fitness evaluations needed. Since the population size is fixed in every generation, the run time of the algorithm can be estimated from the number of generations taken place and the results determined in Section VII-A. Hence here we focus on the solution quality instead.

We run the algorithm for Cases I-IX. As we have examined all candidate solutions, we can acquire the optimal solutions of these cases. We repeat running the algorithm 20 times for each case. Fig. 7 shows the average objective function value computed during the course of search for 40 generations. As absolute values do not help reveal the performance of the algorithm, the objective function values are instead normalized with the corresponding optimal values to standardize the presentation.[4] For each data point, we also provide the error bars for the maximum and minimum values computed in the 20 repeats. The performance of the algorithm in each case is similar. The algorithm starts with relatively low quality solutions and then converges rapidly to the global optimal in a few generations. The gap between the error bars diminishes after more generations have been taken place and this further confirms the convergence of the algorithm. When the problem size increases, it takes slightly more generations to have the algorithm converged. We can conclude that our algorithm is very effective in solving the admission control problem.

We further investigate the total profits gained for the test cases with different AV population sizes. We perform the simulations with the same settings and repeat each test 20 times. Fig. 8 shows the average results with respect to 5, 10, 15, and 20 vehicles. Since the resultant profit highly depends on the parameters of the respective requests and vehicles, the total profits gained from different cases are not directly comparable. Instead for each case, we show the percentage change of profit by normalizing the results with the profit made with 5 AVs. Since all cases show similar trends, for clearer presentation, we give the results for Cases I, IV, and VII in Fig. 8 only. In general, the more vehicles available, the higher profit can be made. However, the increase of profit is marginal; when compared with 5 AVs, the increase is just $1-2\%$ in the presence of 20 AVs. The reason is that more available vehicles may result in more economical routes but the total distance travelled would not be shortened significantly. Fig. 9 shows the average computation times required to perform admission control corresponding to the cases given in Fig. 8.[5] The more vehicles or requests, the longer the computation is.

C. Admission Control in Consecutive Operating Intervals

Here we consider operating the system consecutively for a period of time to entertain the 100 requests in the transportation request pool. We consider two cases of different operating interval durations. In Case 1, there are 10 intervals, in each of which 10 random requests from the pool are to be scheduled. If a request is successfully admitted in an interval, it will be eliminated from the pool. Otherwise, it will be considered again in the subsequent intervals. The setting for Case 2 is similar but we consider total 20 intervals with 5 requests being processed in each interval. Five vehicles are arranged to serve the requests in both cases and we apply our algorithm to each interval for admission control. In other words, we perform 10 and 20 admission control processes in Cases 1 and 2, respectively.

Fig. 10 shows the profit accumulated along the intervals, in which we consider the duration of one interval for Case 1 is that of two intervals for Case 2. Note that the cost is the actual expense on gas based on the traversed distance and the revenue gained from serving each request is the discounted result of having 50% off from the real fare as if the request would be served by a normal taxi in Boston. The discount is used to compensate for the inconvenience of ride sharing and possibly longer ride time. This discount rate may be already attractive to many people to adopt our system instead of the normal taxi service. Hence the profit shown can be projected to a real business running in a similar scale. Fig. 11 provides the numbers of successfully admitted requests along the same interval horizon as in Fig. 10. We can see that Case 2 can produce more profit by successfully admitting more transportation requests. With the same number of vehicles in service, the smaller the number of requests to be scheduled in an interval, the higher the success rate of admission control is. In real situation, we normally cannot dramatically increase the size of the AV fleet and we would not intentionally reduce the number of AVs in service. On the other hand, it is much easier to adjust the number of requests to be scheduled each time by controlling the duration of each operating interval. In general, the shorter the interval, the smaller number of requests there

[4] An optimal solution has the normalized objective function value equal to one.

[5] Note that the simulation is performed in an ordinary computer. In practice, a more powerful computer will be utilized and the computation will be much shorter, especially when parallel computation is employed.

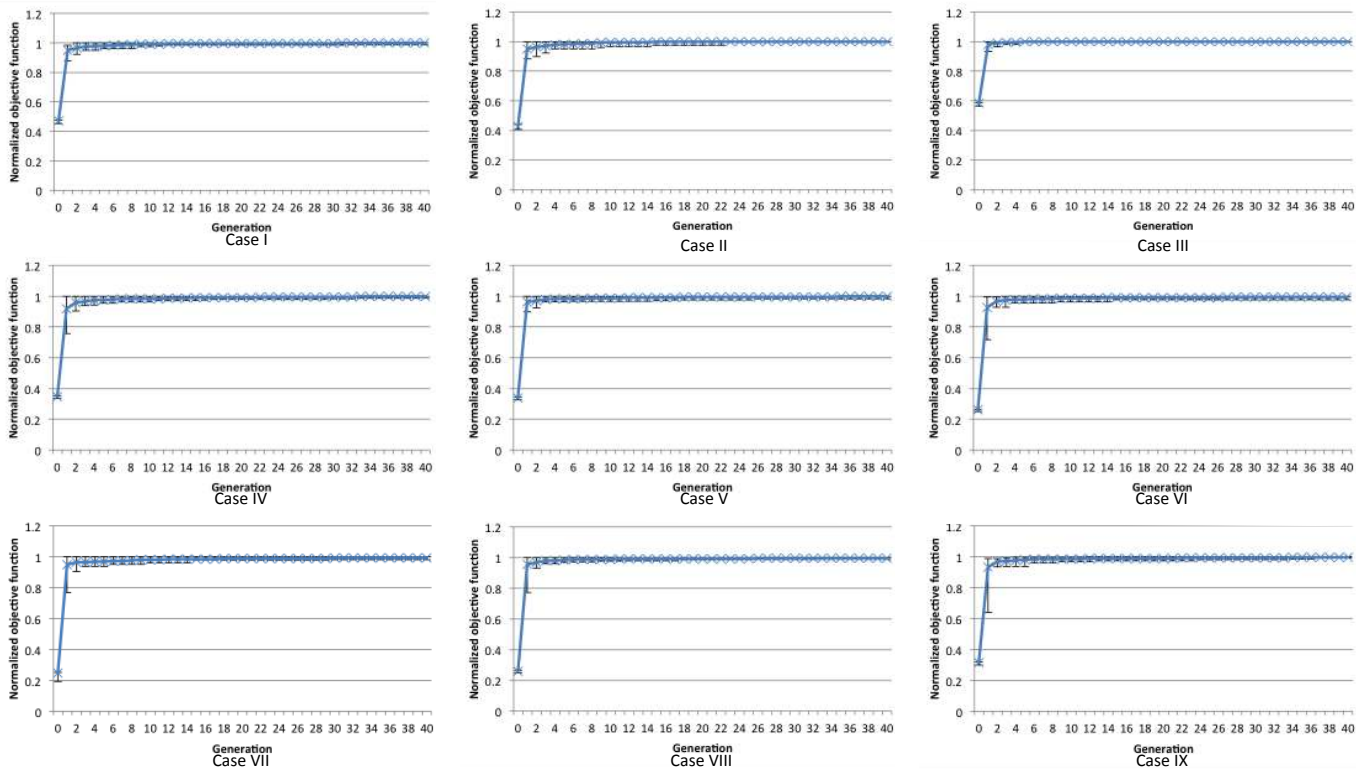


Fig. 7. Evolutions of the algorithm in solving admission control.

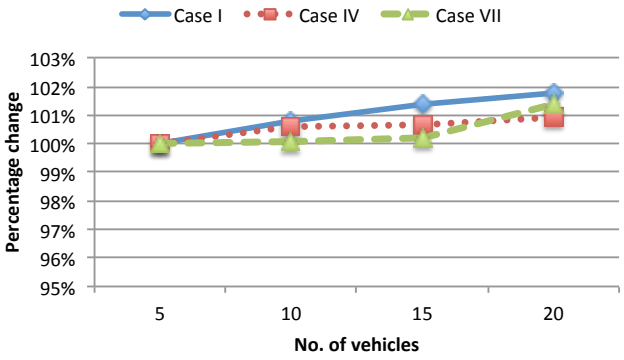


Fig. 8. Profits made with different numbers of vehicles.

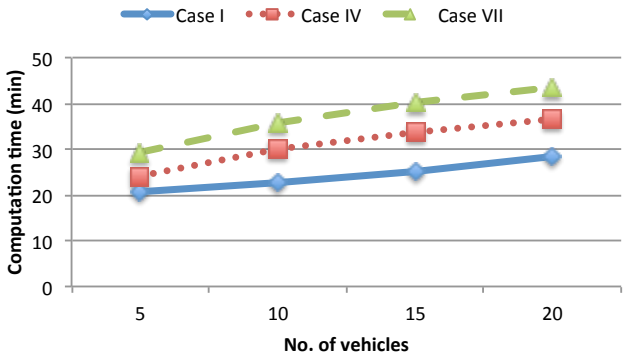


Fig. 9. Computation time for admission control with different numbers of vehicles.

are. Therefore, we would suggest to set the operating interval shorter, resulting in fewer requests to be scheduled each time and higher profits. Moreover, this will make the scheduling problem smaller by requiring shorter computation time to run the algorithm.

VIII. Conclusion

With advancements in technologies, AVs become feasible and can run on the roads. Various vehicular wireless communication technologies allow AVs to be connected and respond cooperatively to instantaneous situations. This constitutes a new form of public transport with high efficiency and flexibility. In this paper, we propose the AV public transportation system supporting point-to-point services with ride sharing capability. The system manages a fleet of AVs and accommodates a number of transportation requests. We focus on two major problems in the system: scheduling and admission control. The former is to configure the most economical schedules and routes for the AVs in order to satisfy the admissible requests. The latter is to determine the set of admissible requests among all requests so as to produce maximum profit. We formulate the scheduling problem as an MILP. The admission control problem is cast as a bilevel optimization problem, in which the scheduling problem is set as a constraint. We propose

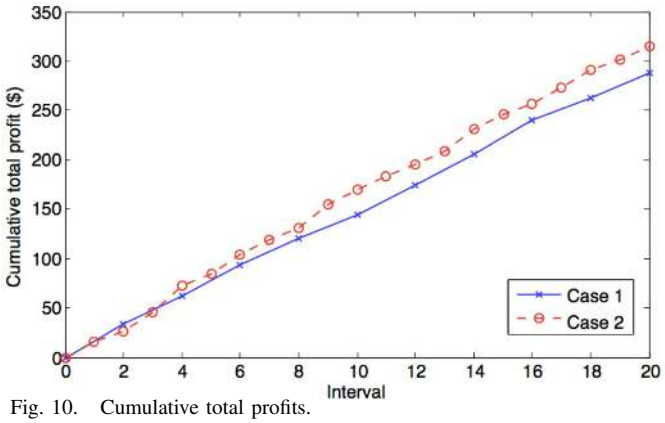


Fig. 10. Cumulative total profits.

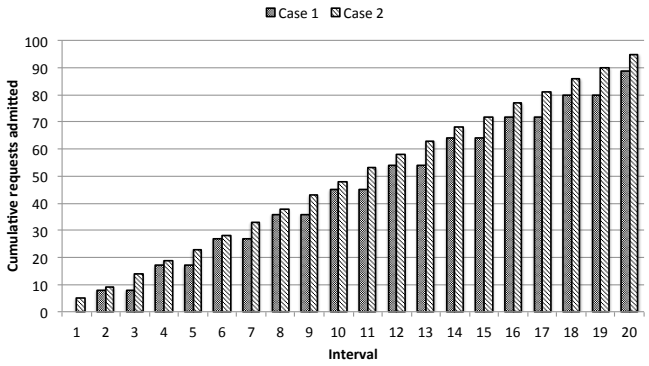


Fig. 11. Cumulative numbers of admitted requests.

a GA-based solution method to address admission control. We perform a series of simulations with a real taxi service dataset recorded in Boston and the simulation results show that our solution method is effective in solving the problem. By shortening the operating intervals, the system can curtail the computation time required to solve the problem by limiting the quantity of the submitted requests and it can also produce higher profit cumulatively. To summarize, our contributions in this paper include: (i) designing the AV public transportation system, (ii) formulating the scheduling problem, (iii) developing distributed scheduling, (iv) formulating the admission control problem, (v) introducing the concept of admissibility and deriving the related analytical results, (vi) proposing an effective method to solve the admission control problem, and (vii) validating the performance of the solution method with real-world transportation service data.

ACKNOWLEDGMENT

Chu was supported by Hong Kong GRF grant HKBU 210412.

REFERENCES

[1] A. Y. S. Lam, Y.-W. Leung, and X. Chu, "Autonomous vehicle public transportation system," in *Proc. the 3rd Int. Conf. on Connected Veh. and Expo*, Vienna, Austria, Nov. 2014.

[2] DARPA. (2007) Urban challenge. [Online]. Available: http://archive.darpa.mil/grandchallenge/

[3] M. Bertozzi, A. Broggi, A. Coati, and R. U. Fedriga, "A 13,000 km intercontinental trip with driverless vehicles: The VIAC experiment," *IEEE Intell. Transp. Syst. Mag.*, vol. 5, no. 1, pp. 28–41, 2013.

[4] Wepedia. (2014) Google driverless car. [Online]. Available: http://en.wikipedia.org/wiki/Google_driverless_car

[5] B. W. Smith. (2013, Jan.) Automated driving: Legislative and regulatory action. [Online]. Available: http://cyberlaw.stanford.edu/wiki/index.php/Automated_Driving:_Legislative_and_Regulatory_Action

[6] I. Technology. (2014) Navia. [Online]. Available: http://induct-technology.com/en/products/navia

[7] M. Online. (2013, Sep.) It really is hands free! self-driving mercedes-benz is unveiled - and it should be available within seven years. [Online]. Available: http://www.dailymail.co.uk/sciencetech/article-2418526/Self-driving-Mercedes-Benz-sale-2020-unveiled.html

[8] D. News. (2014, Mar.) Bmw, audi push self-driving cars closer to reality. [Online]. Available: http://www.nydailynews.com/autos/making-self-driving-cars-everyday-reality-article-1.1251048

[9] M. N. Mladenovic and M. M. Abbas, "Self-organizing control framework for driverless vehicles," in *Proc. 16th Int. IEEE Conf. on Intell. Transp. Syst.*, The Hague, The Netherlands, Oct. 2013.

[10] L. K. J. Hu, W. Shu, and M.-Y. Wu, "Scheduling of connected autonomous vehicles on highway lanes," in *Proc. IEEE Global Comm. Conf.*, Anaheim, CA, Dec. 2012.

[11] P. Petrov and F. Nashashibi, "Modeling and nonlinear adaptive control for autonomous vehicle overtaking," *IEEE Trans. Intell. Transp. Syst.*, p. in press, 2014.

[12] Q. Li, L. Chen, M. Li, S. L. Shaw, and A. Nuchter, "A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios," *IEEE Trans. Veh. Technol.*, vol. 63, no. 2, pp. 540–555, 2014.

[13] D. N. Cottingham, "Vehicular wireless communication," University of Cambridge, Tech. Rep. UCAM-CL-TR-741, Jan. 2009.

[14] A. Dahiya and R. K. Chauhan, "A comparative study of MANET and VANET environment," *J. of Comput.*, vol. 2, no. 7, pp. 87–92, Jul. 2010.

[15] A. Furda, L. Bouraoui, M. Parent, and L. Vlacic, "Improving safety for driverless city vehicles: Real-time communication and decision making," in *Proc. IEEE 71st Veh. Technol. Conf.*, Taipei, Taiwan, May 2010.

[16] M. Alsabaan, K. Naik, and T. Khalifa, "Optimization of fuel cost and emissions using v2v communications," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1449–1461, Sep. 2013.

[17] P. Gomes, C. Olaverri-Monreal, and M. Ferreira, "Making vehicles transparent through v2v video streaming," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 930–938, Jun. 2012.

[18] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti, "Quantifying the benefits of vehicle pooling with shareability networks," *Proceedings of the National Academy of Sciences*, 2014, in press.

[19] S. Ma, Y. Zheng, and O. Wolfson, "T-share: A large-scale dynamic taxi ridesharing service," in *Proc. IEEE 29th International Conference Data Engineering*, Brisbane, Australia, Apr. 2013, pp. 410–421.

[20] A. Y. S. Lam, "Combinatorial auction-based pricing for multi-tenant autonomous vehicle public transportation system," *IEEE Trans. Intell. Transp. Syst.*, 2015, in press.

[21] G. R. Mauri, L. Antonio, and N. Lorena, "Customers' satisfaction in a dial-a-ride problem," *IEEE Intell. Transp. Syst. Mag.*, vol. 1, no. 3, pp. 6–14, Fall 2009.

[22] D. Zheng, W. Ge, and J. Zhang, "Distributed opportunistic scheduling for ad hoc networks with random access: An optimal stopping approach," *IEEE Trans. Inf. Theory*, vol. 55, no. 1, pp. 205–222, Jan. 2009.

[23] W. Ge, J. Zhang, J. E. Wieselthier, and X. Shen, "Phy-aware distributed scheduling for ad hoc communications with physical interference

model," *IEEE Trans. Wireless Commun.*, vol. 8, no. 5, pp. 2682–2693, May 2009.

[24] E. Z. Tragos, G. Tsiropoulos, G. T. Karetsos, and S. A. Kyriazakos, "Admission control for qos support in heterogeneous 4g wireless networks," *IEEE Netw.*, vol. 22, no. 3, pp. 30–37, May–Jun. 2008.

[25] S. Wright, "Admission control in multi-service IP networks: a tutorial," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 2, pp. 72–87, 2nd Quarter 2007.

[26] M. R. Sherif, I. W. Habib, M. Nagshineh, and P. Kermani, "Adaptive allocation of resources and call admission control for wireless atm using genetic algorithms," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 2, pp. 268–282, Feb. 2000.

[27] B. Rong, Y. Qian, K. Lu, R. Q. Hu, and M. Kadoch, "Mobile-agent-based handoff in wireless mesh networks: Architecture and call admission control," *IEEE Trans. Veh. Technol.*, vol. 58, no. 8, pp. 4565–4575, Oct. 2009.

[28] A. Y. S. Lam, Y.-W. Leung, and X. Chu, "Electric vehicle charging station placement: Formulation, complexity, and solutions," *IEEE Trans. Smart Grid*, vol. 5, no. 6, pp. 2846–2856, Nov. 2014.

[29] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.

[30] O. Ben-Ayed and C. E. Blair, "Computational difficulties of bilevel linear programming," *Oper. Res.*, vol. 38, no. 3, pp. 556–560, May/Jun. 1990.

[31] S. Dempe, *Foundations of Bilevel Programming*. Boston: Kluwer Academic, 1993.

[32] Y. Yin, "Genetic-algorithms-based approach for bilevel programming models," *J. Transp. Eng.*, vol. 126, no. 2, pp. 115–120, Mar./Apr. 2000.

[33] J. F. Bard, *Practical Bilevel Optimization: Algorithms and Applications*. Dordrecht: Kluwer Academic Publishers, 1998.

[34] A. Koh, "Solving transportation bi-level programs with differential evolution," in *Proc. IEEE Congress on Evolutionary Computation*, Singapore, Sept. 2007.

[35] M. Mesbah, M. Sarvi, and G. Currie, "Optimization of transit priority in the transportation network using a genetic algorithm," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 3, pp. 908–919, Sept. 2011.

[36] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[37] K. Price, R. M. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. New York: Springer, 2005.

[38] A. Y. S. Lam and V. O. K. Li, "Chemical-reaction-inspired metaheuristic for optimization," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 381–399, Jun. 2010.

[39] D. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, NJ: Prentice Hall, 1989.

[40] Y. Shoham, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge, NY: Cambridge University Press, 2009.

[41] F. Glover and M. Laguna, *Tabu Search*. Boston: Kluwer Academic Publishers, 1997.

[42] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*, 2nd ed. New York: Wiley Interscience, 2004.

[43] C. Savoie. (2014, Aug.) Boston taxi data. [Online]. Available: https://data.cityofboston.gov/Transportation/Boston-Taxi-Data/ypqb-henq

[44] U.S. Department of Energy. (2014, Oct.) Comparing energy costs per mile for electric and gasoline-fueled vehicles. [Online]. Available: http://avt.inel.gov/pdf/fsev/costs.pdf

[45] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *Proceedings of IEEE Int. Symposium on Comput. Aided Control Syst. Design*, Taipei, Taiwan, Sep. 2004.

[46] *IBM ILOG CPLEX V12.1 User's Manual for CPLEX*, 2009.