FDA-VeD: A Future-Demand-Aware Vehicle Dispatching Service

Yang Guo[1], Weiliang Zhao[2], Jian Yang[1,2], Zizhu Zhang[1], Jia Wu[1] and Tarique Anwar[1]
1. Department of Computing, Macquarie University, Australia
2. College of Computer Science and Technology, Dong Hua University, P.R. China.
Emails: {yang.guo5@students., weiliang.zhao@, jian.yang@, zizhu.zhang@hdr., jia.wu@, tarique.anwar@}mq.edu.au

***Abstract*—In this paper, we develop a Future-Demand-Aware Vehicle Dispatching Service (`FDA-VeD`) by considering the relocation of idle vehicles based on the predicted future demands in order to achieve high passenger serving ratio. We evaluate the performance of our system on New York taxi dataset. We demonstrate that our approach achieves a significantly higher serving ratio with a low operating cost increase in comparison with existing methods.**

***Keywords*-Vehicle Dispatching; Idle Vehicle Relocation; Passenger serving ratio; Future demand prediction; On-demand Mobility Service;**

I. Introduction

With the development of ubiquitous GPS-enabled devices and the emergence of vehicle dispatching platforms (e.g., Uber, DiDi), it is easy for passengers to send trip requests to a centralized dispatching platform. However, dispatching vehicles for real-time requests is challenging due to the following reasons: (a) Passengers' trip requests are massive and unevenly distributed. (b) The dynamic traffic conditions further add the difficulties [1]. As the trip requests are usually stochastic, during a specific period of time, it is common that for some areas the number of vehicles cannot satisfy the total trip requests (called "undersupply area"), while for some other areas the available vehicles are more than what are requested (called "oversupply area"). The available vehicles in oversupply areas are idle ones. Proactively relocating idle vehicles to undersupply areas can significantly contribute to the dispatching performance. As shown in Figure 1, if the idle vehicle stays at point P_2 without relocation, it would take 10 minutes to arrive at the pickup location P_4. But if the idle vehicle was proactively relocated to the undersupply area Z, it would take only two minutes. Thus, proactive relocation of idle vehicles to undersupply areas have a significant impact on improving the serving ratio.

Due to the practical applications of vehicle dispatching, the problem of achieving a high serving ratio has drawn significant attention [2]–[5]. However these works have limitations in handling real-time requests with a consideration of potential future demands. As a result, these dispatching algorithms can only run in every high frequency (e.g., every 30 seconds) and brings high operating cost if future demands are considered.

Although some work relocates vehicle in a longer time interval [6], they only consider the future pickup demand and proposed a queueing based algorithm to relocate the idle vehicles. The queuing based algorithm is not efficient to get the relocation solution in real-time. Hence, a dispatching system that proactively relocates idle vehicles at a low cost while considering both the future pickup and drop-off demands is highly needed.

In this paper, we propose a **F**uture-**D**emand-**A**ware **Ve**hicle **D**ispatching service system, called `FDA-VeD`. The `FDA-VeD` system is able to improve the quality of the urban on-demand mobility service, especially in relation to achieving a high serving ratio while incorporating future demand prediction. It contains an offline phase to process basic data (e.g., road graph, estimated travel time, graph partitioning) and an online phase to dispatch and relocate vehicles. The offline phase pre-processes the basic data and trains the future demand prediction model. The online phase finds the maximum matching of trip requests and available vehicles and proactively relocates idle vehicles between subareas. By considering the potential demands in a relative long future time interval (e.g., 10 minutes), idle vehicles are relocated in a low frequency which generates a low operating cost increase. Overall, we make the following main contributions in this paper:

- We develop a system that proactively relocates idle vehicles to minimise the supply and demand gap, and significantly improves the passenger serving ratio without increasing the total number of vehicles.
- We propose an efficient travel time based road graph partitioning algorithm. The relocation of idle vehicles between subareas is done according to their potential future travel demands and vehicles.
- We propose a novel two time-interval based idle vehicle relocation algorithm: advance time-interval (for relocating idle vehicles) and future time-interval (the target interval to minimize supply-demand gap).
- We perform extensive experiments on a real dataset. Results show that `FDA-VeD` outperforms the state-of-the-art vehicle dispatching system in terms of passenger serving.

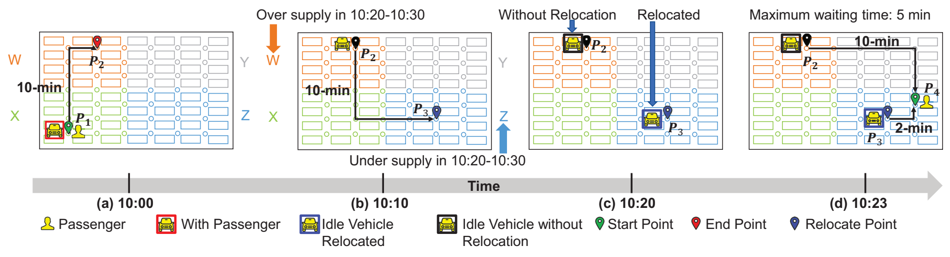


Figure 1. **An example for relocating idle vehicles.** The big rectangle stands for the whole serving area, and the small rectangles and circles stand for the buildings and pickup/drop-off points respectively. There are four subareas: W, X, Y and Z, highlighted by different colours. The subfigures at different time points illustrate the process of idle vehicle relocation. **(a)10:00** A vehicle picked up a passenger at point P_1 at 10:00, and spent 10 minutes to arrive the destination P_2. **(b)10:10** A future supply–demand scenario was predicted at 10:10 and showed that area W would be in oversupply during 10:20-10:30, while area Z would in undersupply during 10:20-10:30. The vehicle could then be relocated to point P_3 belonging to area Z and would arrive at 10:20. **(c)10:20** If the vehicle was relocated, it would stay at P_3. Otherwise, it would remain at P_2. **(d)10:23** At 10:23, a passenger sent a request at P_4 and needed to be picked up in five minutes. It takes 10 minutes from P_2 while only two minutes from P_3. Then the relocated vehicle could serve this request. By proactively relocating idle vehicles between undersupply and oversupply areas, the passenger serving ratio could be improved.

II. PROBLEM FORMULATION

A. *Preliminaries*

The future demand aware vehicle dispatching problem involves three different entities, which are *passengers*, *vehicles*, and *a centralized vehicle dispatching system*. Passengers send their trip requests to the centralized dispatching system using their GPS-enabled smart devices in real time. After receiving a set of trip requests in a batch time, the system needs to match these trips with the available vehicles in a cost-efficient manner. We formulate this problem with the help of following definitions.

*Definition 1 (**Trip Request**):* A *trip request* T_i, defined as a tuple (t_i^p, l_i^p, l_i^d), is a trip requested by a passenger at the t_i^p (the earliest time when the passenger can be picked up) from location l_i^p, to drop off at location l_i^d. A set of trip requests during a particular time interval is denoted by $\mathcal{T} = \{T_1, T_2, ..., T_{n_t}\}$.

*Definition 2 (**Vehicle Information**):* The information about a vehicle v_i is defined as a tuple (p_i, s_i, t_i^d), where p_i is the vehicle destination point, s_i is its current status (0, 1, 2, 3 stand for available, with-passenger, dispatching and relocating respectively), and t_i^d is the destination arriving time.

*Definition 3 (**Served Trip**):* A trip request T_i is called as a *served trip* if the passenger is actually picked up between t_i^p and $t_i^p + \Delta$, where Δ is a pre-defined serving threshold. The set of trip requests already served by the dispatching system is denoted by $\mathcal{T}_{served}$, which is a sub set of the total set of trip requests $\mathcal{T}$. Mathematically, $\mathcal{T}_{served} = \{T_i | t_i^p \leq {}^a t_i^p \leq t_i^p + \Delta\}$, where ${}^a t_i^p$ is the actual pickup time.

*Definition 4 (**Serving Ratio**):* Given the set of served trips $\mathcal{T}_{served}$ and all the trip requests $\mathcal{T}$, the *serving ratio* R of the centralized vehicle dispatching system is defined as the ratio of $\mathcal{T}_{served}$ to $\mathcal{T}$, i.e., $R = \frac{|\mathcal{T}_{served}|}{|\mathcal{T}|}$.

B. *Problem Definition*

For this problem, we focus on improving the passenger serving ratio from the central dispatching platform's perspective, and every vehicle can be relocated according to the platform's instructions. A vehicle $v_i \in \mathcal{V}$ can serve only one trip request at one time. It can start to serve a new trip request only after it has arrived at the destination of its last trip. With a given number of vehicles n_v on a road graph G, a set of real-time trip requests $\mathcal{T}$, a set of historical trips $\mathcal{H}$, the objective of the centralized vehicle dispatching system is to serve the maximum number of real-time trip requests $\mathcal{T}$, and thus achieve a high serving ratio R.

$$\begin{aligned} \text{maximize} \quad & R = \frac{|\mathcal{T}_{served}|}{|\mathcal{T}|}, \\ \text{subject to} \quad & \forall t_i \in \mathcal{T}_{served}, \;\; t_i^p \leq {}^a t_i^p \leq t_i^p + \Delta \end{aligned} \tag{1}$$

III. THE PROPOSED SERVICE: FDA-VeD

We develop a future demand aware vehicle dispatching service system (FDA-VeD) that achieves a high passenger serving ratio with a low operating cost. To achieve this objective, it is important to have available vehicles in areas close to the pickup locations of the trip requests coming into the system. Our solution considers both real-time vehicle dispatching and idle vehicle relocation based on the predicted demand in the near future. The architecture of the FDA-VeD system is shown in Figure 2.

The system works in two phases. It first starts with an offline phase that runs only once for the initial set up. There are fours main modules in this first phase, which are road graph extraction as a preprocessing step, estimating the travel times, partitioning the road graph based on estimated travel times to identify the different subareas, and future demand prediction for the different subareas. On the other hand, the online phase consists of two modules, which are vehicle–request matching and supply–demand balancing. The vehicle–request matching module is used to serve a

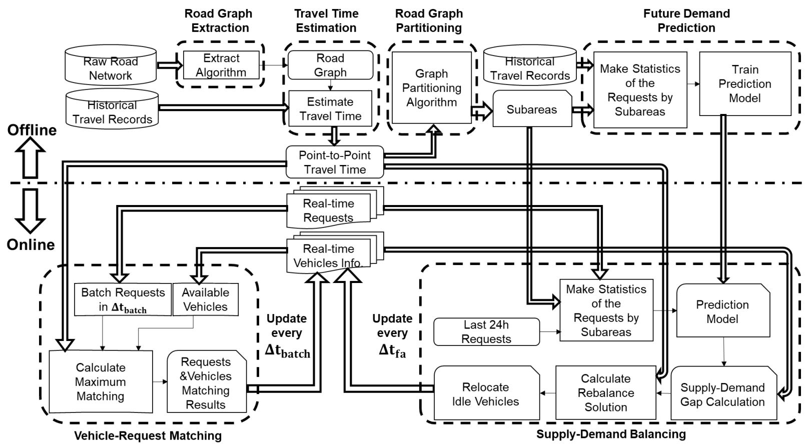

Road Graph Extraction
Travel Time Estimation
Road Graph Partitioning
Future Demand Prediction
Raw Road Network
Extract Algorithm
Road Graph
Graph Partitioning Algorithm
Historical Travel Records
Make Statistics of the Requests by Subareas
Train Prediction Model
Historical Travel Records
Estimate Travel Time
Subareas
Offline
Point-to-Point Travel Time
Real-time Requests
Online
Real-time Vehicles Info.
Batch Requests in Δt_{batch}
Available Vehicles
Update every Δt_{batch}
Update every Δt_{fa}
Make Statistics of the Requests by Subareas
Prediction Model
Last 24h Requests
Calculate Maximum Matching
Requests &Vehicles Matching Results
Relocate Idle Vehicles
Calculate Rebalance Solution
Supply-Demand Gap Calculation
Vehicle-Request Matching
Supply-Demand Balancing

Figure 2. **Architecture of future-demand-aware vehicle dispatching service (FDA-VeD)**

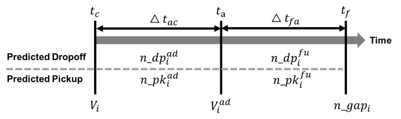


Figure 3. **Calculate the supply–demand gap in advance and future time intervals**

maximum possible number of real-time trip requests with a limited number of available vehicles on the road graph and runs every Δt_{batch} minutes. The supply–demand balancing module minimizes the supply–demand gap in the serving area, which means that it minimizes the difference between the numbers of supplied vehicles and travel demands in a specific area during a future time interval and runs every Δt_{fa} minutes.

A. *Offline Phase*

1) Road Graph Extraction: The road network topology of a city is the most fundamental information required for vehicle dispatching. In this work, we capture the required topology information in the form of a road graph, defined earlier. The road graph is essential to compute the estimated travel times, which is also shown in Figure 2. Most open data of city roads are not specially designed for vehicle dispatching purpose, so we develop an algorithm to extract the points and directed links from the generally available open road network data. The original road network in its raw form is defined as a graph $G' = (P', L')$ and the extracted road graph is defined as $G = (P, L)$. In the extraction algorithm, all intersection points are reserved. The length of each link is less than $length_{max}$ meters. In this research $length_{max} = 200$, which makes any passenger could find a nearest pickup/drop-off point in 100 meters. In the following research, the points P in road graph $G = (P, L)$ are used as the pickup/drop-off point for a passenger's trip record. Each trip request's pickup/drop-off location will be matched to the nearest point.

2) Travel Time Estimation: For a new trip request, it is essential to have an estimation of the travel time from an available vehicle's current location to the desired pickup point, in order to find the available vehicles to serve the request. In our system, as shown in Figure 2, we need to have an estimation of all point-to-point travel times, required for the other following modules of road graph partitioning, vehicle–request matching, and supply–demand balancing.

To calculate the point-to-point travel times, we first calculate the travel times corresponding to each directed link l_i in the road graph G. For this calculation, we use a recently proposed travel time estimation algorithm [7]. Here, we divide the historical trip record $\mathcal{H}$ by hours to determine the difference in times within 24 hours. The precision can be improved further by dividing $\mathcal{H}$ by months, weekdays, hours and so on. In this work, the computed travel time is different at different hours of the day. With the travel times for each link, we calculate all the shortest travel times between each pair of points in the road graph G. $ett_{i,j,h}$ denotes the shortest travel time from point i to point j in hour h.

3) Road Graph Partitioning: We partition the road graph to divide the whole serving area to subareas. As shown in Figure 2, it is based on these subareas that the supply–demand gap is calculated and the idle vehicles are relocated to the undersupply subareas.

The whole serving area, denoted by $\mathcal{A}$, contains all the points $P \in G$. As shown in Equation 2, the objective of the road graph partitioning step is to divide $\mathcal{A}$ into a set of k non-overlapping subareas, each of which, denoted by A_i, contain a small set of points of P, such that their union is P.

$$\begin{aligned} \mathcal{A} &= \cup_{i=1}^{k} A_i \\ A_i \cap A_j &= \varnothing \ (i \neq j) \end{aligned} \tag{2}$$

Different from existing road network partition problem, we are interested in the pickup/drop-off locations based on the travel time between them and a central location in these locations that can serve as the partitions. We redefine the road graph partitioning problem according to this objective and propose a light-weight heuristic algorithm for its solution. Our road graph partitioning approach is based on the following two considerations.

- *Based on travel time:* The partitioning of the area is done based on travel times between each pair of points, rather than the suburb division or location information. After the having the resulting subareas, a central point in each of them can be found, from where the largest number of points in the corresponding subareas can be served within the passenger's waiting time Δ.
- *Reasonable subarea size*: The size of a subarea should not be too big. If a subarea is too big, many vehicles would be dispatched to its central point which could cause high traffic congestion. Subareas of reasonable sizes can be obtained by setting a reasonable maximum number of points (n_{max}) in a subarea, based on the maximum length of l_i in the road graph $G = (P, L)$.

Considering the above points, we proposed a heuristic road graph partitioning algorithm. There are three main steps for this algorithm: (1) find the longest travel time slot in 24 hours ; (2) set the maximum number of points in a subarea, notated as n_{max}, and the the total number of subareas, notated as $n_{sub} = \lfloor n_{point}/n_{max} \rfloor$; (3) divide subarea based on travel time. To be more specific, we first find the point that can achieve the biggest number of points in maximum waiting time. Then we set this point as a central nodes and add its top n_{max} nearest nodes to a new subarea. After finding a subarea, we delete the nodes belonging to it and search for the next subarea in the reaming points. Finding road graph partitioning solution can be solved in $\mathcal{O}(n_{sub}|A|^2)$ running time.

4) Future Demand Prediction: In this module, we train a model to predict the travel demands at time intervals in the near future in different subareas. The trained prediction model is used in the supply–demand balancing module (discussed in Section III-B2), as shown in Figure 2, to know about the potential future demand in advance so that the supply can be provided accordingly. Travel demands prediction is an important research area for vehicle dispatching and fleet management. It has received a significant amount of attention in the recent past. All such of travel demands prediction techniques are suitable to be used in the proposed `FDA-VeD` system. Travel demand prediction has received considerable attention [8], [9] in recent years. All methods of travel demand prediction could be used in a `FDA-VeD` system. To present our system and experimental results, we selected the random forest regression model [10] to predict the future demand, as it is easy to implement and efficient to train.

As shown in Figure 3, the `FDA-VeD` proactively relocate the idle vehicles to minimize the supply–demand gap for a future time interval $(t_a, t_f]$. All the relocation progress is doing in a advance time interval $(t_c, t_a]$, which means all relocated idle vehicles should arrive their destinations before the time point t_a. To calculate the future supply–demand gap, four variables should be calculated: the number of pickup demands in the advance time interval in each subarea ($n\_pk_i^a$), the number of drop-off demands in the advance time interval in each subarea ($n\_dp_i^a$), the number of pickup demands in the future time interval in each subarea ($n\_pk_i^f$), the number of drop-off demands in the future time interval in each subarea ($n\_dp_i^f$). Then for two time intervals—$(t_c, t_a]$ and $(t_a, t_f]$—we build two models to predict the pickup/drop-off demands. Both models have same input: every subarea A_i's pickup and drop-off demand time series data in last 24 hours.

The model predicts the pickup/drop-off demand number in the time interval $(t_c, t_a]$ by producing the pickup demand number $n\_pk_i^a$ and $n\_dp_i^a$ corresponding to each subarea A_i. Similarly, the other model produces the demands number in the time interval $(t_a, t_f]$.

B. Online Phase

The online phase dispatch available vehicles for real-time requests, and relocate idle vehicles between subareas. There are two modules belong to online phase: vehicle–request matching and supply–demand balancing.

1) Vehicle–Request Matching: The trip requests continuously coming into the system need to be served by the available vehicles. This module forms batches of the incoming trip requests, denoted by $\mathcal{T}_b$, in a batch time interval Δt_{batch}, and finds the matching vehicles for them from the set of available vehicles $\mathcal{V}_a$.

The vehicle–request matching module will run every batch time (Δt_{batch} minutes). For each running, the module will take all the trip requests received in last Δt_{batch} minutes and this request set is notated as $\mathcal{T}_b$. Here, we use t_c stands for the current time point and then the batch requests set $\mathcal{T}_b$ is defined as follows.

$$\mathcal{T}_b = \{T_i | (t_c - \Delta t_{batch}) < t_i^p \leq t_c\} \tag{3}$$

To find the maximum matching between the sets of vehicles and requests, we construct a bipartite graph $G^{VR} = (\mathcal{T}_b, \mathcal{V}_a, E)$, where E is the set of connecting edges between between $\mathcal{T}_b$ and $\mathcal{V}_a$. An edge is created between a vehicle v_i and a trip request T_i, if v_i is able to arrive at the desired pickup point of T_i from its current location before $t_i^p + \Delta$, according to the estimated travel time $ett_{i,j,h}$.

Finding the maximum matching for the bipartite graph G^{VR} can be solved in $\mathcal{O}(|E|(|\mathcal{T}_b| + |\mathcal{V}_a|)^{1/2})$ running time using the Hopcroft-Karp algorithm [11]. The vehicle–request matching module will dispatch for received requests every t_{batch} minutes and then re-build a bipartite graph G^{VR}, and repeat the maximum matching operation between $\mathcal{T}_b$ and $\mathcal{V}_a$.

2) Supply–Demand Balancing: While the trip requests are being served by the available vehicles in a real urban dispatching scenario, it is highly likely that some subareas may get a larger (or smaller) number of accumulated vehicles than required. This would result in some subareas with abundant number of vehicles and others with a scarcity of vehicles, thus eventually resulting in a poor passenger serving ratio due to poor utilization of vehicles. The supply–demand balancing module solves this problem by relocating the idle vehicles from the subareas with an abundant supply of vehicles to those with a scarcity, and thus minimize the supply–demand gap in the whole serving area. One possible approach is to relocate the vehicles based on the real-time demand and supply at the current time point. But note that the relocation takes time, and by the time relocation would happen, the supply and demand scenario in the subareas is likely to change. We solve this issue by predicting the future demand in advance, and relocating the vehicles during the available time to serve the future predicted travel demand. The subareas are generated by the offline road graph partitioning module, and the prediction model to know the future travel demand in each subarea is built by the offline future demand prediction module. This module consists of two main steps. Firstly, it calculates the gap between the predicted supply and demand, and then relocates the idle vehicles accordingly.

Supply–Demand Gap Calculation: It is important to know the gap between the future demand and supply in order to make a wise decision on an effective rebalancing of the subareas. This is done by calculating the demand-supply gap and relocating the vehicles in such a way that minimizes the gap. Firstly, the statistics of the requests by subareas is performed to generate the input data for the prediction model. Secondly by taking the inputs to the prediction models trained in the future demand prediction module, the potential future pickup and drop-off demands in each subarea during advance time interval and future time interval $(n\_pk_i^a, n\_dp_i^a, n\_pk_i^f, n\_dp_i^f)$ are obtained.

The supply–demand gap n_gap_i for each subarea A_i is calculated as follows. As shown in Figure 3, there are two time intervals: the advance time interval $(t_c, t_a]$ and the future time interval $(t_a, t_f]$. The advance time interval is used for relocating idle vehicles in such a way that relocation starts at t_c and thereby get a balanced supply–demand between subareas at time t_a. The supply and demand in the future time interval is the target to balance. With this balanced supply–demand, the trip requests are served effectively across the whole serving area during the future time interval $(t_a, t_f]$.

The vehicle amount at current time t_c in subarea A_i is n_V_i. To calculate the supply–demand gap in $(t_a, t_f]$, we firstly obtain the vehicle amount at the advance time t_a:

$$n\_V_i^{ad} = n\_V_i + n\_dp_i^a - n\_pk_i^a \tag{4}$$

Then we calculate the drop-off–pickup gap (gap_i') in each subarea in the future time interval $(t_a, t_f]$:

$$n\_gap_i' = n\_dp_i^f - n\_pk_i^f \tag{5}$$

Consider the potential vehicle amount $n\_V_i^{ad}$ at t_a, and the future drop-off–pickup gap gap_i', we calculate the supply–demand gap with the following formula:

$$n\_gap_i = max(V_i^{ad}, 0) + min(n\_gap_i', 0) \tag{6}$$

Here we explain why we use formula 6 to calculate the supply–demand gap.

- V_i^{ad} (the vehicles amount at time point t_a) defined how many vehicle could be used to relocate in subarea i. If $V_i^{ad} \leqslant 0$, there is no vehicle available to be relocated. If $V_i^{ad} > 0$, there is V_i^{ad} vehicles available for relocation. Then the first part in this formula is $max(V_i^{ad}, 0)$.
- n_gap_i' defined the drop-off–pickup gap in the future time interval. IF $n\_gap_i' > 0$, it means subarea i will have more drop-off than pickup in subarea A_i during $(t_a, t_f]$ and no vehicles are needed to relocate to this area. If $n\_gap_i' \leqslant 0$, the subarea A_i need n_gap_i' vehicles. So the second part in this formula is $min(n\_gap_i', 0)$.

If $n\_gap_i > 0$, it means subarea A_i has more vehicles than needed. Such subareas are called *oversupply area*. On the other hand, if $n\_gap_i < 0$, it means subarea A_i needs more vehicles. Such subareas are called *undersupply area*.

Relocation of Idle Vehicles: The relocation of idle vehicles is to minimize the supply–demand gap in the whole serving area in the time interval $(t_a, t_f]$. One constraint associated with this relocation is that each relocated idle vehicle should arrive at its destination before t_a. This requires an intelligent relocation strategy in order to effectively rebalance the serving area, while satisfying the constraint in the limited amount of time.

To minimize the supply–demand gap, we propose an effective algorithm to relocate idle vehicles between subareas. Before describe this algorithm, we set two concepts: **available vehicle** and **idle vehicle**. An available vehicle means that it is not assigned to any travel request and empty now. An idle vehicle should firstly be an available vehicle and selected as a candidate to relocate from oversupply areas to undersupply areas. Then not all available vehicles in undersupply area can be idle vehicles. Only those in oversupply area have the possibility to become idle vehicles. Concretely, firstly find all the oversupply subareas. Then list all available vehicles in every oversupply subarea. When the number of available vehicles is small that the gap n_gap_i, then all the available vehicles are added into the idle vehicle set V_{idle}. Otherwise, randomly select n_gap_i vehicles from available vehicles, and add them to the idle vehicle set V_{idle}. Then, build the bipartite graph $G^{balance}$ to find the maximum matching between idles vehicle sets and central nodes. As shown in Figure 2, after the physical relocation of vehicles, the information about them is updated in the system. The supply–demand balancing module runs every Δt_{fa} minutes.

IV. Experiments and Results

A. Dataset

Trip records dataset The trip records dataset used in the experiments is from New York City in January 2011 (could be downloaded at: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page). The original dataset contains 13.4 million records. In the experiments, we focus on the trip record whose pickup location and drop-off location are both in Manhattan Island. After the filter, there are 11.2 million records left.

Road network The road network of Manhattan Island was downloaded from www.openstreetmap.org. According to the road conditions in Manhattan, we extracted the following ways: primary, primary_link, secondary, tertiary, residential, unclassified, road and living street. Several other classes are not considered. These include footpaths, trunks or service roads, as they are unlikely to contain pickup or drop-off locations.

B. Experiment Setup

For the whole system, the passenger's maximum waiting time $\Delta = 5$ minutes.

1) Offline Phase:

- **Road Graph Extraction:** The raw road network of Manhattan Island is used to extract road graph. The raw road network $G' = (P', L')$ contains 13587 points and 20919 directed links. Then using the road graph extraction module, we obtained the network $G = (P, L)$ containing 5085 points and 10126 directed links.
- **Travel Time Estimation:** To estimate the travel time, the trip record's pickup and drop-off locations are matched to the nearest points in G. Then filter the record whose pickup point and drop-off point is the same and travel time is shorter than 2 and or longer than 60 minutes. After filtering, there are 10.9 million records. Then we obtained the estimated travel time $\{ett_{i,j,h}\}$ in 24 hours.
- **Road Graph Partitioning:** In this module, the maximum number of nodes in a subarea n_{max} equals to 100 and 50 subareas were generated. As the average length of all links L in graph G is 97.6 meters, the approximate size of a subarea is around $1km^2$. We test the performance of the system by varying n_{max} from 10 to 5000, see Section IV-D2.
- **Future demand prediction:** Here, we set three different value for advance time slot Δt_{ac}: 10, 20, 30. The future time slot Δt_{fa} is set as 10. For each pair of parameters, we train two random forest regression models to predict the future demands for all subareas in the advanced time interval $(t_c, t_a]$ and the future time interval $(t_a, t_f]$ respectively. Trip records during 20110101-20110110 are used to train the models.

2) Online Phase:

- **Vehicle–Request Matching:** For this module, there are two parameters to set: the batch time Δt_{batch} and the number of vehicles $|\mathcal{V}|$. Here, we set $\Delta t_{batch} = 1$, and $|\mathcal{V}| = 6,000$. The experiments will make based on this number of vehicles. Furthermore, we also investigate the effect of the vehicle number changing, where $|\mathcal{V}|$ increased from 2,000 to 10,000.
- **Supply–Demand Balancing:** Based on the above settings, there is no new parameter need to be set. This module will update every Δt_{fa} to relocate idle vehicles to balance supply and demand between subareas.

C. Evaluation Metrics

To evaluate the performance of a vehicle dispatching system, three evaluation metrics were used: serving ratio, with-passenger ratio and gain–cost ratio.

- **Passenger serving ratio**, notated as R, is defined in Definition 4.
- **With-passenger ratio**, notated as r_1, is the ratio of the with-passenger travel distance against total travel distance. As described in Definition 2, there are four status of a vehicle (0, 1, 2, 3 stand for available, with-passenger, dispatching and relocating respectively). We use d_1, d_2, d_3, d_{all} stands for a vehicle's with-passenger, dispatching, relocating, and total travel distance. Then the with-passenger ratio $r_1 = \frac{d_1}{d_{all}}$.
- **Gain–cost ratio**, notated as r_{gc}, is the ratio of the increased with-passenger travel distance and the relocating travel distance. The *increased with-passenger travel distance* is the gain and calculated by $d_g = d_1 - d_1^{baseline}$. Here, $d_1^{baseline}$ is the with-passenger length of a baseline model. The relocating travel distance is the cost and calculated by $d_c = d_3$. Then the gain–cost ratio is calculated by $r_{gc} = \frac{d_g}{d_c}$.

D. Results

We evaluate `FDA-VeD` with the travel requests during 20110112-20110118 (seven days) in Manhattan Island. As there does not exist any other work exactly on the same problem. We select the most closely related work is recently published in *Nature* by Vazifeh et al. [2]. We compare our results with their online batch model [2], called `OBM-VeD` in this paper. `OBM-VeD` is a system containing only three modules: road graph extraction, travel time estimation, and vehicle–request matching. It does not consider the potential future demand to proactively relocate idle vehicles between subareas. It could also be thought as an `FDA-VeD` system with an extreme case—when the whole area is divided into only one subarea. Specifically, when $n_{max} = 5000$, `FDA-VeD` system is equivalent to `OBM-VeD`.

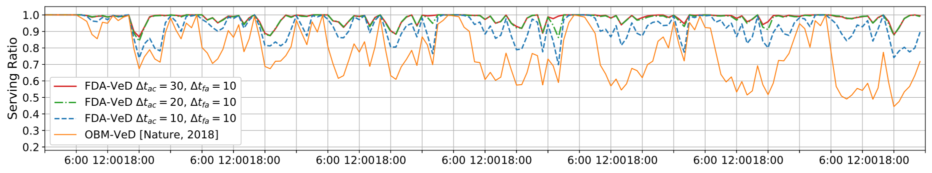
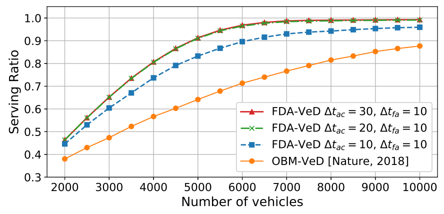
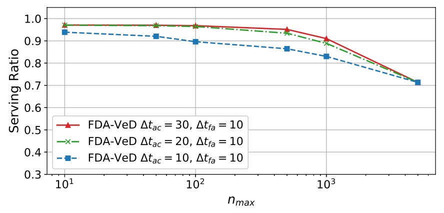


Figure 4. Passenger serving ratio and Number of requests by hour in Manhattan Island (20110112-20110118)



(a) R against different number of vehicles



(b) R against different size of subarea

Figure 5. Passenger serving ratio in Manhattan Island (20110112-20110118)

Figure 4 shows the serving ratio R by hour for OBM−VeD and FDA−VeD with different parameters. It is clear that the FDA−VeD outperforms OBM−VeD significantly in terms of serving ratio. R in seven days for OBM−VeD is 0.71, and the FDA−VeD achieves 0.89, 0.96, 0.97 with Δt_{ac} as 10, 20, 30 minutes respectively. Using a 10 minutes advanced time ($\Delta t_{ac} = 10$) to relocate idles vehicles for the future time interval $(t_a, t_f]$, R can be improved by 18% compared to OBM−VeD. When Δt_{ac} is increased from 10 to 20, R can be further improved significantly, results 25% improvement compared to OBM−VeD. By increasing Δt_{ac} from 20 to 30, the performance sees further improvement, but not significant (1%).

1) Impact of Fleet Size: The fleet size is an important factor that significantly influences the serving ratio. Here we fixed the subarea size and changed the fleet size. Figure 5(a) shows the serving ratio with different numbers of vehicle. When $|\mathcal{V}|$ increases from 2000 to 10000, the FDA−VeD system always performs better than OBM−VeD. When $|\mathcal{V}|$ is small (e.g. 2000) or large (e.g. 10000), the improvement of the FDA−VeD are relatively small compared with $|\mathcal{V}| = 6000$. When the $|\mathcal{V}|$ is very small, there are not so many vehicles could be used to relocate because many subareas are undersupplied. When the $|\mathcal{V}|$ is very large, OBM−VeD has a high serving ratio; thus, the improvement of FDA−VeD is relatively small.

Table I
Operating Cost

Item	OBM−VeD	FDA−VeD $\Delta t_{fa}=10$		
		$\Delta t_{ac}=10$	$\Delta t_{ac}=20$	$\Delta t_{ac}=30$
d_1	145.8	185.7	203.4	204.4
d_2	44.9	54.5	58.0	58.3
d_3	0	6.0	19.7	24.9
d_{all}	190.7	246.2	281.1	287.6
r_1	0.76	0.75	0.72	0.71
d_g	-	39.9	57.6	58.6
d_c	0	6.0	19.7	24.9
r_{gc}	-	6.64	2.93	2.36

Figure 5(a) shows that no matter how $|\mathcal{V}|$ changes, the FDA−VeD with $\Delta t_{ac} = 20$ and $\Delta t_{ac} = 20$ always have nearly the same serving ratio. It means 20 minutes is enough for Manhattan Island to relocate all idle vehicles to the places that need them, while 10 minutes is not.

2) Impact of Subarea Size: The size of subarea is decided by the parameter n_{max} and will affect the subarea divisions. Different values of n_{max} are tested, including 10, 50, 100, 500, 1000, 5000. When $n_{max} = 5000$, as there are only 5085 points in the road graph, the whole serving area would be divide into only one subarea and the relative FDA−VeD system is equivalent to that of OBM−VeD.

Figure 5(b) shows that by increasing subarea size n_{max}, the serving ratio has a downtrend. When $n_{max} = 1000$, the whole area is divided into 5 areas and there is no big ratio improvement for FDA−VeD with $\Delta t_{ac} = 10, 20$. When $n_{max} = 10$, a relatively small value, there is considerable improvement for all three Δt_{ac} values.

3) Operating Cost: Compared to OBM−VeD, the FDA−VeD system will increases the operating cost, as it needs to relocate idle vehicles. There are two metrics to evaluate the operating cost: with-passenger ratio r_1 and gain–cost ratio r_{gc}. Table I shows the average vehicle's travel distance per day and the metrics. When $\Delta t_{ac} = 30$, the with-passenger ratio r_1 decreases from 0.76 to 0.71 (5% decrease), while with-passenger distance d_1 increases from 145.8 to 204.4 (40.1% increase) and $r_{gc} = 2.36$, that means every 1km relocating distance can bring 2.36 km with-passenger distance. When $\Delta t_{ac} = 10$, r_1 only decreased by 1%, while d_1 increased by 27.4% and every

1km relocating distance can bring 6.64km with-passenger distance. Compared with the increase of serving ratio, `FDA-VeD` system operating cost's increase is very low.

V. Related Work

To handle dispatching task with large number of requests and vehicles, Vazifeh et al. [2] proposed a scalable solution to dispatch vehicle for a batch of real-time requests and achieved the maximum matching of available vehicles and real-time requests. Zhang et al. [12] proposed an combinatorial optimization model to dispatch vehicles for real-time travel requests. However, only real-time requests are used to dispatch vehicles which can lead to idle vehicles in low demand areas.

There exist some works that consider the potential future demand in vehicle dispatching systems [3], [13], [14]. Lowalekar et al. [3] used potential future demands to select which requests to serve. The request which destination has a higher probability of being requested in the future will be served first. Xu et al. [13] modelled vehicles dispatching as a sequential decision-making problem and design algorithm considering both the instant reward and future state-value to optimize long-term global efficiency. Cheng et al. [14] proposed a queueing-based vehicle dispatching framework, which first predicts the future travel demand of each region, then estimates the idle time periods of vehicles through a queueing model for each region. Then they used a batch-based vehicle dispatching algorithm to assign suitable vehicles to requests. However, these works are passive vehicle dispatching solution, which means they only dispatch vehicles for received requests and do not proactively relocate idle vehicles before received requests. If an area has no request at present but demands in the future, their model can not dispatch vehicles for this area in advance.

The idle-vehicle relocation strategies proactively relocate idle vehicles to high demand area and improve the serving ratio by improving the utilization rate of vehicles [4]–[6]. Alonso-Mora et al. [4], [5] propose a framework to dispatch vehicles and relocate idle vehicles for online ride-sharing. In their work, idle vehicles are relocated in every 30 seconds, which is a very short period, and can increase relocation costs significantly. Sayarshad et al. [6] propose a queuing-based formulation to solve the idle-vehicle relocation problem for on-demand mobility services. They made idle-vehicle relocation in a low frequency, every 15 minutes. However, instead of developing an effective technique to divide the serving area, they simply used the already-defined boundaries (e.g., suburb boundary) for this. Also, they only consider the pickup location of requests, without taking into account their drop-off locations. This means that new idle vehicles need to be relocated into the subareas.

VI. Conclusion

In this paper, we developed a vehicle dispatching system called `FDA-VeD` for on-demand urban mobility. The system maintains a balance between the demand and supply in different subareas by considering the relocation of idle vehicles based on their predicted future demands. By considering both the pickup and drop-off demand for each area, we could relocate idle vehicles for a long future time interval and low frequency. Extensive experiments show that with a small operating cost, `FDA-VeD` significantly helps in achieving a high passenger serving ratio.

References

[1] U. Ritzinger, J. Puchinger, and R. F. Hartl, "A survey on dynamic and stochastic vehicle routing problems," *Int. J. Prod. Res.*, vol. 54, no. 1, pp. 215–231, 2016.

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

[3] M. Lowalekar, P. Varakantham, and P. Jaillet, "Online spatio-temporal matching in stochastic and dynamic domains," *Artificial Intelligence*, vol. 261, pp. 71–112, 2018.

[4] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment." *PNAS*, vol. 114, no. 3, pp. 462–467, 2017.

[5] A. Wallar, M. V. D. Zee, J. Alonso-Mora, and D. Rus, "Vehicle rebalancing for mobility-on-demand systems with ride-sharing," in *IEEE/RSJ IROS*, 2018, pp. 4539–4546.

[6] H. R. Sayarshad and J. Y. J. Chow, "Non-myopic relocation of idle mobility-on-demand vehicles as a dynamic location-allocation-queueing problem," *Transp. Res. Part E: Logist. Transp. Rev.*, vol. 106, pp. 60–77, 2017.

[7] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti, "Quantifying the benefits of vehicle pooling with shareability networks," *PNAS*, vol. 111, no. 37, pp. 13 290–13 294, 2014.

[8] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li, "Deep multi-view spatial-temporal network for taxi demand prediction," in *AAAI*, 2018.

[9] K. Zhang, Z. Feng, S. Chen, K. Huang, and G. Wang, "A framework for passengers demand prediction and recommendation," in *SCC*. IEEE, 2016, pp. 340–347.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *JMLR*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[11] J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM Journal on Computing*, vol. 2, no. 4, pp. 225–231, 1973.

[12] L. Zhang, T. Hu, Y. Min, G. Wu, J. Zhang, P. Feng, P. Gong, and J. Ye, "A taxi order dispatch model based on combinatorial optimization," in *SIGKDD*. ACM, 2017, pp. 2151–2159.

[13] Z. Xu, Z. Li, Q. Guan, D. Zhang, Q. Li, J. Nan, C. Liu, W. Bian, and J. Ye, "Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach," in *SIGKDD*. ACM, 2018, pp. 905–913.

[14] P. Cheng, C. Feng, L. Chen, and Z. Wang, "A queueing-theoretic framework for vehicle dispatching in dynamic car-hailing," in *ICDE*. IEEE, 2019, pp. 1622–1625.