Real-time Ridesharing for Transportation Hubs with Demand and Supply Uncertainty

Zengxiang Lei
Lyles School of Civil Engineering, Purdue University
550 Stadium Mall Dr, West Lafayette, IN, 47907
lei67@purdue.edu

Xinwu Qian, Ph.D.
Lyles School of Civil Engineering, Purdue University
550 Stadium Mall Dr, West Lafayette, IN, 47907
qian39@purdue.edu

Xiaowei Chen
Lyles School of Civil Engineering, Purdue University
550 Stadium Mall Dr, West Lafayette, IN, 47907
chen3379@purdue.edu

Satish V. Ukkusuri, Ph.D.
Lyles School of Civil Engineering, Purdue University
550 Stadium Mall Dr, West Lafayette, IN, 47907
sukkusur@purdue.edu
(Corresponding Author)

Word Count: 5455 words + 6 figures$\times$ 0 + 1 table$\times$ 250 = 5705 words

Submission Date: July 31, 2020

ABSTRACT

Transportation hubs in major cities generate a significant amount of trips by taxis and for-hail vehicles (FHV), with many of the trips sharing similar destinations. This suggests promising opportunities to leverage the collective travel needs with dedicated ridesharing solutions to reduce the externalities of excessive traffic at transportation hubs. In this study, we develop a novel dynamic ridesharing approach to serve trips from the transportation hub by considering (1) demand (new passengers) and supply (newly available vehicles) in the near future and (2) the uncertainty of future predictions. Our approach consists of two stages. In the first stage, we develop a data structure called hub mobility tree to generate potential combinations of shareable trips as candidate schedules efficiently. Then the generated schedules are used in the second stage to formulate the stochastic hub-based ridesharing problem (SHRP), which is a stochastic integer programming problem with the objective to maximize the total expected ridesharing profit over time. Due to the prohibitive number of shareable trips, we then approximately solve SHRP by the sample average approximate method (SAA), and a dual Lagrangian technique is implemented to further improve the scalability of the solution approach. We demonstrate the performances of the proposed method by simulating the ridesharing service at JFK airport using NYC taxi and FHV data. The results indicate that the proposed method outperforms the myopic ridesharing (maximize profit for a single time step) and the rolling horizon method with point estimation of future demand and supply.

Keywords:Dynamic ridesharing; Transportation hubs; Rolling horizon; Stochastic integer program

INTRODUCTION

Several inefficiencies in urban transportation systems exist, including the low usage rate of public transit modes, the excessive private vehicle ownership and the relatively low vehicle occupancy (*1*). These lead to wasted vehicle capacity, more substantial congestion and emissions, and unnecessary energy consumption (*2, 3*). Among different places in urban space, transportation hubs such as railway terminals and airports suffer most from these inefficiencies (*4*), since they bring together many commuters in short periods and fail to leverage the collective behaviors to efficiently serve people with similar itineraries and time schedules. According to the NYC 2018 TLC factbook, the airports (JFK and LGA) account for more than 30,000 daily pickups of taxis and FHVs, while having the lowest level (0-10%) of the share of shared rides among all taxi zones (*5*). With the number of visitors increase in transportation hubs and so as the emissions and other negative effects, it is important to develop a solution to reduce those inefficiencies.

One of the promising solutions to achieve this goal is to promote ridesharing service (*6*) at the transportation hub, which can significantly increase vehicle occupancy (*7*) and therefore reduce the vehicle mileage and alleviate the traffic congestion surrounding the hub. Here we specifically consider the ridesharing service provided by a ride-sourcing platform (like Uber or Lyft) as it is close to the real-world scenario. In this service, passengers at transportation hubs make trip requests in real-time, then the platform combines different requests into different schedules and assigns these schedules to available vehicles.

The effectiveness of ridesharing is governed by two factors. The first factor is the passengers' willingness to share their rides. And the second one is the quality of the vehicle dispatching and passenger assignment strategies. To increase the number of ridesharing users, current practice is to provide a discount for using ridesharing service (*8, 9*). Then with the ridesharing users are given, the assignment and dispatching algorithms will decide the schedules of vehicles to serve those passengers, and the platform can make profits by matching more passengers into single rides to save the payment to drivers. Note with more efficient ridesharing algorithms, additional profit can be made, and this profit can help to increase the discount level. This creates a positive feedback to attract more users and further promote the ridesharing service. Therefore, this study focuses on ridesharing algorithms for vehicle dispatching and passsenger assignments.

Developing a ridesharing algorithm that optimally assigns passengers to drivers has long been recognized as a challenging problem (*10*). Even with the whole set of drivers and passengers revealed, this problem is recognized as a variant of travel salesman problem with a time window, which is in NP-complete (*11*), let alone in real practice the exact future information is not available. Most of exist work (*11–17*) therefore developed algorithms to optimally assign ridesharing trips to serve passengers within the current time step. Because this type of algorithms only consider one time step, the long term optimal of ridesharing schedules cannot be guaranteed. Recent studies started to use future prediction of travel demand in the routing of ridesharing problem (*18, 19*) and management of vehicle fleets (*20–22*). Yet, few studies consider the predictions of demand and supply on solving ridesharing scheduling in the system-level. In addition, how to conduct ridesharing optimally in the face of uncertainty, where we do not have accurate point estimation of future demand and supply, has not been studied. These leave a gap for developing a ridesharing algorithm which considers the future (demand and supply) information with a probabilistic distribution for future supply and demand are available.

In addition to the gap, there are several distinct features of the passenger trips that can facilitate the ridesharing framework with future prediction information. First of all, trips from a

transportation hub share the same trip origin: the hub itself. Therefore, the complexity of finding shareable rides can be hugely reduced. Second, the arrival of passengers at transportation hubs depends highly on the timetable (of trains and flights) at the hubs, thereby leading to more predictable demand, which makes it more likely to have a reliable prediction model for optimizing ridesharing scheduling.

In this work, we develop a hub-based ridesharing approach to facilitate the ridesharing service within transportation hubs. We first design a data structure called hub mobility tree to facilitate the generation of candidate schedules of sharing rides. Based on the generated schedules, we formulate a stochastic optimization problem with rolling horizons which considers the demand (new passengers) and supply (newly available vehicles) in the near future and the stochastic nature of future predictions. To solve this problem, we adopt the framework of the sample average approximate method (SAA) and solve the Lagrangian dual of sampled scenarios to improve the scalability. Finally, we valid the performances of the proposed approach by simulating the ridesharing service in JFK airport using the real world taxi+FHV operational data in NYC, 2019.

The rest of the paper is structured as follows. In the next section, we formally define the hub-based ridesharing problem and propose the solution algorithm based on rolling-horizon and stochastic programming. In section three, we compared our method to the myopic benchmarks using New York City's (NYC) taxi and FHV data to demonstrate the effectiveness of our method. Finally, in section four, we summarize this work and discuss the next steps.

METHODOLOGY

In this section we design a real-time ridesharing mechanism for transportation hubs. For each time step t, we match the passengers and the available vehicles at the hub based on their information and the prediction of future demand and supply. For the sake of descriptive simplicity, we consider the case of one hub in the rest of this section.

Assumption

We make the following assumptions to reasonably simplify the problem.

1. The first assumption is on the choice behavior of ridesharing trips. For each person, we assume his/her trip deviation, measured by the difference between the travel time in shared ride and the travel time in the directed trip, is less than a threshold, otherwise the direct trip is preferred.
2. We consider the zonal-level ridesharing and assume the trip destinations are drawn from a finite set of zones. In real world, each zone can be a partition of the network or other divisions of urban space. This assumption allows us to consider finite combinations of rides.
3. We also assume the traffic condition is stable so we can pre-calculate the feasible combinations of the shared rides. This can be relaxed to the statement that the traffic condition is stable for a certain time period (e.g., one hour) and consider multiple time periods.
4. We consider the number of available vehicles is independent to the ridesharing model. This allows us to model the vehicle supply independently with the ridesharing assignments. However, in real world, as the ridesharing generally using less vehicles to serve the same level of demand, less vehicles will be attracted to the hub. This can be improved by using real-time vehicle information to predict the number of newly available vehicles so we can still get good estimation despite the ridesharing decisions.

5. Last but not the least, a short-term prediction model is assumed to be available for travel demand and vehicle supply at the hub. The output of this model will be the number of new passengers heading toward each zone, the number of newly available vehicles and the corresponding probability.

Definitions and Notations

This section describes some important concepts and variables in formatting the hub-based rideshare problem.

Definition 3.1 (Zone)**.** A zone z characterized by its zone ID is a spatial unit for service pickups and drop-offs. The travel time between zone i and zone j is represented as τ_{ij}, and the travel cost that corresponds to the optimization objective is denoted as f_{ij}. We assume the travel time satisfies triangular inequality, that is, $\tau_{ij} \leq \tau_{ik} + \tau_{kj}$.

Definition 3.2 (Schedule)**.** A schedule s consists of an ordered sequence of zone ID to indicate the service order. For example, $s_j = (0,1,1,2,3)$ means the vehicle picks up three passengers and then delivers them with the order $z_1 \to z_2 \to z_3$. The number of passengers heading to zone i served by schedules s_j is denoted as λ_{ij}. For this example, we have $\lambda_{1j} = 2$, $\lambda_{2j} = 1$. We denote the k-th element in s_j as s_j^k. The travel time from the hub to s_j^k as $\tau_{s_j^k}^{s_j} = \sum_{i=1}^{k-1} \tau_{s_j^i s_j^{i+1}}$, and the profit generated by this schedule as g_j. Here we consider the profit (saved cost) generated by this schedule as $c_j = \sum_{i=1}^{|s_j|-1} (f_{0 s_j^{i+1}} - f_{s_j^i s_j^{i+1}})$.

Definition 3.3 (Deviation threshold)**.** A deviation threshold for each passenger is defined as the maximal additional travel time that one can accept for switching from the direct trip to a shared ride, that is, $\tau_i^s \leq (1+\alpha)\tau_{0i}$. We call the $(\tau_i^s - \tau_{0i})/\tau_{0i}$ as the deviation factor. In practice, the deviation threshold α can be assumed to be constant or mined from historical orders, and can be extended to more inclusive functions based on passengers' behaviors.

Definition 3.4 (Prediction model)**.** Let d_t be the number of new passengers represented by a $n \times 1$ vector with the i-th element be the number of passengers heading to zone i and v_t be the number of new available vehicles from the beginning of step t to the start of time step $t+1$. A prediction model ξ generates the prediction of these information, that is, $\xi_t = (d_t, d_{t+1}, \ldots, d_{t+T_{pred}-1}, v_t, \ldots, v_{t+T_{pred}-1})$, and the corresponding probability p_{ξ_t}. We further assume the prediction results for different time steps are independent to each other.

The essential notations used in this paper are summarized in Table 1.

TABLE 1: Notations

Variables	Description
$\mathcal{Z}$	Set of zones for ridesharing service, $\mathcal{Z} = \{z_0, z_1, z_2, \ldots, z_n\}$, where z_0 is the hub
$\mathcal{S}$	Set of schedules, $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$, the k-th element of s_j is denoted as s_j^k
$\mathcal{T}$	Set of time steps, $\mathcal{T} = \{1, 2, \ldots, T\}$
a_t	Number of passengers at the start of time step t, represented by a $n \times 1$ vector
b_t	Number of available vehicles at the hub at the beginning of time step t
d_t	Number of new passenger within time step t
v_t	Number of new available vehicles in time step t
ξ_t	Prediction result for future demand and supply in time step t, $\xi_t = (d_t, d_{t+1}, \ldots, d_{t+T_{pred}-1}, v_t, v_{t+1}, \ldots, v_{t+T_{pred}-1})$
τ_{ij}	Travel time for traveling from zone i to zone j
$\tau^s_{s^k}$	Travel time to zone s^k in the schedule s, $\tau^s_{s^k} = \sum_{i=1}^{k-1} \tau_{s^i s^{i+1}}$
c	Profit generated by schedules, denoted as a $m \times 1$ vector $c = \{c_1, c_2, \ldots, c_m\}$
Λ	Incidence matrix between passengers and schedules. The i-th row, j-th column λ_{ij} denotes the number of passengers heading to zone i served by schedule s_j
α	The deviation threshold of the ridesharing trip
β	Compensation for additional waiting for one passenger per time step
γ	Capacity of each vehicle.

Problem definition

Before we dive into the dynamic version of the hub-based ridesharing problem (HRP), it is helpful to introduce the static version first. With the perfect information of new demand d_t and supply v_t at each time step t, the hub-based ridesharing problem is to find the schedules for all available vehicles to maximize the profit (saved cost) generated by this service while satisfying the vehicle capacity and deviation constraints. This can be formulated as the following integer linear programming (ILP) with a set of the decision variable x_t which is a vector with the length as the size of the schedule set $\mathcal{S}$. The j-th element of x_t represents the number of vehicles assigned to schedule j in time step t.

$$\text{(HRP)} \max_{x_1, x_2, \ldots, x_T} \sum_{t=1}^{T} [c^\intercal x_t + \beta \mathbf{1}^\intercal_{n\times 1}(\Lambda x_t - a_t)] \tag{1}$$

$$\text{s.t. } \Lambda x_t \leq a_t, t = 1, 2, \ldots, T \tag{2}$$

$$\mathbf{1}^\intercal_{n\times 1} x_t \leq b_t, t = 1, 2, \ldots, T \tag{3}$$

$$\tau_i^{s_j} \leq (1+\alpha)\tau_{0i},\ i = 1, 2, \ldots, n; j = 1, 2, \ldots, m \tag{4}$$

$$\sum_{i=1}^{n} \lambda_{ij} \leq \gamma,\ j = 1, 2, \ldots, m \tag{5}$$

$$a_{t+1} = a_t + d_t - \Lambda x_t, t = 1, 2, \ldots, T \tag{6}$$

$$b_{t+1} = b_t + v_t - \mathbf{1}^\intercal_{n\times 1} x_t, t = 1, 2, \ldots, T \tag{7}$$

$$x_t \in \mathbb{N}^m, t = 1, 2, \ldots, T \tag{8}$$

where $\mathbf{1}_{n\times 1}$ is the $n \times 1$ all-ones vector, $\mathbb{N}$ is the set of non-negative integers. The initial demand a_1 and supply b_1 are given. The objective of this ILP consists of two parts: $c^T x_t$ which is the profit (saved cost) generated by the ridesharing service, and $\beta \mathbf{1}^{\intercal}_{n\times 1}(\Lambda x_t - a_t)$ which denotes the compensation for additional waiting for passengers. Equation (2) ensures the number of served passengers does not exceed the actual demand. Equation (3) states one vehicle can only be assigned to one schedule. Equation (4) restricts the deviation factor for each trip does not exceed the deviation threshold. Equation (5) defines the capacity constraints. Finally, Equation (6) and (7) ensure the numbers of vehicles and passengers are consistent over time.

We recognize three fundamental difficulties in solving the HRP. The first one lies in the horizon T which can be very large (infinite if we consider the service is 24/7) in real world. As ILP is known as belonging to NP-complete, it is unlikely that we can get the solution for the full horizon. The second challenge comes from $\mathcal{S}$, the set of possible schedules. Given the number of zone n and the vehicle capacity γ, the number of all possible schedules is $\sum_{i=1}^{\gamma} n^i = \frac{n^{\gamma+1}}{n-1}$. Since the number of decision variables in each time step is equal to the number of candidate schedules, this also significantly increases the computational load for solving HRP. The last issue is the absence of the perfect information of future demand and supply in real practice.

The first challenge can be addressed heuristically with rolling horizon approach, and studies suggest that close-to-optimal solution can be achieved with the rolling horizon heuristic (*23*). Assume we are at time step t_0 and want to solve the HRP. Instead of maximizing $\sum_{t=t_0}^{T}[c^{\intercal}x_t + \beta \mathbf{1}^{\intercal}_{n\times 1}(\Lambda x_t - a_t)]$, we solve x_{t_0} under the same constraints but with the objective as

$$\max \sum_{t=t_0}^{t_0+T_{horizon}} [c^{\intercal}x_t + \beta \mathbf{1}^{\intercal}_{n\times 1}(\Lambda x_t - a_t)] \tag{9}$$

Although by no means the solution of (9) is equivalent to the solution of the full horizon HRP when $t_0 + T_{horizon} < T$, we may still obtain a better solution than doing optimization for the single time step. For the choice of $T_{horizon}$, note a passenger need to be served before waiting for $T_{profit} = \lfloor c_{\max}/\beta \rfloor$ steps in order to have positive benefit, where $c_{\max}$ is the maximum profit can be obtained among all potential schedules. One can choose $T_{horizon} > T_{profit}$ to fully exploit the potential for matching passengers within t_0 with future demand.

For the second issue, we propose three strategies to reduce the number of possible schedules $|\mathcal{S}|$. First note constraints (4) and (5) are independent to the x_t, so we can reduce the size of $|\mathcal{S}|$ by pre-calculating the schedules that satisfy (4) and (5). In addition, with the demand a_t in each time step t is given, we can draw the schedules that cover a_t. We further develop a data structure called hub-mobility tree to reduce the repeated computational efforts for generating the set of candidate schedules. Thirdly, as the number of candidate schedules can still be too large, a heuristic method is adopted to drop schedules with little profit. Thereby, we obtain the set of candidate schedules $\mathcal{S}_t$ for each time step t using the demand vector $a_{t_0} + \sum_{i=t_0}^{t-1} d_i$. We denote the size of $\mathcal{S}_t$ as m_t, the profit vector as c_t, and the corresponding incidence matrix as Λ_t.

Finally, to address the uncertainty of future demand and supply, we extend the rolling horizon HRP to a stochastic setting. we assume we have the prediction of future demand and supply for $T_{horizon}$ steps, denote as ξ_{t_0}. The stochastic hub-based ridesharing problem is then defined as follows.

$$\text{(SHRP)} \max_{x_{t_0}} c_{t_0}^{\intercal} x_{t_0} + \beta \mathbf{1}_{n\times 1}^{\intercal}(\Lambda_{t_0} x_{t_0} - a_{t_0}) + \mathbb{E}[Q(x_{t_0}, \xi_{t_0})] \tag{10}$$

$$\text{s.t. } \Lambda x_{t_0} \leq a_{t_0} \tag{11}$$

$$\mathbf{1}_{n\times 1}^{\intercal} x_{t_0} \leq b_{t_0} \tag{12}$$

$$x_{t_0} \in \mathbb{N}^{m_{t_0}} \tag{13}$$

where $\xi_{t_0} = (d_{t_0}, d_{t_0+1}, \ldots, d_{t_0+T_{horizon}-1}, v_{t_0}, v_{t_0+1}, \ldots, v_{t_0+T_{horizon}-1})$

$$Q(x_{t_0}, \xi_{t_0}) = \max_{x_{t_0+1},\ldots,x_{t_0+T_{horizon}}} \sum_{t=t_0+1}^{t_0+T_{horizon}} [c_t^{\intercal} x_t + \beta \mathbf{1}_{n\times 1}^{\intercal}(\Lambda_t x_t - a_t)] \tag{14}$$

$$\text{s.t. } \Lambda_t x_t \leq a_t, t = t_0+1, \ldots, t_0+T_{horizon} \tag{15}$$

$$\mathbf{1}_{n\times 1}^{\intercal} x_t \leq b_t, t = t_0+1, \ldots, t_0+T_{horizon} \tag{16}$$

$$a_{t+1} = a_t + d_t - \Lambda_t x_t, t = t_0, \ldots, t_0+T_{horizon}-1 \tag{17}$$

$$b_{t+1} = b_t + v_t - \mathbf{1}_{n\times 1}^{\intercal} x_t, t = t_0, \ldots, t_0+T_{horizon}-1 \tag{18}$$

$$x_t \in \mathbb{N}^{m_t}, t = t_0+1, \ldots, t_0+T_{horizon} \tag{19}$$

Formulation (SHRP) represents a two-stage stochastic integer program. In the first stage we want to find the x_{t_0}, the optimal assignment between vehicles and schedules at time step t_0, to maximize the expected total profit within $T_{horizon}$. In the second stage, the optimal benefit together with the corresponding path is solved for given x_{t_0} and ξ_{t_0} from the expectation of the recourse function $Q(x_{t_0}, \xi_{t_0})$. We then solve the SHRP by a sample average approximate (SAA) method, which derives the expected objective function of the stochastic problem by an average estimation from a random sample, and then solves sample average approximating problem by deterministic optimization techniques.

The rest of the section shows the details of the schedule generation and SAA method.

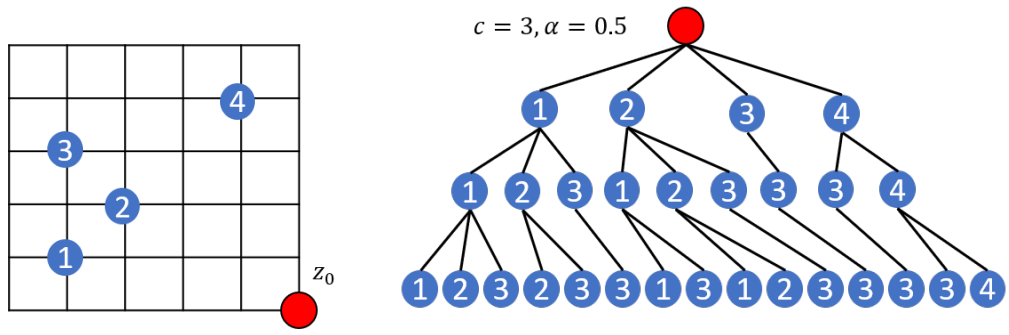
Schedule generation

Hub mobility tree

We first present a special data structure named hub mobility tree that can facilitate the searching of the possible schedules. The root of the tree is the hub, and the maximal depth is the maximal capacity of the vehicle. The first level of the tree involves the first delivery zone and the corresponding travel time from the hub, then the second level involves the second delivery zone and the corresponding travel time from, etc. Each branch keeps growing until it reaches the maximal depth or the deviation threshold is violated.

Figure 1 shows an toy example of the hub mobility tree. Consider in the current time step, we observe three from-hub requests r_1, r_2, r_3 with $z_{r_1}^d = z_1$, $z_{r_2}^d = z_2$, $z_{r_3}^d = z_3$. Then we perform a top down search to obtain all possible schedules are (z_0, z_1), (z_0, z_2), (z_0, z_3), (z_0, z_1, z_2), (z_0, z_1, z_3), (z_0, z_2, z_1), (z_0, z_2, z_3), (z_0, z_1, z_2, z_3), (z_0, z_2, z_1, z_3).

Although the complexity of building the hub mobility tree in the worst case is $O(n^{\gamma})$, real network is usually easy to deal with since the deviation factor is monotonically increasing as more nodes are inserted, so for each node we may reduce the searching scope significantly by iterating



(a) Grid network with 1 hub and 4 zones

(b) Hub mobility tree

FIGURE 1: Toy example for hub mobility tree

the corresponding zones of the nodes in the same level and with the same parent. In case the computational time is too long, we propose a heuristic algorithm which grows the hub mobility tree in a greedy manner based on the potential of saving travel cost measured by the saved cost and the deviation factor. In this algorithm, each node N is characterized by seven parameters: its level $N.level$, zone $N.zone$, the travel time to this node $N.time$, the saving cost to this node $N.save$, the parent node $N.parent$, the child nodes $N.child$ and the potential $N.potential$.

The algorithm is presented in Algorithm 1. We use a sorted array S to manage all nodes to search. Since the time complexity of insertion in such data structure is $O(log(n))$, each iteration for growing a new node is in $O(nlog(n))$, which is clearly tractable.

Schedule generation from the hub mobility tree

With the hub-based mobility tree, we extract the candidate schedules for the input demand vector by the following three steps. First, a top-down search is performed to obtain all candidate schedules that satisfy those demands. Second, we query duplicate schedules that cover the same group of passengers and keep the one with the highest saving cost. Finally, we trim the set of candidate routes by dropping the schedules with little saving which less than a given threshold C. The detailed algorithm is shown as follows.

The sample average approximation method

The sample average approximate method (SAA) is a general framework for solving stochastic programs, usually with the prohibitive number of scenarios. For SHRP, SAA proceeds by solving the following approximated SHRP (ASHRP) problem.

$$(\text{ASHRP}) \max_{x_{t_0}} c^\intercal x_{t_0} + \beta \mathbf{1}_{n\times 1}^\intercal (\Lambda_{t_0} x_{t_0} - a_{t_0}) + \frac{1}{M}\sum_{i=1}^{M} Q(x_{t_0}, \xi_{t_0}^i) \tag{20}$$

where the constraints are the same as in SHRP, and $\xi_{t_0}^1, \xi_{t_0}^2, \ldots, \xi_{t_0}^M$ are the predictions of possible future demand and supply information that are independently generated from the prediction model.

Although SAA is using $\frac{1}{M}\sum_{i=1}^{M}[Q(x_{t_0}, \xi_{t_0}^i)]$ to approximate the true expectation of the second stage optimal value $\mathbb{E}[Q(x_{t_0}, \xi_{t_0})]$, it has been shown in (*24*) that under mild conditions by solving the approximated problem one can obtain an optimal solution to the true problem with

Algorithm 1 Hub Mobility Tree Generation Algorithm

Input: $Z = (z_0, z_1, z_2, \ldots, z_n)$ service zones, where z_0 is the hub, the function of travel time t, the function of travel cost f, the deviation threshold α, the vehicle capacity γ.

Output: The root R of the hub mobility tree.

$R \leftarrow$ new Node.
$R.level \leftarrow 0$, $R.zone \leftarrow z_0$, $R.time \leftarrow 0$, $R.save \leftarrow 0$, $R.parent \leftarrow -1$
A sorted array of nodes to search $S \leftarrow$ empty set
for i=1,2,...,n **do**
    $N \leftarrow$ new Node.
    $N.level \leftarrow 1$, $N.zone \leftarrow z_i$, $N.time \leftarrow t(z_0, z_i)$, $N.save \leftarrow 0$, $N.parent \leftarrow R$, $N.potential \leftarrow \infty$
    Add N to $R.child$
    Add N to S
end for
while S is not empty and not reach the maximal running time **do**
    Select the node $M \in S$ with the largest potential
    Set of nodes $S_{new} = GROW(M)$
    for $N \in S_{new}$ **do**
        Insert N into S based on $N.potential$,
    end for
end while
return R

function GROW(M)
    $S_{add} \leftarrow$ empty set
    for $N_{candidate} \in M.parent.child$ **do**
        $z_{candidate} \leftarrow N_{candidate}.zone$
        if $z_{candidate} != M.parent.parent.zone$ or $z_{candidate} == M.parent.zone$ **then**
            if $M.time + t(M.zone, z_{candidate}) < (1+\alpha) t(z_0, z_{candidate})$ **then**
                $N \leftarrow$ new Node.
                $N.level \leftarrow M.level + 1$, $N.zone \leftarrow z_{candidate}$, $N.time \leftarrow M.time + t(M.zone, z_{candidate})$, $N.save \leftarrow M.save + f(z_0, z_{candidate}) - f(M.zone, z_{candidate})$, $N.parent \leftarrow M$
                if $(N.time - t(z_0, N.zone) - 1) \neq 0$ **then**
                    $N.potential \leftarrow N.save/(N.time - t(z_0, N.zone) - 1)$
                else
                    $N.potential \leftarrow \infty$
                end if
                Add N to $M.child$
                if $N.level < \gamma$ **then**
                    Add N to S_{add}
                end if
            end if
        end if
    end for
return S_{add}
end function

Algorithm 2 Schedule Generation Algorithm

Input: The root R of the hub mobility tree, demand vector a, and the minimum saving C

Output: The set of candidate schedules $\mathcal{S}$, profit vector c, incidence matrix between passengers and schedules Λ

$\mathcal{S} \leftarrow$ empty list, $c \leftarrow$ empty list, $\Lambda \leftarrow$ empty list
for Node $N \in R.child$ **do**
    new schedule $s \leftarrow$ empty list, new incidence vector $\lambda = \mathbf{0}_{n\times 1}$.
    Add $N.zone$ to s
    $\lambda[N.zone] \leftarrow \lambda[N.zone] + 1$
    if $\lambda[N.zone] \leq a[N.zone]$ **then**
        if $N.save \geq C$ **then**
            Add s to $\mathcal{S}$, add $N.save$ to c, add λ to Λ
        end if
        $Query(N, s, \lambda, a)$
    end if
end for
function QUERY(M, s, λ, a)
    for Node $N \in M.child$ **do**
        Add $N.zone$ to s
        $\lambda[N.zone] \leftarrow \lambda[N.zone] + 1$
        if $\lambda[N.zone] \leq a[N.zone]$ **then**
            if $N.save \geq C$ **then**
                Add s to $\mathcal{S}$, add $N.save$ to c, add λ to Λ
            end if
            $Query(N, s, \lambda, a)$
        end if
    end for
end function

probability approaching one exponentially fast as the sample size is increased. These conditions are that (1) the set of feasible solutions in the first stage is finite, (2) the recourse function is measurable, that is, $\mathbb{E}[Q(x_{t_0}, \xi_{t_0})]$ is finite and (3) the distribution of $\xi^i_{t_0}$ has a finite support which means $\xi^i_{t_0}$ has finite number of possible realizations. For SHRP, the first condition clearly holds as the number of passengers and schedules in the current time step is finite. The second and third conditions always hold in case the number of passengers in future time steps d_t and the length of the horizon $T_{horizon}$ are finite.

ASHRP is a deterministic ILP and can be solved directly by commercial integer programming solvers. However, with the number of scenarios M gets higher, the computational time will increase exponentially so this type of methods is doomed to fail. To address this issue here we introduce the dual decomposition technique proposed in (*25*). The basic ideal of this method is to relax some of the constraints to decompose the original problem into several subproblems. Note the ASHRP is equivalent to solving

$$z_D = \max_{x^1_{t_0}, x^2_{t_0}, \ldots, x^M_{t_0}} \{\frac{1}{M}\sum_{i=1}^{M}[c^\intercal x^i_{t_0} + \beta \mathbf{1}^\intercal_{n\times 1}(\Lambda_{t_0} x^i_{t_0} - a_{t_0}) + Q(x^i_{t_0}, \xi^i_{t_0})] : x^1_{t_0} \leq x^2_{t_0} \leq \cdots \leq x^M_{t_0} \leq x^1_{t_0}\} \tag{21}$$

where the above constraints ensure that $x^1_{t_0} = x^2_{t_0} = \cdots = x^M_{t_0}$, and $x^i_{t_0}$ also need to satisfy the same constraints of x_{t_0} in SHRP. By relaxing the constraints shown in (21) we can solve the above problem by solving single scenario and them combine all scenarios together. Let λ be the vector of Lagrangian multipliers and a proper matrix H^i to describe the coefficients of $x^i_{t_0}$ in the constraints, we solve

$$D_i(\lambda) = \max_{x^i_{t_0}} \{c^\intercal x^i_{t_0} + \beta \mathbf{1}^\intercal_{n\times 1}(\Lambda_{t_0} x^i_{t_0} - a_{t_0}) + Q(x^i_{t_0}, \xi^i_{t_0}) + \lambda^\intercal H^i x^i_{t_0} : \lambda \geq 0\} \tag{22}$$

$$D(\lambda) = \frac{1}{M}\sum_{i=1}^{M} D_i(\lambda) \tag{23}$$

The Lagrangian dual of ASHRP can be written as

$$z_{LD} = \min_{\lambda} D(\lambda) \tag{24}$$

where λ can be updated by its sub-gradient in $D(\lambda)$ which is $\partial D(\lambda) = \frac{1}{M}\sum_{i=1}^{M} H^i x^i_{t_0}$.

A well-established result (*26*) is that the optimal value of the Lagrangian dual, z^*_{LD}, is the upper bound of the optimal value of original problem z^*_D since z^*_{LD} gives the optimal value over the convex hull of the feasible region. Therefore for some choice of λ with the corresponding solution of $D(\lambda)$ satisfies the constraints in the original problem, the solution is also the optimal solution of the original problem, and λ is the optimal solution of the Lagrangian dual problem. Based on this result, a branch and bound algorithm is proposed in (*25*) and we implement that algorithm for solving ASHRP. The detailed algorithm is shown as follows.

Step 1: Set the lower bound and upper bound of x_{t_0} to ensure for every element $x^{(k)}_{t_0}$, $\Lambda^{(k)}_{t_0} x^{(k)}_{t_0}$ does not violate the demand constraint (Constraint (11)); Let P_1 be the problem (1), add it to the set of problems $\mathcal{P}$; Set the current known upper bound to be ∞ and lower bound to be $-\infty$.

Step 2: Select the problem $P_i \in \mathcal{P}$ with the highest upper bound, solve its Lagrangian relaxation (Equation (24)).

Step 3: Set the solved objective value to be the upper bound of problem P_i, if P_i's upper bound is lower than the current known lower bound, then (stop exploring it and) go to Step 2.

Step 4: Take the average of all scenario solutions $x^i_{t_0}$, round each element by the order of corresponding profit $c^k_{t_0}$ and ensure the demand constraint is not violated, that is, for each element $\bar{x}^k_{t_0}$, its rounded result $\hat{x}^k_{t_0} = \min$ (rounded value of $\bar{x}^k_{t_0}$, the maximal value that fit the demand constraint).

Step 5: Solve the optimal value of $c^\intercal \hat{x}_{t_0} + \beta \mathbf{1}^\intercal_{n\times 1}(\Lambda_{t_0}\hat{x}_{t_0} - a_{t_0}) + \frac{1}{M}\sum_{i=1}^{M} Q(\hat{x}_{t_0}, \xi^i_{t_0})$ to get the lower bound of problem P_i, if the lower bound of P_i is greater than the current known lower bound, then let the current lower bound be P_i's lower bound and remove all problem in $\mathcal{P}$ with upper bound lower than current lower bound.

Step 6: Select the component of $x^k_{t_0}$ with the largest variance among the multi-scenario's solutions for P_i. Add two new problems to $\mathcal{P}$ obtained from P_i by adding the constraints $x^k_{t_0} \le \lfloor \hat{x}^k_{t_0} \rfloor$ and $x^k_{t_0} \ge \lfloor \hat{x}^k_{t_0} + 1 \rfloor$. Let the upper bound of new problems be the upper bound of P_i.

Step 7: Set the current known upper bound to be the maximal upper bound of all problems in $\mathcal{P}$. If $\mathcal{P} = \emptyset$ or the difference between current known upper bound and current known lower bound is less than a given threshold. Stop the algorithm and output $\hat{x}_{t_0}$ as the optimal solution.

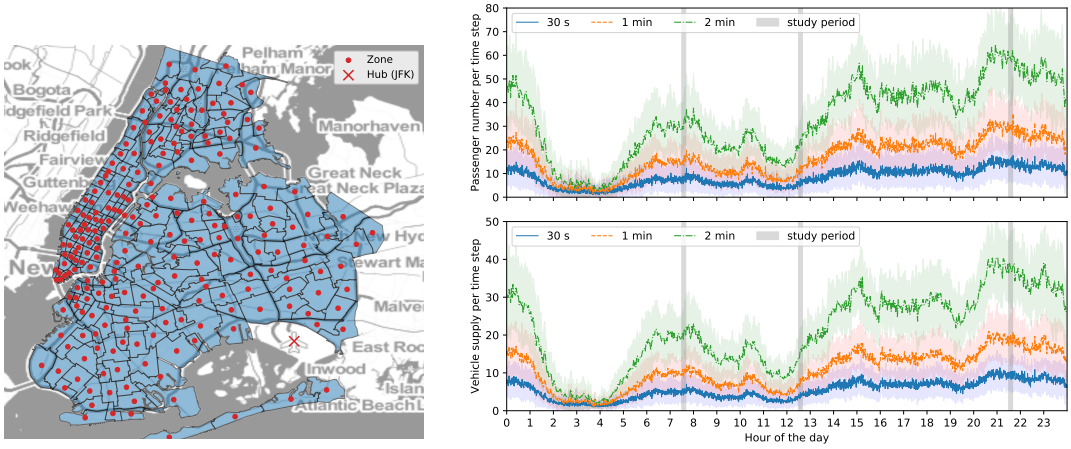
Note for solving one instance of ASHRP using the above algorithm, the computational time is polynomial to three factors: the time of solving single scenario subproblem $D_i(\lambda)$, the number of scenarios and the number of iterations for updating λ. In practice, one can solve different subproblems in parallel, and test a group of randomly generated Lagrangian multipliers also in parallel, then choose the one that returns the maximal objective value to approximate the original solution. Thereby, the time consumption for solving one problem in the above algorithm can be brought down to the same level of solving single scenario subproblem and the sub-optimal choice of the Lagrangian multiplier will just lead to a slightly worse upper bound.

After solving one instance of ASHRP, the remaining problem is to evaluate the quality of the solution. This can be done by testing the solution on another group of scenarios (usually with a larger size than for solving the solution) generated independently. And one framework to improve the solution is to solve ASHRP for multiple scenario sets, evaluate each of them and select the best one. Nevertheless, this will increase the computational cost. Since solving one instance of ASHRP is already challenging, here we only test the performance of one instance of ASHRP. Interested readers are referred to (*24*) to find more details.

NUMERICAL EXPERIMENT

Experiment settings

We demonstrate the effectiveness of our method on a real scale problem by simulating the hub-based ridehsharing service at JFK airport in NYC. To quantify the modeling parameters, we collect information from publicly available datasets including road information, geographical subdivisions of the city, trip records of taxi and for-hail vehicle (FHV) associated with each taxi zone in March 2019 during which there is no reported exceptional event.



(a) Study area (b) Demand and supply per time step

FIGURE 2: Overview of NYC data for numerical experiment

As shown in Figure 2 (a), we take the entire NYC except for the State Island (due to the low level of demand) as our study area, and consider serving all demand with ridesharing algorithm to investigate the potential of hub-based ridesharing services. Since for FHV trips the passenger number is not provided in the dataset, we randomly assign the passenger number to FHV trips based on the passenger number distribution for taxi trips. Then we aggregate the demand by three different time steps (30s, 60s and 120s) to obtain the mean and variance of demand for each OD pair within each time step. Figure 2 (b) shows the average number of passengers for each time step and the shaded area represents the standard deviation observed in the historical data. For supply information, we use the mean and variance of requests served per time step as the mean and variance of the corresponding vehicle flows. We select three time periods to present AM peak (7:30-7:40), off-peak (12:30-12:40), and evening peak(21:30-21:40) as our study period to perform numerical experiments. In the simulation, the demand and supply for each time step d_t and v_t are draw from the normal distributions based on the mean and variance extracted from historical data, then the value is rounded and truncated (to $[0, \lceil mean + 3 * variance \rceil]$) to get valid inputs in the simulation.

We use the travel distance as the profit function between different zones; In other words, we consider each element of profit matrix c as the saving of travel distance for each candidate schedule. And we set the trade-off coefficient between saved travel distance and additional waiting of one passenger β as 2km per minute. The vehicle capacity γ is set to 3.

We implement the branch and bound algorithm for solving ASHRP in Python 3.7, which calls CPLEX 12.5.1 in parallel to obtain the solution for each scenario. The numerical experiments are performed on a workstation with 16 Intel 2.90 GHz Xeon E3-2690 CPUs to take advantage of parallelism. After some preliminary tests, we select the number of scenarios $M = 5$, $T_{horizon} = 5, 5$ and 3 respectively for time step length $\Delta t = 30$s, 60s and 120s. The threshold for stopping criteria is 2% for the overall problem and for each subproblem solved by CPLEX, the minimal gap is set to 1% to ensure one instance of ASHRP can be solved within one time step in most of the case. We also set the maximum number of problems to explore in the branch and bound algorithm as 50 to ensure the result will be returned in reasonable time.

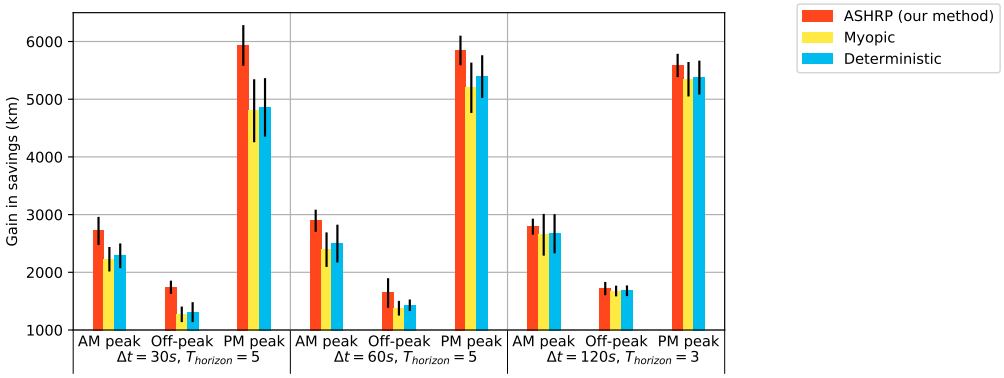


FIGURE 3: Ridesharing profits (saved vehicle distance) under different approaches and parameters

Besides our proposed method, we consider two other approaches as the baselines in this study.The first one is the **myopic** ridesharing algorithm which maximizes the profit within a single time step; The second one is the **deterministic** rolling horizon algorithm in which only one scenario represented by the mean of the future demand and supply is considered. Then for each method with each set of parameters, we calculate the ridesharing profits under four random seeds $(0, 10, 100, 1000, 10000)$ and take the average as the final performance.

Performance comparison

Figure 3 shows the performance of different ridesharing approaches, the errorbars represent the standard deviation of the results generated by using different random seeds. It can be observed that ASHRP outperforms both the myopic ridesharing and deterministic rolling horizon approach in all cases. For $\Delta t = 30s$, ASHRP outperforms the myopic ridesharing by 490 km (22%) in AM peak, 469 km (37%) in off-peak and 1132 km (25%) in PM peak; ASHRP obtains more savings than deterministic rolling horizon by 431 km (19%) in AM peak, 431 km (33%) and 1073 km (22%). We also observe that ASHPR has more stable gains. Similar statements hold for $\Delta t = 60s$, except in off-peak the ASHRP savings have relatively high variance. The performance of myopic and deterministic methods also becomes better when compared with the corresponding results for $\Delta t = 30s$. For $\Delta = 120s$, the performance of baseline methods become even better given more information is revealed for each step, while the saving of ASHRP decreases due to the increased problem complexity, especially for PM peak, where the level of demand is the highest and the ASHRP fails to reach the optimality condition before the maximum iteration reached. It is worth to noting that ASHRP still has better performance with $\Delta t = 30s$ than the deterministic rolling horizon method with $\Delta_t = 120s$. Since in real world application, small Δt is preferred to support quick response to the users' request, these results sufficiently demonstrate the effectiveness of ASHRP.

To gain further insights on the differences among the three methods, we visualize the accumulated savings and vehicle occupancy (passengers per vehicle trip) under different choices of model parameters. Figure 4 shows the accumulated savings under different parameters for different time periods. It can be observed that ASHRP results have relatively low savings in the beginning. Then after certain time steps, the performance of ASHRP surpasses the other methods. For $\Delta t = 120s$, given the total number of the time steps is low, we hardly can observe this phe-

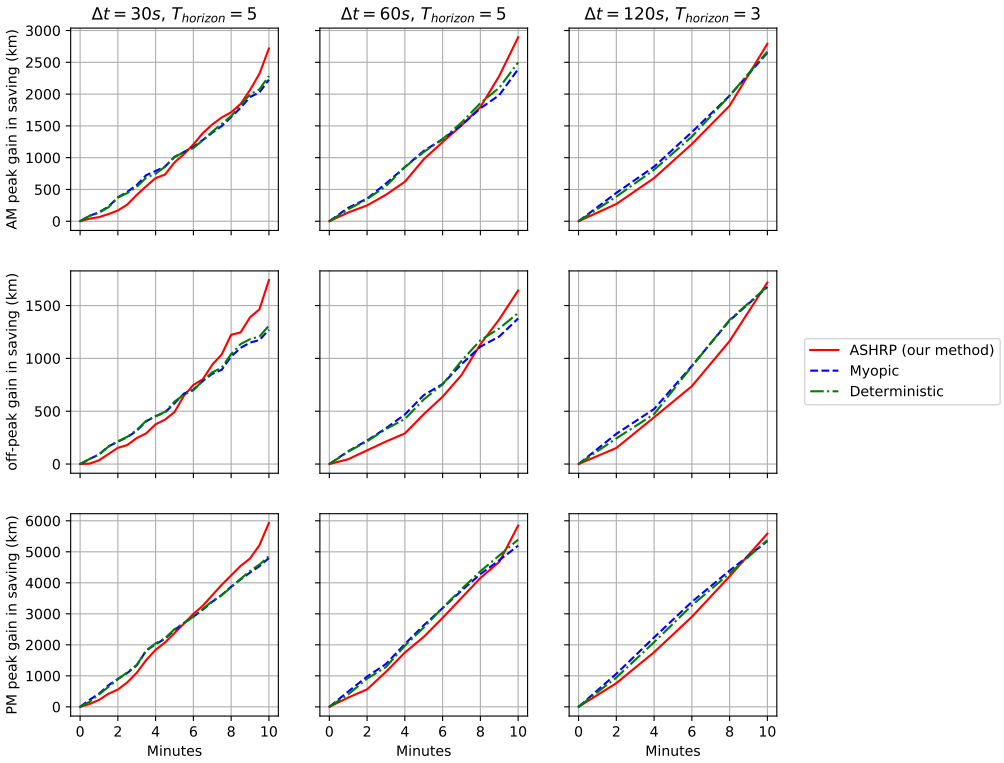


FIGURE 4: Accumulative profit (gain in savings) under different parameters

nomenon. This also explains why the performance of ASHRP would decrease for large Δt in our case study.

Figure 5 shows the average vehicle occupancy of vehicle trips per time step in different cases. It can be observed that ASHRP results in the highest vehicle occupancy in nearly all cases. The results of the deterministic rolling horizon method have higher vehicle occupancy than the myopic algorithm's, but the difference is much smaller than the gap between ASHRP and other methods. This suggests that although the deterministic rolling horizon method can capture some potential matching between exist passengers and future demand, it not as competitive as ASHRP which exploits more potential ridesharing schedules based on the probability.

Computational time

We record the total CPU time spent on solving every single time step's ASHRP problem from CPLEX Python API and summarize the statistics of the CPU time for each case in Figure 6. Note we solve each subproblem in parallel, so the real time cost for solving each ASHRP instance is smaller than the total CPU time. Here we still use the CPU time because it is directly generated by commercial solver while other timers (like Python runtime) can be hugely influenced by detailed implementations. From Figure 6, we first observe that a few records have extremely high CPU times, which heavily inflate the mean for solving each ASHRP. This suggests an early stop mechanism is required for adopting this algorithm in practice. Besides this, we also observe that the CPU time surges as the time step increases from 60s to 120s. The underlying reason is that the number of schedules increases exponentially as more passengers are considered into a single time step, and so as the time cost for solving single scenario subproblems. To our surprise, when $\Delta t = 30s$,

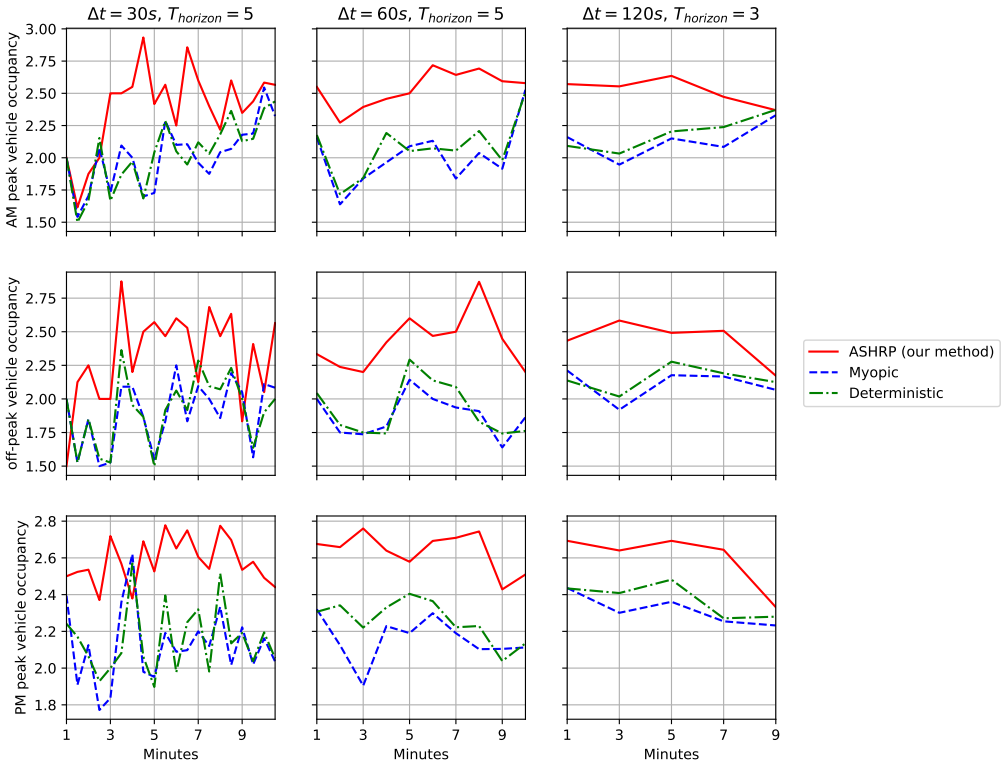


FIGURE 5: Vehicle occupancy under different parameters

the computational cost for solving the off-peak problem is higher than solving the problem of AM peak and PM peak. This might be caused by the low objective value, which leads to a more strict threshold for the solution algorithm to converge.

CONCLUSION

In this study, we develop a dynamic ridesharing algorithm tailored to transportation hubs. In the problem formulation, we consider both the future effect and the stochastic of the future state of demand/supply. Based on the theory of stochastic integer program, we propose a solution algorithm based on scenario decomposition using Lagrangian relaxation. To demonstrate the effectiveness of the proposed approach, we use NYC taxi+FHV data and Google Map API to obtain the road traffic information, then simulate the ridesharing services at JFK airport. The numerical results confirmed that our approach consistently outperform the myopic solution and the rolling horizon solution with point-wise estimation. For future studies, the solution algorithm can be further improved by introducing early stop mechanisms and the proposed approach can be further validated with more comprehensive configurations.

AUTHOR CONTRIBUTIONS

The authors confirm contribution to the paper as follows: study conception and design: Z. Lei, X. Qian, X. Chen, S.V. Ukkusuri; data collection: Z. Lei, X. Chen; analysis and interpretation of results: Z. Lei, X. Qian; draft manuscript preparation: Z. Lei, X. Chen, X. Qian, S.V. Ukkusuri. All authors reviewed the results and approved the final version of the manuscript.

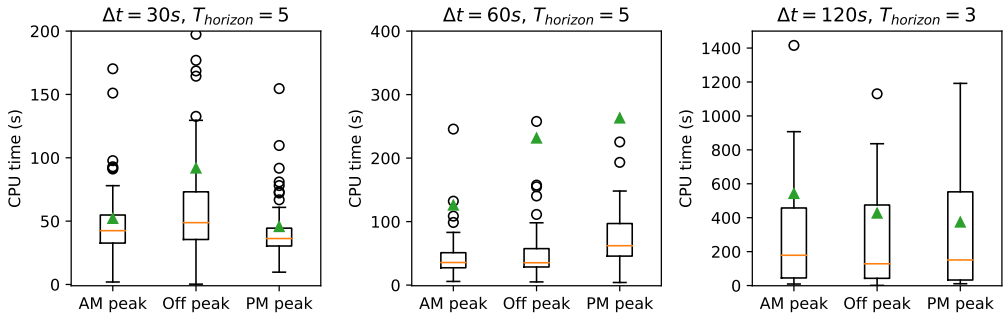


FIGURE 6: CPU time under different parameters. The red line represent the median, the green triangle marks the mean and each box extends from the lower to upper quartile values of the CPU time

ACKNOWLEDGEMENT

This work is partly funded by the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy, under Award Number DE-EE0008524. The authors are solely responsible for the findings in this paper.

REFERENCES

[1] Adella Santos, Nancy McGuckin, Hikari Yukiko Nakamoto, Danielle Gray, and Susan Liss. Summary of travel trends: 2009 national household travel survey. Technical report, 2011.

[2] Kai Zhang and Stuart Batterman. Air pollution and health risks due to vehicle traffic. *Science of the total Environment*, 450:307–316, 2013.

[3] Shuguang Ji, Christopher R Cherry, Matthew J. Bechle, Ye Wu, and Julian D Marshall. Electric vehicles in china: emissions and health impacts. *Environmental science & technology*, 46(4):2018–2024, 2012.

[4] Salvatore Failla, Enzo Bivona, and Nazareno Ventola. Exploring airports' landside congestion impacts on the dynamic of passengers satisfaction. In *International Conference of the System Dynamics Society*, number 1. System Dynamics Society, 2014.

[5] NYCTLC. 2018 TLC fact book, accessed July, 2020. Available online at `https://www1.nyc.gov/assets/tlc/downloads/pdf/2018_tlc_factbook.pdf`.

[6] Masabumi Furuhata, Maged Dessouky, Fernando Ordóñez, Marc-Etienne Brunet, Xiaoqing Wang, and Sven Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28–46, 2013.

[7] Mustafa Lokhandwala and Hua Cai. Dynamic ride sharing using traditional taxis and shared autonomous taxis: A case study of nyc. *Transportation Research Part C: Emerging Technologies*, 97:45–60, 2018.

[8] Uber. What is uberpool, accessed July, 2020. Available online at `https://www.uber.com/us/en/ride/uberpool/`.

[9] Lyft. About shared rides, accessed July, 2020. Available online at `https://help.lyft.com/hc/en-us/articles/115013078848-About-Shared-rides#sharedpricing`.

[10] Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295–303, 2012.

[11] Shuo Ma, Yu Zheng, and Ouri Wolfson. T-share: A large-scale dynamic taxi ridesharing

service. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 410–421. IEEE, 2013.

[12] Douglas Oliveira Santos and Eduardo Candido Xavier. Dynamic taxi and ridesharing: A framework and heuristics for the optimization problem. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[13] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 114(3):462–467, 2017.

[14] Neda Masoud and R Jayakrishnan. A real-time algorithm to solve the peer-to-peer ride-matching problem in a flexible ridesharing system. *Transportation Research Part B: Methodological*, 106:218–236, 2017.

[15] Jiyao Li and Vicki H Allan. A ride-matching strategy for large scale dynamic ridesharing services based on polar coordinates. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 449–453. IEEE, 2019.

[16] Yi Xu, Yongxin Tong, Yexuan Shi, Qian Tao, Ke Xu, and Wei Li. An efficient insertion operator in dynamic ridesharing services. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1022–1033. IEEE, 2019.

[17] Amirmahdi Tafreshian and Neda Masoud. Trip-based graph partitioning in dynamic ridesharing. *Transportation Research Part C: Emerging Technologies*, 114:532–553, 2020.

[18] Qiulin Lin, Lei Dengt, Jingzhou Sun, and Minghua Chen. Optimal demand-aware ride-sharing routing. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 2699–2707. IEEE, 2018.

[19] Matthew Tsao, Dejan Milojevic, Claudio Ruch, Mauro Salazar, Emilio Frazzoli, and Marco Pavone. Model predictive control of ride-sharing autonomous mobility-on-demand systems. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6665–6671. IEEE, 2019.

[20] Ramon Iglesias, Federico Rossi, Kevin Wang, David Hallac, Jure Leskovec, and Marco Pavone. Data-driven model predictive control of autonomous mobility-on-demand systems. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2018.

[21] Michael Wittmann, Lorenz Neuner, and Markus Lienkamp. A predictive fleet management strategy for on-demand mobility services: A case study in munich. *Electronics*, 9(6):1021, 2020.

[22] Yifang Liu, Will Skinner, and Chongyuan Xiang. Globally-optimized realtime supply-demand matching in on-demand ridesharing. In *The World Wide Web Conference*, pages 3034–3040, 2019.

[23] Chih-Hua Hsu and Haw-Ching Yang. Real-time near-optimal scheduling with rolling horizon for automatic manufacturing cell. *IEEE Access*, 5:3369–3375, 2016.

[24] Shabbir Ahmed, Alexander Shapiro, and Er Shapiro. The sample average approximation method for stochastic programs with integer recourse. *Submitted for publication*, pages 1–24, 2002.

[25] Claus C CarøE and Rüdiger Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1-2):37–45, 1999.

[26] Marshall L Fisher. The lagrangian relaxation method for solving integer programming problems. *Management science*, 27(1):1–18, 1981.