Network-Flow-Based Efficient Vehicle Dispatch for City-Scale Ride-Hailing Systems

Yuhang Xu, Wanyuan Wang, Guangwei Xiong, Xiang Liu, Weiwei Wu, *Member, IEEE*, and Kai Liu, *Senior Member, IEEE*

***Abstract*— Ride-hailing systems (RHSs) provide passengers with convenient and flexible mobility services, have played an important role in modern urban transportation. With the limited vehicles, RHSs wish to optimize the dispatch of vehicles to requests with the objective of serving as many requests as possible. To address such a city-scale vehicle dispatch problem with thousands of vehicles and requests in each epoch, existing algorithms always take a tradeoff between effectiveness (i.e., real-time) and efficiency (i.e., service rate), such as ignoring future demands to guarantee real-time or solving a complex combinatorial optimization to improve service rate. To guarantee the service rate in a real-time fashion, this paper proposes two novel network flow-based vehicle dispatch algorithms. A network flow-based algorithm (NFBA) is provided to deal with offline scenarios. By constructing the vehicle-shareability network, a min-cost flow is built to find the optimal dispatch of vehicles to requests. To improve the request service rate in real-time, an efficient multi-sample multi-network flow-based algorithm (MNFBA) is proposed for the online scenarios. Each min-cost flow is utilized for a sample of future requests, and online vehicle dispatch policy is averaged over these flows. Extensive simulations based on real-world trip datasets in New York City are conducted. The experimental results show that compared to the benchmarks, our proposed algorithm can generate the dispatch of vehicles to requests within seconds, but can greatly increase the daily request service rate.**

***Index Terms*— Urban computing, ride-hailing systems, network flow, scalable algorithm.**

I. Introduction

AS ONE of the representative business models of the sharing economy, ride-hailing enables individuals with idle vehicles to provide mobility services to others in need [1]. In 2018, the global online ride-hailing market size was $24.4 billion, and it is likely to reach $1036 billion by the end of 2025, with a compound annual growth rate of 19.8% during 2019-2025. Ride-hailing systems (RHSs) such as Didi, Uber, and Lyft have been an indispensable component of the transportation infrastructure [2]. According to Uber statistics, by April 2019, over 75 million passengers, and more than one-third of Americans use ride-hailing services.

With the increasing number of passengers, it becomes more and more difficult for passengers to achieve a quick service, especially during peak hours. Moreover, for the online RHSs, both of the current requests and future requests should be taken into consideration. Serving the combinational current and future requests optimally is complicated and time-consuming, especially in the city-scale RHSs with thousands of vehicles and requests in each epoch. Therefore, how to optimize the request service rate (i.e., efficiency) within the whole horizon in a real-time manner (i.e., effectiveness) is the main focus of online RHSs.

Existing vehicle dispatching algorithms always make a tradeoff between effectiveness and efficiency (Detail review is given in section 2). For example, to achieve the real-time dispatching, bipartite graph $G(U, V, E)$ is usually constructed, where U represents the idle vehicles, V represents the requests in each epoch, and the weighted edge E might represent the distance or price between U and V [1], [3]–[10]. Effective matching algorithms can be used to maximize/minimize the weights such that local optimal dispatching is achieved [4], [11], [12]. Since the future requests are ignored, these real-time algorithms might be myopic. By taking the future requests into consideration, combinatorial optimization schemes are widely utilized [13]–[15]. Based on the predicted arrivals of vehicle and request, a global offline optimization is generated periodically, which can be used to guide online matching [13], [14]. On the other hand, Lowalekar *et al.* [15] use the historical data to generate several samples of future requests, and an integer programming model is used to optimize these combinatorial sample requests. Due to the limitation of prediction accuracy or sample size, the efficiency improvement of these methods is limited. An extended multi-stage stochastic programming model is introduced in [16], which takes into account the requests of multi stages in the future. Although this algorithm has been validated to be efficient in improving service rate but its high computational cost is unable to meet the real-time requirement, especially in the city-scale RHSs.

Against this background, this paper proposes a novel network-flow-based vehicle dispatch approach for effective and efficient RHSs. We first propose a min-cost flow network

Manuscript received August 29, 2020; revised December 22, 2020; accepted January 15, 2021. Date of publication February 5, 2021; date of current version May 31, 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102200, in part by the Natural Science Foundation of China under Grant 61632008, Grant 61902062, Grant 61672154, Grant 61972086, Grant 61932007, Grant 61806053, and Grant 61807008, in part by NSFC under Grant 11771365, in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20180356 and Grant BK20180369, in part by the Postgraduate Research and Practice Innovation Program of Jiangsu Province of China under Grant KYCX19_0089, and in part by the Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education. The Associate Editor for this article was J. Blum. *(Yuhang Xu and Wanyuan Wang contributed equally to this work.) (Corresponding author: Guangwei Xiong and Xiang Liu.)*

Yuhang Xu, Wanyuan Wang, Guangwei Xiong, Xiang Liu, and Weiwei Wu are with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China (e-mail: yuhang_xu@seu.edu.cn; wywang@seu.edu.cn; guangweixiong@seu.edu.cn; xiangliu@seu.edu.cn; weiweiwu@seu.edu.cn).

Kai Liu is with the School of Computer Science, Chongqing University, Chongqing 400030, China (e-mail: liukai0807@gmail.com).

Digital Object Identifier 10.1109/TITS.2021.3054893

1558-0016 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on December 29,2024 at 04:35:46 UTC from IEEE Xplore. Restrictions apply.

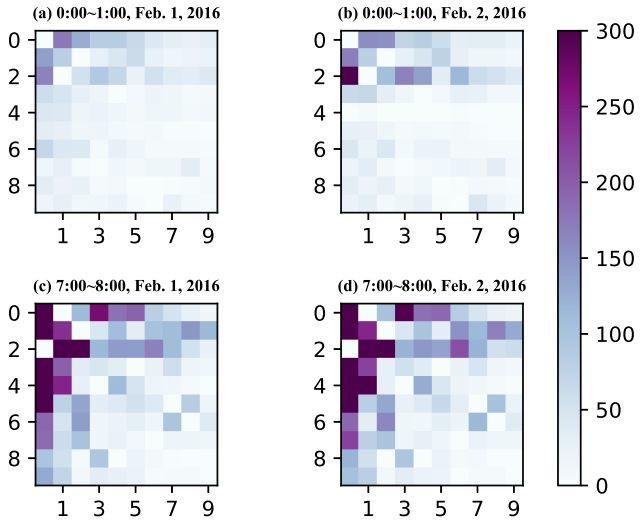


Fig. 1. A snapshot of historical data from New York City on the number of requests between 10 popular pick-up and drop-off zones. The gray scale reflects different number of requests, which indicates that the number of requests in the same period on adjacent days is close.

to address the offline city-scale vehicle dispatch problem, where requests are known in prior. Inspired by this insight, we propose a multi-network flow-based algorithm for the online vehicle dispatch problem. Being aware of that passenger requests follow similar distributions in regular days (as shown in Figure 1), we consider taking multiple samples of future requests from the historical data. For each of these samples, we exploit the time-effective min-cost flow mechanism to achieve the dispatch of vehicles to requests. The online real-time vehicle dispatch solution is achieved by averaging the flow over multiple samples.

The contributions of this paper can be summarized as follows.

- In offline scenarios, a min-cost network flow-based algorithm is proposed to find the optimal dispatch solution.
- In online scenarios, an efficient multi-network flow-based algorithm is designed by averaging the min-cost flows over multiple samples. This algorithm can generate a vehicle dispatch solution in real time while significantly improving the request service rate.
- Extensive experiments based on a real trip dataset from New York City are conducted to show the effectiveness of the proposed algorithm. The simulation results show that our algorithm outperforms the benchmarks in terms of the request service rate, the accumulated driver income, and the average running time.

The remainder of the paper is structured as follows. Section 2 reviews the related research work. Section 3 introduces the system model and formulates the vehicle dispatch problem. Section 4 addresses the offline vehicle dispatch problem. Section 5 deals with the online vehicle dispatch problem. The experiment results are presented in Section 6. Section 7 concludes this paper.

II. Related Work

Vehicle dispatch and demand forecasting are two research topics that are relevant to our work. In this section, we first categorize the related work on vehicle dispatch in ride-hailing systems into three types: 1) dispatch for individual request, 2) dispatch for batch requests, and 3) dispatch for batch requests considering future demand. Fig. 2 depicts the classifications of vehicle dispatch in ride-hailing systems. Then, we briefly review the techniques that address passenger demand forecasting.

A. Vehicle Dispatch

Vehicle dispatching or order matching is a well-studied research topic in recent years. For online vehicle dispatch, passengers, drivers and platforms have different points of interest, including passenger waiting time, vehicle usage efficiency, platform revenue, etc. Thus, a large number of online vehicle dispatch algorithms with different optimization goals have been brought up.

1) Dispatch for Individual Request: In the early stage, simple vehicle dispatch strategies are commonly employed by ride-hailing systems to meet online requirements and reduce passenger waiting time. In [11], [12], [17], [18], a greedy mechanism is proposed in which requests are assigned to the nearest serviceable vehicles in a first-in, first-out mode. With the advancement of technology, the criteria used to find the nearest vehicle has changed from the original straight line distance, Manhattan distance, to real-time distance information based on the road network structure and traffic conditions [17], [18]. The greedy mechanism is a system-assigning vehicle dispatch approach that tends to benefit passengers. Meanwhile, driver-grabbing is another kind of dispatch approach for individual request which brings more benefits to drivers. In this approach, the ride-hailing system broadcasts the passenger request to nearby drivers, lets the drivers make decisions and the driver who sends back his acceptance decision first would be dispatched to serve the request. Recently, Duan *et al.* [19] combine the two existing approaches, jointly consider passenger's and driver's interests, and introduce a dynamic programming algorithm to optimally solve the increase ratio of the size of the dispatch region. All the above vehicle dispatch approaches for an individual request can effectively reduce the pickup distance, guarantee low computational complexity, and keep a short dispatching time. However, optimization work at an individual request level often sacrifices the global optimality to get a suboptimal solution quickly that can easily lead to a mismatch between supply and demand.

2) Dispatch for Batch Requests: Aware of the problems with individual request assignment, some works have attempted to conduct stage global dispatch optimization for requests within a time interval, which are classified as batch requests dispatch methods. To find a globally optimal solution, the bipartite matching model is widely used in [1], [3]–[10]. The vehicle dispatch problem for batch requests can be represented by a bipartite graph $G(U, V, E)$, where U represents the idle vehicles, V represents the requests

Authorized licensed use limited to: ShanghaiTech University. Downloaded on December 29,2024 at 04:35:46 UTC from IEEE Xplore. Restrictions apply.

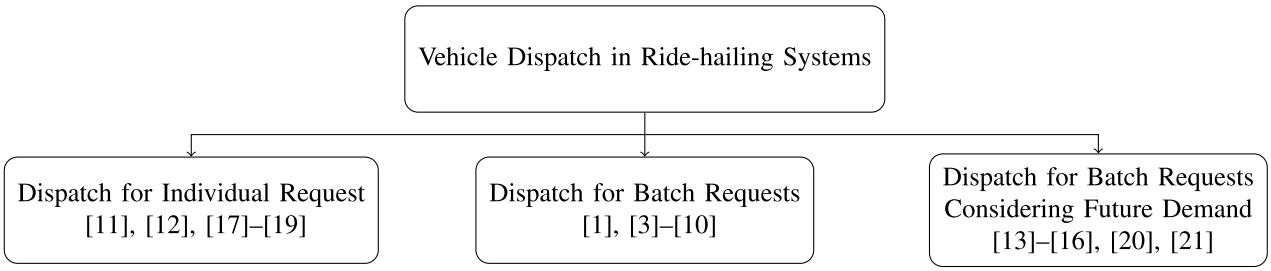


Fig. 2. Classification of vehicle dispatch researches in ride-hailing systems.

and E represents the possible vehicle-request match set. In [4], [5], the optimization goal is to fulfill as many requests as possible, thus the dispatch problem is actually a maximum bipartite matching problem. With a simple graph translation, the original problem can be solved by using the max-flow algorithm. More studies choose to assign different weights to the edges in E and the dispatch problem is translated into a minimum weight maximum bipartite matching problem. Depending on the optimization objective, the edge weight can be calculated based on different metrics such as the pickup distance [4], the trip revenue [7], or more complex combinations of metrics. Furthermore, considering the different preferences of drivers and passengers, researchers in [8], [9] pay their attention to the classic stable matching problem, and different heuristic algorithms are provided to achieve a stable passenger-driver matching in ride-hailing systems. Recently, Some other researchers have tried to get a breakthrough in the theory. Dickerson *et al.* [10] propose a novel model for online matching with reusable resources (e.g. vehicles) and present an LP-based adaptive algorithm that achieves an online competitive ratio of $\frac{1}{2} - \epsilon$. However, In their model, each vehicle has a fixed position and needs to come back to its original position before it can be matched again which is an invalid assumption in practice. Zhao *et al.* [1] define a new model OSM-KIID for preference-aware task assignment in on-demand taxi dispatching. To maximize the expected total profits (the main objective) and minimize the expected total number of blocking edges (unstable matching edges) simultaneously, a linear programming based online algorithm is presented which achieves an online ratio of at least $1 - \frac{1}{e}$ and has at most $0.6 \cdot |E|$ blocking edges expectedly. The above algorithms can derive the optimal vehicle dispatch solution within a stage, which improves the efficiency of the vehicles or the revenue of the platform. However, the limitation is that they all make decisions without considering future demand that may lead to the supply-demand mismatch in the long run.

3) Dispatch for Batch Requests Considering Future Demand: To eliminate this limitation and provide sustainable services for passengers, some researchers begin to take future demand into account when designing dispatch algorithms. To minimize taxis' total idle mileage and maintain service fairness across the whole city, a robust taxi dispatch optimization model under spatial-temporal correlated uncertainties of predicted demand is developed in [20], [21]. Meanwhile, a data-driven algorithm is designed to construct uncertainty sets that provide the desired level of probabilistic guarantee for the robust taxi dispatch solutions. Unlike the above work, more studies consider the impact of current allocations on the future by projecting the amount of demand in the next stage. To increase the total number of assigned vehicle-request pairs, algorithms in [13], [14] perform a global optimization offline periodically based on the prediction of vehicles and requests, and use the offline results to guide real-time supply and demand matching. Lowalekar *et al.* [15] first use the historical data to generate several samples of future requests, and an integer programming model is used to optimize these combinatorial sample requests. The methods introduced in [13]–[15] can be classified as two-stage algorithms.

Due to the limitation of prediction accuracy or sample length, these two-stage algorithms can only slightly improve system efficiency. Then, in [16], the stochastic programming model introduced in [15] is extended to multiple stages that takes into account the requests of multi-stages in the future. Given the complexity of solving multi-stage stochastic planning models, the work has to make a trade-off between optimization efficiency and computation time. To meet the online time requirement, the Benders decomposition method is used to get a suboptimal solution quickly while sacrificing efficiency. Similar to the work in [15], [16], we also take samples from historical data as possible future requests. However, different from the above work, an efficient multi-network flow-based algorithm is proposed to generate a vehicle dispatch solution. Due to the highly scalable multi-network flow model, our algorithm can take into account future demand for a longer period while ensuring real-time performance. It is worth noting that looking ahead for a longer period does help to improve the efficiency of vehicle dispatch. In section 6, extensive experiments are conducted to prove this nature.

B. Demand Forecasting

Demand forecasting is an option when future demand needs to be considered. It is no doubt that demand forecasting is a tricky task because the intrinsic uncertainties of future demand result from many factors, such as weather, holiday and city events, etc. Therefore, It is difficult for predictive models to accurately learn the spatial-temporal patterns in demand. However, the good news is that as technology continues to evolve, the accuracy of demand forecasting is improving. The traditional method of demand forecasting is the time series prediction. In [22], a novel methodology based on the auto-regressive integrated moving average (ARIMA) is

Authorized licensed use limited to: ShanghaiTech University. Downloaded on December 29,2024 at 04:35:46 UTC from IEEE Xplore. Restrictions apply.

TABLE I
NOTATIONS

Notation	Explanation
$\mathcal{Z}$	Zone set, $\lvert\mathcal{Z}\rvert = l$
$\mathcal{A}$	Vehicle set, $\lvert\mathcal{A}\rvert = m$
$\mathcal{R}$	Request set, $\lvert\mathcal{R}\rvert = n$
$\mathcal{T}$	Epoch set, $\lvert\mathcal{T}\rvert = T$
V^+	Candidate vertex set of starting point
L	Number of epochs in each sample
N	Number of samples
W	The time distance matrix
P	Min-cost flow solution
J	Vehicle dispatch Solution
U	Number of covered vertices
S	Solution matrix, S_i stores the solution to the min-cost flow problem defined on the i-th sample
H	Cumulative matrix, $H_{i,j}$ is the number of type (z_i, z_j) requests fulfilled in the returns of sub-problems
I	Probability matrix, $I_{i,j}$ is the possibility that a vehicle in zone z_i is dispatched to zone z_j
θ	Length of one epoch
$\alpha_{i,j,t}$	Number of requests from zone z_i to z_j in epoch t
β_i	Number of idle vehicles in zone z_i
$\gamma_{i,t}$	Number of vehicles that will reach zone z_i in epoch t

introduced for predicting the spatial distribution of taxi-passengers for a short-term time horizon using streaming data. Currently, with the popularity of neural networks, many studies have tried to apply neural networks to demand prediction. In [23], a sequence learning model based on recurrent neural networks and mixture density networks is proposed to predict taxi demand in different areas of a city. In [24], a temporal guided network (TGNet) based on fully convolutional networks and temporal guided embedding, is proposed to capture spatiotemporal patterns. In [25], a reinforcement learning framework is designed to improve the passenger demand prediction via learning the salient spatial-temporal dynamics. Considering that current demand forecasting techniques cannot get sufficiently accurate results at a fine-granularity, our algorithm directly gets the distribution of future demand by sampling from historical data. However, in our experiments, we also design two baseline algorithms based on the forecasting model introduced in [22] and [24].

III. MODEL

In this section, we introduce the RHSs and formulate the vehicle dispatch problem. Notations used throughout this paper are summarized in Table I.

Ride-hailing Systems (RHSs). The time of the daily RHSs is discretized into T decision epochs. Let $\mathcal{T} = \{1, \ldots, T\}$ be the set of epochs, and each epoch includes θ minutes such that the passenger will not wait too long. After the start of each epoch, each passenger who needs the service sends a request, and RHSs decide which requests to accept at the end of each epoch.

Zone. Let $\mathcal{Z} = \{z_1, z_2, \ldots, z_l\}$ be the set of zones in a city. Let $w_{i,j} \in \mathbb{Z}$ denote the time distance between zones z_i and z_j, which indicates that a vehicle needs to take $w_{i,j}$ epochs to move from zone z_i to z_j.

Vehicle. Let $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ be the set of vehicles. Each vehicle a_i is parameterized by an initial zone $z_{a_i}^{init}$. We focus on the ride-hailing scenarios, where each vehicle can serve only one request at a time. To reduce the pickup distance of drivers and the waiting time of the passengers, we assume that the vehicle can only be dispatched to requests that lie in the same zone.

Request. Let $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ be the set of requests. Each request r_i is represented by a tuple $\langle t_i^p, t_i^d, z_i^p, z_i^d \rangle$, where t_i^p denotes the release epoch, t_i^d denotes the drop-off epoch, z_i^p denotes the pickup zone and z_i^d denotes the drop-off zone. The request must be accepted or rejected within its release epoch. Throughout this paper, we use the request, passenger and demand interchangeably.

We call two requests r_i and r_j *consecutive requests* if it is feasible for a vehicle to pick up the request r_j immediately after serving the request r_i. Consecutive requests r_i and r_j need to meet the spatial-temporal constraints, i.e., 1) the drop-off zone of r_i and the pick-up zone of r_j should lie in the same zone, i.e., $z_i^d = z_j^p$; 2) the release epoch of later request r_j should not be earlier than the drop-off epoch of the former request r_i, i.e., $t_i^d + w_{z_i^d, z_j^p} \leq t_j^p$. A *consecutive path* is a sequence of consecutive requests $\langle r_1, r_2, \cdots, r_g \rangle$, where continuous requests r_j and r_{j+1} $(1 \leq j \leq g)$ are consecutive. Such a consecutive path can be served by a single vehicle.

A. Problem Formulation

Let $x_{i,j,t} \in \{0, 1\}$ represents whether vehicle a_i is dispatched to serve the request r_j in epoch t. That is, if vehicle a_i is dispatched to serve request r_j in epoch t, then $x_{i,j,t} = 1$ and otherwise $x_{i,j,t} = 0$. Let $y_{i,k,t} \in \{0, 1\}$ represents whether vehicle a_i is idle in zone z_k in epoch t, which means if vehicle a_i is idle in zone z_k in epoch t, $y_{i,k,t} = 1$ and otherwise $y_{i,k,t} = 0$.

The objective of the vehicle dispatch problem is to maximize the number of served requests within the horizon T, i.e.,

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{t=1}^{T} x_{i,j,t} \tag{1}$$

Each vehicle can only serve at most one request in one epoch, we have

$$\sum_{j=1}^{n} x_{i,j,t} \leq 1, \quad \forall 1 \leq i \leq m, \ 1 \leq t \leq T. \tag{2}$$

Each request can only be accepted at most once in its release epoch, we have

$$\begin{aligned} &x_{i,j,t} = 0, \quad \forall 1 \leq i \leq m, \ 1 \leq j \leq n, \ t \neq t_j^p; \\ &x_{i,j,t} \in \{0, 1\}, \quad \forall 1 \leq i \leq m, \ 1 \leq j \leq n, \ t = t_j^p; \\ &\sum_{i=1}^{m} x_{i,j,t} \leq 1, \quad \forall 1 \leq j \leq n, \ t = t_j^p. \end{aligned} \tag{3}$$

At the beginning, all vehicles are idle in their initial zones and each vehicle can be idle in one zone in each epoch. That

Authorized licensed use limited to: ShanghaiTech University. Downloaded on December 29,2024 at 04:35:46 UTC from IEEE Xplore. Restrictions apply.

is,

$$y_{i,k,1} = \begin{cases} 1, & \forall 1 \le i \le m,\ k = z_{a_i}^{init}; \\ 0, & \forall 1 \le i \le m,\ k \ne z_{a_i}^{init}. \end{cases}$$

$$\sum_{k=1}^{l} y_{i,k,t} \le 1, \quad \forall 1 \le i \le m,\ 1 \le t \le T. \tag{4}$$

Because the vehicle can only be dispatched to serve the requests that lie in the same zone, a_i must be idle in zone z_j^p in epoch t_j^p if vehicle a_i is dispatched to serve request r_j in epoch t_j^p.

$$x_{i,j,t_j^p} \le y_{i,z_j^p,t_j^p}, \quad \forall 1 \le i \le m,\ 1 \le j \le n. \tag{5}$$

If vehicle a_i is dispatched to serve request r_j and $r_{j'}$ in epoch t and $t'(> t)$ respectively, there must be a long enough period between t and t' such that a_i can complete request r_j before serving request $r_{j'}$. Thus, we have

$$t + w_{z_j^p, z_j^d} - t' \le M \cdot (2 - x_{i,j,t} - x_{i,j',t'}), \quad \forall 1 \le i \le m,\ 1 \le j,\ j' \le n,\ t < t'. \tag{6}$$

where $M \in \mathbb{Z}$ is a large constant to guarantee the time constraint.

For a vehicle that serves the request r_j at the epoch t_j^p, it must be 1) busy in next $w_{z_j^p z_j^d}$ epochs, and 2) idle again at the epoch t_j^d within the zone z_j^d. Thus, we have

$$y_{i,k,t'} \le 1 - x_{i,j,t}, \quad \forall i, j, k, t < t' < t + w_{z_j^p z_j^d};$$
$$y_{i,z_j^d,t'} \ge x_{i,j,t}, \quad \forall i, j, t, t' = t_j^d. \tag{7}$$

Therefore, the vehicle dispatch problem is summarized in the following definition (IP formulation).

Definition 1 (Vehicle Dispatch Problem): The Vehicle Dispatch problem is to determine the dispatch of vehicles to maximize the number of served requests (1) with the satisfaction of the aforementioned constraints (2)-(7).

In this paper, we consider both offline and online vehicle dispatch problems. In the offline setting, we have complete information about the requests within the planning horizon. In the online setting, we only have information about the requests in the current epoch.

IV. Algorithm for Offline Vehicle Dispatch

In this section, we address the offline vehicle dispatch problem which is meaningful because it could help the RHS administrators to understand the upper bound of their fleet's service capacity. We provide an optimal algorithm NFBA to compute the optimal vehicle dispatch solution for all vehicles. The idea is to construct a vehicle-shareability network and transform the original problem into an equivalent min-cost flow problem.

In the first step, we construct a vehicle-shareability network introduced in [5] to represent the relationship between requests first. The vehicle-shareability network is a directed acyclic graph $G = (V, E)$ in which each vertex $v \in V$ represents a request and each directed edge $e \in E$ connects two consecutive requests. Figure 3 presents an example of the vehicle-shareability network.

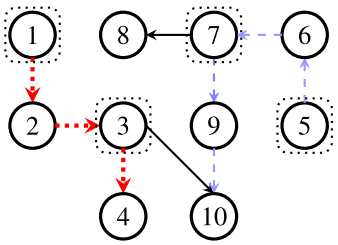


Fig. 3. A tiny example of a vehicle-shareability network. Each vertex represents a request and each directed edge indicates the consecutive relationship between the two requests it is connected to. The dotted vertex indicates that there is a vehicle in the pick-up zone of the request. Initially, there are two available vehicles, one in the pick-up zone of requests 1 and 3, and the other in the pick-up zone of requests 5 and 7. Thus, only request 1, 3, 5, 7 might be the starting point of a legal path. The red dotted path (1→2→3→4) and blue dashed path (5→6→7→9→10) are disjoint and can serve 9 requests in total, which gives an optimal dispatch solution in this case.

Each path in the graph is a potential pick-up sequence for one vehicle. Since a request can only be served at most once, the dispatch solution of m vehicles would correspond to m disjoint paths in the vehicle-shareability network. Considering the initial zones of the vehicles, only a part of the vertices on the vehicle-shareability network can be the candidate starting points of legal paths. We denote the set of vertices in this type by V^+. For example, in Figure 3, the dotted vertices are the candidate starting points of a legal path. Meanwhile, due to the limitation of the number of vehicles, the number of paths starting from the requests in zone z_i cannot exceed the number of initially available vehicles in zone z_i. As shown in the example in Figure 3, since there is only one vehicle in the pick-up zone of requests 5 and 7, we cannot choose two paths starting from 5 and 7 simultaneously. Therefore, the vehicle dispatch problem can be transformed into a novel m-paths coverage problem under special constraints on the vehicle-shareability network, which is defined as follows:

Definition 2 (NmPC): Given a directed acyclic graph $G = (V, E)$ and a candidate vertex set of starting point V^+, where vertices in V^+ belong to K subsets $\{\mathcal{S}_1, \cdots, \mathcal{S}_K\}$. There is no intersection between any two subsets (i.e., $\mathcal{S}_i \cup \mathcal{S}_j = \emptyset, \forall i, j \le K$) and each subset has a budget B_i which indicates the maximum number of selected paths starting from the vertices in subset $\mathcal{S}_i$. The objective of the novel m-paths coverage problem is to find m disjoint paths starting from the vertices in V^+ that satisfy the budget constraint while covering the maximum number of vertices.

In the second step, based on the vehicle-shareability network, we construct a new request-link graph $G' = (V', E')$ by adding the vehicle information and define a min-cost flow problem on it. The construction process is as follows.

- Create a source vertex s with demand $-m$ and a sink vertex s' with demand m in graph G'.
- For each vehicle a_i, create a vertex a_i with demand 0 and create a directed edge from vertex s to vertex a_i with capacity 1 and weight 0.
- Create an *in* vertex v_{in}^i, an *out* vertex v_{out}^i for each vertex $v_i \in V$ and a directed edge from vertex v_{in}^i to v_{out}^i with capacity 1 and weight 0 in graph G'.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on December 29,2024 at 04:35:46 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 Network Flow-Based Algorithm (NFBA)

Input:
The vehicle set, $\mathcal{A}$; the request set, $\mathcal{R}$;
Output:
The vehicle dispatch solution, J; number of satisfied requests, U;

1: Construct the vehicle-shareability network G;
2: Construct the corresponding request-link graph G' and define a min-cost flow problem on it;
3: Solve the min-cost flow problem defined on G' and get its solution P;
4: Transform the solution P to be an optimal solution J for the NmPC problem and count the number of satisfied requests U;
5: **return** solution J and the number of satisfied requests U.

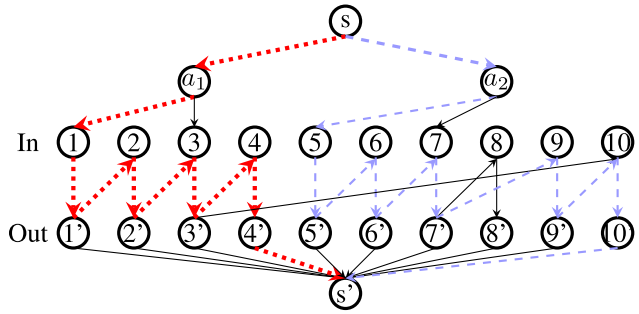


Fig. 4. The corresponding request-link graph constructed from the vehicle-shareability network shown in Figure 3. The red dotted and blue dashed flows in this graph correspond to the red dotted and blue dashed paths in Figure 3, respectively.

- Check each vehicle and request pair (a_i, v_j), if the initial zone of the vehicle a_i and the pick-up zone of the request v_j are the same, we create a direct edge from vertex a_i to vertex v_{in}^j with capacity 1 and weight 0.
- For each edge $e(v_i, v_j) \in E$, add a directed edge from vertex v_{out}^i to vertex v_{in}^j with capacity 1 and weight -1 to the graph G'.
- For each *out* vertex, create a directed edge from it to the sink vertex s' with capacity 1 and weight 0.

Figure 4 shows the corresponding request-link graph generated from the vehicle-shareability network in Figure 3.

Then, we show that the optimal solution to the NmPC problem can be obtained by computing the min-cost flow on the request-link graph.

Theorem 1: By solving the min-cost flow problem defined on the constructed request-link graph, we can obtain an optimal solution to the NmPC problem defined on the vehicle-shareability network.

Proof: Suppose P is the min-cost flow in graph G' which can sent an amount of flow m from vertex s to vertex s' with the minimum cost of ω. Since the capacity of all edges in graph G' is unit, the min-cost flow P is consisting of m disjoint unit flows. We denote these m flows as $p_1, p_2, \ldots, p_m$ and assume each flow has a cost ω_i. Since the weight of the edges is negative, both ω and ω_i are negative. Each flow p_i can be transformed into a path in G. For example, the flow $\{s, a_1, v_{in}^1, v_{out}^1, v_{in}^2, v_{out}^2, v_{in}^3, v_{out}^3, v_{in}^4, v_{out}^4, s'\}$ in G' can be transformed to the path $\{v_1, v_2, v_3, v_4\}$ in G. For each flow p_i with cost ω_i, its corresponding path in graph G covers $-\omega_i$ edges and $1-\omega_i$ vertices. Therefore, the m paths in G corresponding to these m flows can cover $\sum_{i=1}^{m}(1-\omega_i) = m-\omega$ vertices in total and we denote this coverage solution as J. Then, we prove that the solution J meets the staring point constraint and the vehicle number constraint. Since each vehicle vertex only connects to the request vertices in its initial zone in G', the starting points of the paths must belong to the set of candidate starting points V^+. Meanwhile, since the flow into each vehicle vertex is 1 in G', it can only choose one of its successor request vertices to pass the flow. Thus, vehicle number constraint is guaranteed. Suppose there is another coverage solution J' in G which covers η vertices where $\eta > m-\omega$. These m paths would cover η vertices and $\eta - m$ edges simultaneously. Since the weight of each edge in G' is -1, there would be another flow P' which can sent an amount of flow m with cost $-(\eta - m) < \omega$. Thus, P' is cheaper than P which implies P is not the min-cost flow. This conflicts with our hypothesis. Hence, J is the optimal solution to the NmPC problem on G. □

Based on the results above, we implement an algorithm NFBA to compute the optimal vehicle dispatch solution.

Theorem 2: The optimal vehicle dispatch solution can be obtained by algorithm NFBA in $\mathcal{O}(n^4\ log(n))$ time, where n is the number of requests.

Proof: The calculation of the algorithm is mainly divided into three parts: constructing the vehicle-shareability network, constructing the request-link graph and solving the min-cost flow problem. We regard the construction of an edge or a vertex as an atomic action. The vehicle-shareability network includes n vertices and at most $\frac{n(n-1)}{2}$ edges whereby such network can be constructed in $\mathcal{O}(n + \frac{n(n-1)}{2}) = \mathcal{O}(n^2)$ time. According to the graph conversion rules introduced above, there are $2n+m+2$ vertices and at most $m+mn+2n+\frac{n(n-1)}{2}$ edges in the corresponding request-link graph. Due to $m \ll n$, the request-link graph can be constructed in $\mathcal{O}(n^2)$ time. The min-cost flow problem can be efficiently solved by the network simplex algorithm [26] in polynomial time and the computation complexity is $\mathcal{O}(|V|^2|E|log(|V| \cdot c))$ where c is the maximum absolute cost of any edges. In this problem, the maximum absolute cost of all edges equals one, i.e., $c = 1$. Thus, this min-cost flow problem can be solved in $\mathcal{O}((2n + m+2)^2(m+mn+2n+\frac{n(n-1)}{2})log(2n+m+2) = \mathcal{O}(n^4log(n)))$. Hence, the optimal vehicle dispatch solution can be obtained by NFBA in $\mathcal{O}(n^2+n^2+n^4log(n)) = \mathcal{O}(n^4\ log(n))$ time. □

V. Algorithm for Online Vehicle Dispatch

In this section, we address the online vehicle dispatch problem. Compared with the offline problem, the online dispatch problem is even more challenging. Instead of knowing all requests in advance, we only have the information about the requests in the current epoch. To deal with this challenge, we provide an efficient multi-sample multi-network flow-based algorithm MNFBA to address the online vehicle dispatch problem.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on December 29,2024 at 04:35:46 UTC from IEEE Xplore. Restrictions apply.

Algorithm 2 Multi Network Flow-Based Algorithm (MNFBA)

Input:
Index of current epoch, q; request data of current epoch, $\alpha_{i,j,q}$; idle vehicle data of current epoch, β_i; upcoming vehicle data, $\gamma_{i,t}$; number of samples, N;

1: Get N samples of future requests from historical data over the same period;
2: **for** i = 1 to N **do**
3: Construct a zone-centric vehicle-dispatching graph and define a min-cost flow problem F_i on it;
4: **end for**
5: Solve each min-cost flow problem F_i and get the dispatch solution S_i for the current epoch. Then, obtain the cumulative matrix H by adding all S_i;
6: Normalize matrix H by row to get the sampling probability matrix I;
7: For each vehicle a_i in zone z_j, we obtain its target zone by sampling according to the probability vector of z_j. If its target zone is z_k, we check $\alpha_{j,k,q}$ first. If $\alpha_{j,k,q} > 0$, vehicle a_i is dispatched to serve one request of this type and $\alpha_{j,k,q}$ is updated to $\alpha_{j,k,q} = \alpha_{j,k,q} - 1$. Otherwise, vehicle a_i remains idle in zone z_j.

The high-level idea of the design is as follows. On the arrival of each epoch, we first take the request information of the same period in historical data as possible samples of future requests and use each sample to construct a zone-centric vehicle-dispatching graph. Then, we solve the multiple min-cost flow problems defined on each zone-centric vehicle-dispatching graph to get the number of vehicles dispatched to each zone. The results returned by these network flow problems are normalized as the sampling probability. Finally, taking the sampling probability as input, we generate the online vehicle dispatch solution by sampling vehicles dispatched to each zone. Now we introduce the detailed design of the algorithm.

In the first step, we get multiple samples of future requests from the historical data and use each sample to generate a zone-centric vehicle-dispatching graph. To ensure real-time performance, we set a constant sampling length L. On the arrival of an epoch, we get multiple samples from the historical data in the same period with a length of L. That is, suppose the current epoch is q, the sampled data will range from epoch $[q+1, q+L]$. For each sample, we construct a zone-centric vehicle-dispatching graph. Let $\alpha_{i,j,t}$ be the number of requests released in zone z_i and destined for zone z_j in epoch t, which are the statistics information of requests obtained from the current epoch and the sample in the following L epochs. Let β_i denote the number of idle vehicles in zone z_i in epoch q. Let $\gamma_{i,t}$ denote the number of vehicles that are currently running and will become idle in zone z_i in epoch t.

Based on this information, we construct a zone-centric vehicle-dispatching graph and match the problem of maximizing the number of served requests to be the min-cost flow problem on the constructed graph. In this graph, each vertex represents the state of a zone in a special epoch. Each edge and the amount of flow specified on it represents the movement of vehicles between the two zones connected by the edge. The starting point of constructing this graph is to ensure the balance of traffic in each zone in each epoch. Here, we introduce two decision variables $\lambda_{i,j,t}$ and $\rho_{i,t}$ to denote the number of vehicles dispatched from zone z_i to z_j in epoch t and the total number of idle vehicles in zone z_i in epoch t, respectively. The flow balance constraint can be expressed by the following equation.

$$\rho_{i,t} = \rho_{i,t-1} - \sum_{j \in Z} \lambda_{i,j,t-1} + \sum_{j \in Z} \lambda_{j,i,t-w_{j,i}}, \quad \forall i \in \mathcal{Z},\ t \in [1, T] \tag{8}$$

The left side of the equation indicates the number of idle vehicles in zone z_i in the current epoch. The right side of the equation describes the change in the number of idle vehicles in zone z_i. The change includes two parts: the outflow and inflow. The outflow includes all vehicles dispatched to other zones in the previous epoch, i.e., $\sum_{j \in \mathcal{Z}} x_{i,j,t-1}$. The inflow includes vehicles coming from other zones and arriving zone z_i in the current epoch, i.e., $\sum_{j \in \mathcal{Z}} x_{j,i,t-w_{j,i}}$. The steps to construct the graph are as follows.

- Create a source vertex s with demand $-\sum_{i=1}^{l} \beta_i$ and a sink vertex s' with demand $\sum_{i=1}^{l} \beta_i + \sum_{i=1}^{l} \sum_{t=1}^{L} \gamma_{i,t}$.
- For the current epoch and each zone $z_i \in \mathcal{Z}$, create a vertex z_q^i with demand 0 and create a directed edge from vertex s to it with capacity β_i and weight 0.
- For each epoch $t \in \{q+1, \ldots, q+L\}$ and each zone $z_i \in \mathcal{Z}$, create a vertex z_t^i with demand $-\gamma_{i,t}$.
- For epoch $q+L+1$ and each zone $z_i \in \mathcal{Z}$, create a vertex z_{t+1+L}^i with demand 0 and create a directed edge from it to vertex s' with weight 0.
- For each epoch $t \in \{q, \ldots, q+L\}$ and each zone $z_i \in \mathcal{Z}$, create a directed edge from vertex z_t^i to vertex z_{t+1}^i with weight 0.
- For each zone $z_i \in \mathcal{Z}$ and each epoch $t \in \{q, \ldots, q+L\}$, if $\alpha_{i,j,t} \neq 0$, create a directed edge from vertex z_t^i to vertex $z_{t+w_{i,j}}^j$ with capacity $\alpha_{i,j,t}$ and weight -1. If $\alpha_{i,j,t} \neq 0$ and $t + w_{i,j} > q + L + 1$, create a directed edge from vertex z_t^i to vertex z_{q+L+1}^i with capacity $\alpha_{i,j,t}$ and weight -1.

Figure 5 depicts the constructed zone-centric vehicle-dispatching graph. To improve the stability of sampling, we obtain N samples, construct a zone-centric vehicle-dispatching graph for each sample, and define a min-cost flow problem on it. Combining all the subgraphs, we get the multi-sample zone-centric vehicle-dispatching graph. Figure 6 is a tiny example of the multi-sample zone-centric vehicle-dispatching graph, which takes two samples for future requests from the historical data over the same period with a sample length of 2.

On the arrival of a new epoch, we would resample and construct a new multi-sample zone-centric vehicle-dispatching graph to get the vehicle dispatch for the new epoch.

In the second step, we solve the min-cost flow problem on the constructed graph and obtain the sampling probability for further vehicle dispatching. We notice that the min-cost

Authorized licensed use limited to: ShanghaiTech University. Downloaded on December 29,2024 at 04:35:46 UTC from IEEE Xplore. Restrictions apply.

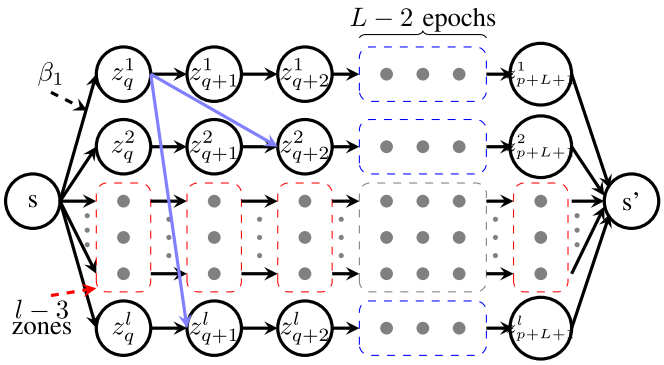


Fig. 5. The constructed zone-centric vehicle-dispatching graph. The blue edges represent the movement of vehicles between different zones which corresponds to the request completion process. And, each blue edge takes the number of requests in the corresponding type as its capacity. The blue edge between vertex z_q^1 and vertex z_{q+2}^2 indicates that it takes two epochs to move from zone z_1 to zone z_2.

flow on each constructed graph determines the number of vehicles to be dispatched to each zone, which maximizes the number of served requests under the given sample of requests. As multiple samples are incorporated, we can solve the min-cost flow problems on each zone-dispatching graph in parallel to maximize the request service rate. As the flow generated in different samples are different, we use a cumulative matrix H to reflect the total amount of vehicle dispatched to each zone. After computing the min-cost flow solution on each graph, we can get the number of vehicles to be dispatched in the current epoch, e.g., the part surrounded by the red dotted line in Figure 6. We count the number of requests of each type that appears in this part and store the result in the cumulative matrix H. H is a matrix of size $l \cdot l$ in which the element $H_{i,j}$ represents the number of vehicles to be dispatched from zone z_i to z_j. And then, we normalize H by row to get the sampling probability matrix I as follow, where each element $I_{i,j}$ indicates the probability that the vehicle in zone z_i should be dispatched to zone z_j.

$$I_{i,j} = \begin{cases} \dfrac{H_{i,j}}{\sum_{k=1}^{l} H_{i,k}}, & \text{if } \sum_{k=1}^{l} H_{i,k} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Take the case in Figure 6 as an example. Let metrics $S1$ and $S2$ in Figure 7 be the computed min-cost flow solutions for the two samples that specify how many vehicles are dispatched to every zone in the current epoch. We can get the cumulative matrix H by adding the matrix $S1$ and $S2$. Then, we normalize the matrix H by row to get the sampling probability matrix I.

Finally, we generate the vehicle dispatch solution according to the obtained sampling probability matrix I. For each vehicle a_i in zone z_j, we obtain its target zone by sampling according to the probability vector I_j. If its target zone is z_k, we check the number of real requests from zone z_j to z_k first, i.e., $\alpha_{j,k,q}$. If $\alpha_{j,k,q} > 0$, vehicle a_i is dispatched to serve a request of this type and $\alpha_{j,k,q}$ is updated to $\alpha_{j,k,q} = \alpha_{j,k,q} - 1$. Otherwise, vehicle a_i remains idle in zone z_j. For instance, in the example of Figure 6, the vehicle in zone z_1 has a probability 75% to be dispatched to zone z_2, and a probability 25% to be dispatched to zone z_3. Suppose that the result of the sampling is zone z_2.

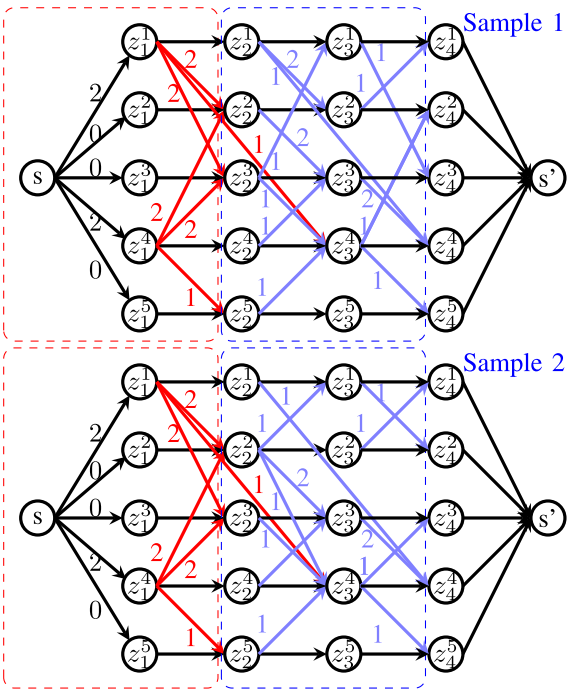


Fig. 6. An example of the multi-sample zone-centric vehicle-dispatching graph. In this case, the city has five zones z_1, z_2, z_3, z_4, z_5, the current epoch is 1 and no vehicle is running. The time distance between zone z_1 and z_4 is two epochs and the time distance between any other two zones is one epoch. Currently, there are four vehicles available, two vehicles in zone z_1, and the other two vehicles in zone z_4. The red edges represent the real requests in the current epoch and the blue edges represent the sample requests in the next two epochs. The number marked on each edge denotes the corresponding capacity.

S1	z_1	z_2	z_3	z_4	z_5
z_1	0	1	1	0	0
z_2	0	0	0	0	0
z_3	0	0	0	0	0
z_4	0	1	0	0	1
z_5	0	0	0	0	0

S2	z_1	z_2	z_3	z_4	z_5
z_1	0	2	0	0	0
z_2	0	0	0	0	0
z_3	0	0	0	0	0
z_4	0	2	0	0	0
z_5	0	0	0	0	0

H	z_1	z_2	z_3	z_4	z_5
z_1	0	3	1	0	0
z_2	0	0	0	0	0
z_3	0	0	0	0	0
z_4	0	3	0	0	1
z_5	0	0	0	0	0

I	z_1	z_2	z_3	z_4	z_5
z_1	0	$\frac{3}{4}$	$\frac{1}{4}$	0	0
z_2	0	0	0	0	0
z_3	0	0	0	0	0
z_4	0	$\frac{3}{4}$	0	0	$\frac{1}{4}$
z_5	0	0	0	0	0

Fig. 7. The returned min-cost flow solution metrics for the case in Figure 6.

Since there are two requests in zone z_1 going to zone z_2, the vehicle is dispatched to serve one of these two requests.

Now we provide the theoretical insights behind the design of the online algorithm and analyze its performance. The normalized sampling probability is obtained from the min-cost flow on the constructed zone-centric vehicle-dispatching graph, which maximizes the number of served requests under the given sample.

Theorem 3: When the distribution of future requests is consistent with the given sample, the online vehicle dispatch solution obtained by MNFBA in the current epoch is optimal for maximizing the expected number of served requests.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on December 29,2024 at 04:35:46 UTC from IEEE Xplore. Restrictions apply.

Proof: Given the real request information in the future L epochs, the optimal dispatch solution, denoted by OPT, should maximize the total number of requests served in the current and next L epochs. Let OPT^c denotes the dispatch solution for the current epoch in the optimal solution. Since the sampling requests are consistent with the real future results, we show that the OPT can be achieved by solving the min-cost flow problem, defined in Algorithm MNFBA, which constructs a zone-centric vehicle-dispatching graph $G'' = (V'', E'')$. There are two main reasons: Firstly, the capacity of each edge in the graph G'' equals to the number of requests between the two zones it is connected to which guarantees that the output solution of the min-cost flow problem is feasible. Secondly, the weight of all these edges is -1 which means serving one request would reduce the total cost by one. Thus, the absolute value of the minimum cost is equal to the number of served requests, and therefore the min-cost flow P on G'' corresponds to the optimal solution OPT. Let X denote the solution generated from P and X^c denotes the solution for the current epoch in X. Thus, we have $OPT^c = X^c$.

For the solution generated under the given sample, the sum of the elements of the i-th row of X^c is equal to the number of idle vehicles in zone z_i, i.e., $\sum_{j=1}^{l} {X^c}_{i,j} = \beta_i$ where ${X^c}_{i,j}$ denotes the number of vehicles dispatched from zone z_i to z_j in the current epoch. After normalizing X^c by row, we obtain the sampling possibility matrix I. According to our online vehicle dispatch strategy, the expected number of vehicles dispatched from zone z_i to z_j will be $\beta_i \cdot I_{i,j} = \beta_i \cdot \frac{{X^c}_{i,j}}{\beta_i} = {X^c}_{i,j}$ which exactly equals to ${X^c}_{i,j}$ and also equals to $OPT^c_{i,j}$. Note that $OPT^c_{i,j}$ is the number of vehicles dispatched from zone z_i to z_j in OPT for the current epoch. Therefore, our online vehicle dispatch strategy is the optimal dispatch strategy for the current epoch in expectation. □

The online vehicle dispatch solution can be obtained by algorithm MNFBA in $\mathcal{O}(|V|^2|E|\log|V|)$ time, where $|V|$ and $|E|$ are the number of vertices and edges in the graph, respectively. The calculation of the MNFBA algorithm can be split into three phases: constructing the zone-centric vehicle-dispatch graph, solving the multiple min-cost flow problems, and dispatching vehicles. In the graph construction phase, we also regard the construction of an edge or a vertex as an atomic action. In the worst case, the time distance between any two regions is one epoch, and there is at least one request between each region pair in each epoch, i.e., the region vertices between two adjacent epochs are fully connected. Therefore, for each sample, $(L+1)l+2$ vertices and at most $(L+2)l^2$ edges would be created, i.e., $|V| = (L+1)l+2$ and $|E| = (L+2)l^2$. Thus, the graph for each sample can be constructed in $\mathcal{O}(|V|+|E|)$ time. Since the maximum absolute cost of all edges in the zone-centric vehicle-dispatch graph is 1, i.e., $c = 1$. Then, using the network simplex algorithm [26], the min-cost flow problem defined on each subgraph can be solved in $\mathcal{O}(|V|^2|E|\log|V|)$ time. In the vehicle dispatching phase, we need to traverse all $\beta = \sum_{i=1}^{l} \beta_i$ vehicles, thus its time complexity is $\mathcal{O}(\beta)$. Since all the samples are independent of each other, the graph construction and min-cost problem-solving phases of all samples can be performed simultaneously. Meanwhile, it is worth noting that in our prototype $\beta \ll |E|$. Hence, the overall time complexity of the algorithm is $\mathcal{O}(|V|+|E|+|V|^2|E|\log|V|+\beta) = \mathcal{O}(|V|^2|E|\log|V|)$. Usually, the number of regions l takes values in the range of hundreds, and the sample length L takes values in the range of tens, so the time complexity of the algorithm can easily meet the requirements of real-time computation, which we have also demonstrated experimentally in the next section.

Theorem 4: Without considering the arrival of subsequent vehicles, the expected competition ratio c of MNFBA is

$$c \geq \frac{\sum_{i=1}^{l}(\sum_{j=1}^{l} \min(\beta_i \cdot I_{i,j}, \alpha_{i,j,q}) - I_{i,i} \cdot \beta_i)}{m \cdot L + Q} \tag{10}$$

where m is the total number of idle vehicles in all zones in the current epoch, Q equals to $\sum_{i=1}^{l} \min(\mu_i, \beta_i)$, and μ_i is the number of requests with a trip length of one epoch in zone z_i in the current epoch.

Proof: To prove the lower bound on the expected competition ratio, we construct a worst case and show the value of the expected competitive ratio in that case. In the worst case, all vehicles can only serve the requests scheduled for the current epoch and cannot serve any more requests in the next L epochs. Therefore, the expected number of requests served according to the dispatch solution of our algorithm is $\sum_{i=1}^{l}(\sum_{j=1}^{l} \min(\beta_i \cdot I_{i,j}, \alpha_{i,j,q}) - I_{i,i} \cdot \beta_i)$ in the $1+L$ epochs. While in the ideal case, each vehicle may serve at most one request per epoch in the next L epochs, which can serve $m \cdot L$ requests in total. The maximum number of requests served in the current epoch is $Q = \sum_{i=1}^{l} \min(\mu_i, \beta_i)$. Thus the maximum number of requests served in the $1+L$ epochs is $m \cdot L + Q$. In this case, the ratio of the solution generated by our algorithm to the optimal solution is $\frac{\sum_{i=1}^{l}(\sum_{j=1}^{l} \min(\beta_i \cdot I_{i,j}, \alpha_{i,j,q}) - I_{i,i} \cdot \beta_i)}{m \cdot L + Q}$. □

Example 1: In the case shown in Figure 6, $\beta_1 = \beta_4 = 2$, $a_{1,2,1} = a_{1,3,1} = a_{4,2,1} = a_{4,3,1} = 2$, $a_{1,4,1} = a_{4,5,1} = 1$, $m = 4$, $L = 2$, $Q = min(5,2) + min(5,2) = 4$. We also get the probability matrix I. Thus, we have $c \geq \frac{min(2\times0.75,2)\times2 + min(2\times0.25,2) + min(2\times0.25,1)}{4\times2+4} = \frac{4}{12} = \frac{1}{3}$. Suppose the final dispatch result is that the two vehicles in z_1 are all dispatched to zone z_2, one vehicle in zone z_4 is dispatched to zone z_2 and the other vehicle in zone z_4 is dispatched to zone z_5. Thus, four vehicles can complete four requests in epoch 1. In the worst case, they cannot serve new requests in the next two epochs, then a total of 4 requests are served in the three epochs. Obviously, in the optimal solution, four vehicles can complete one request per epoch, and they can serve a total of 12 requests in three epochs. Thus, c is equal to $\frac{1}{3}$ which is exactly the lower bound.

To conclude, MNFBA has the following notable advantages. Firstly, the multi-network flow architecture enables the algorithm good scalability. Secondly, compared with prediction-based algorithms, it is easier to implement and more accurate. Thirdly, inspired by the optimal min-cost flow solution, it can significantly improve the efficiency of vehicle dispatch while guaranteeing real-time performance.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on December 29,2024 at 04:35:46 UTC from IEEE Xplore. Restrictions apply.

TABLE II
BASIC PARAMETERS IN THE EXPERIMENTS

Parameter	Numerical value
Longitude range	$(-74.3280, -73.6317)$
Latitude range	$(40.5503, 40.8445)$
Zone edge length	1.2 KM
Number of zones	370
Average speed	8.5 m/s
Basic fare	2.5 $
Charge per kilometer	1.26 $
Fuel cost per kilometer	0.1 $

VI. EXPERIMENTS

This section evaluates the effectiveness and efficiency of the online vehicle dispatch algorithm MNFBA. All experiments were executed on a Windows PC with an Intel i7-7700K CPU and 32GB of RAM running at 4.20 GHz.

A. Baseline Algorithms

We compare the proposed online vehicle dispatch algorithm MNFBA with the following four baseline algorithms.

- GD [11]: In this algorithm, decisions are made in a greedy manner. At the end of each interval, requests are processed in chronological order and the nearest idle vehicle in the same zone will be dispatched to serve the request.
- TSOTS [15]: This algorithm is a two-stage optimization algorithm which takes the next stage demands into account when making the current stage vehicle dispatch solution. The ARIMA model introduced in [22] is used to predict the next stage demands.
- TSONN: This algorithm is also a prediction-based algorithm which is similar to TSOPS, but different in the demand prediction method. In this algorithm, the next stage demands are predicted using a novel neural network model proposed in [24].
- OLSTM [16]: In this algorithm, the dispatch problem is formulated as a multiple-stage stochastic optimization problem. Scenarios observed in the past data are taken as samples and considered as potential future scenarios. Solutions are obtained through a linear programming solver, which is not applicable for large-scale online vehicle dispatch due to computation overhead.

B. Data Processing and Experiment Setting

The trip dataset used in the experiments is a real trip dataset from New York City [27]. Each piece of data contains the start time of a trip and the coordinates of its pick-up and drop-off points. The longitude range and latitude range of New York City's coordinates are set to $(-74.3280, -73.6317)$, $(40.5503, 40.8445)$. The whole city is divided into 370 non-overlapping, same-sized hexagonal zones (the side length is about 1.2 KM) using Uber's open-source library H3 [28]. We use the trip data of February 2016, and some preprocessing operations are conducted on the raw data. First, some out-of-bounds trips are removed according to their coordinates of the boarding and alighting positions. And then, trips with the same pick-up and drop-off zone are filtered. After processing, there are about 300,000 requests per day. We conduct a 7-day online simulation based on the request data from 22nd to 28th February 2016.

Unless other specified, the fleet size is set to 10000. To measure the distance, we use the real road network data provided by OpenStreetMap [29]. Given two GPS coordinates, we map the coordinates to their nearest nodes on the road network first. And then, we consider the distance between these two nodes as the distance between the coordinates. We assume that all drivers are experienced and they can always find the shortest route in driving. Due to the lack of speed information in the trip dataset, we assume all vehicles move at a speed of 8.5m/s. In the experiment, similarly to [15], [16], the length of each epoch is set to 5 minutes in order to avoid too sparse orders in each epoch and to prevent passengers from waiting too long. For baseline algorithm OLSTM, we set its sample size to 10 and its sample length to 5 by default. For our algorithm MNFBA, we set its sample size and sample length to 10 by default.

We evaluate the solutions generated by different algorithms according to the following three metrics:

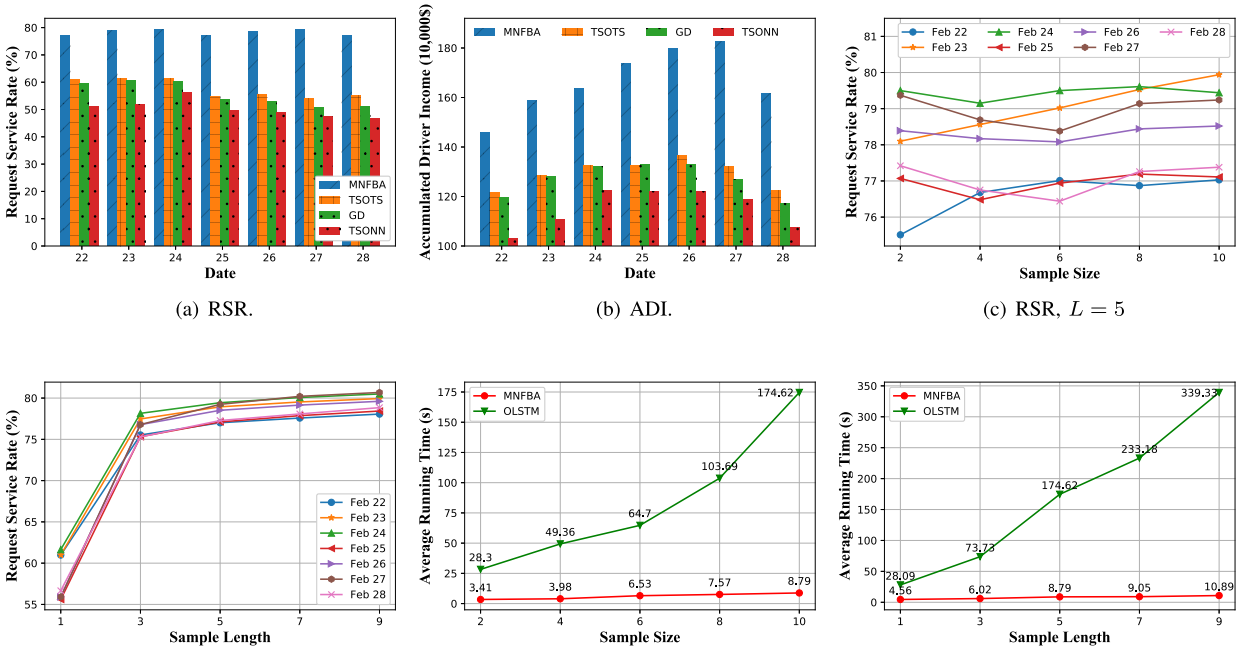
- Request service rate (RSR), the ratio of the number of served requests to the total number of requests.
- Accumulated driver income (ADI), the sum of all drivers' revenue. The following revenue model is used to measure the revenue for each request: $R = A + B \cdot d_{drop} - C \cdot (d_{pick} + d_{drop})$ [30], where R represents the revenue, A represents the basic fare, B represents the charge per kilometer, and C represents the fuel cost per kilometer. In our case, we set $A = 2.5\$$, $B = 1.26\$/\text{KM}$, and $C = 0.1\$/\text{KM}$.
- Running time (RT), the average running time required to calculate the vehicle dispatch solution per epoch.

C. Solution Quality Evaluation

We compare the performance of different online algorithms in terms of RSR and ADI first. As it is shown in Figure 8(a), the performance of MNFBA largely surpasses algorithm GD and prediction-based algorithms TSOTS, TSONN in the RSR metric. The performance of TSOTS is slightly better than GD because it takes demands in the next stage into consideration. TSONN suffers from the lowest RSR among all algorithms because when the time and space granularity are very small, the demand data is relatively sparse, and it is difficult for the neural network to learn a good global prediction model. Meanwhile, the RSR of MNFBA shows good stability, which keeps between 77% and 82%. In Figure 8(b), we compare the performance of the online algorithms in terms of ADI. The results show that the trends of these algorithms on ADI is similar to that of RSR.

To investigate the performance of our proposed algorithm MNFBA when the size of the fleet changes, we list the average RSR, ADI, and RT under different sizes of the fleet in Table III. The fleet size is set to 5000, 10000, and 15000 respectively. We first focus on RSR and ADI which are used to measure the efficiency of algorithms. Algorithm GD

Authorized licensed use limited to: ShanghaiTech University. Downloaded on December 29,2024 at 04:35:46 UTC from IEEE Xplore. Restrictions apply.



(a) RSR. (b) ADI. (c) RSR, $L = 5$

(d) RSR, $N = 10$. (e) RT, $L = 5$. (f) RT, $N = 10$.

Fig. 8. (a) shows the daily request service rate of online algorithms. (b) shows the daily accumulated driver income of online algorithms. (c) shows the daily RSR of MNFBA under different sample sizes when the sample length is fixed at 5. (d) shows the daily RSR of MNFBA under different sample lengths when the sample size is fixed at 10. (e) shows the average running time of MNFBA under different sample sizes when the sample length is fixed at 5. (f) shows the average running time of MNFBA under different sample lengths when the sample size is fixed at 10.

TABLE III

THE EFFECT OF THE NUMBER OF VEHICLES ON VEHICLE DISPATCH ALGORITHMS WITH RESPECT TO RSR, ADI AND RT

No. of Vehicles	5,000			10,000			15,000		
Comparison Metrics	RSR(%)	ADI($)	RT(s)	RSR(%)	ADI($)	RT(s)	RSR(%)	ADI($)	RT(s)
MNFBA	75.14	1,559,859	9.19	79.80	1,747,986	9.27	82.62	1,840,720	9.20
OLSTM	64.06	1,279,854	121.06	76.59	1,608,508	174.62	82.18	1,773,391	188.39
GD	38.91	889,397	0.03	55.44	1,271,133	0.08	67.70	1,539,482	0.11
TSOTS	40.35	895,499	2.54	57.55	1,294,540	2.97	68.91	1,558,297	3.02
TSONN	33.89	780,712	0.39	50.29	1,152,061	0.54	61.01	1,393,983	0.49

is not good enough as it ignores future requests and may cause vehicles to get stuck in zones with few future requests. The performance of TSOTS is slightly better than GD while TSONN is even worse than GD, which indicates that taking future request distribution information into consideration is important, but the performance of prediction-based algorithms is highly depending on the accuracy of prediction. Among these algorithms, our proposed algorithm MNFBA achieves the highest growth in terms of RSR and ADI, which proves that looking ahead for a longer period does help to improve the efficiency of vehicle dispatching. In comparison, OLSTM is the only one that can achieve RSR and ADI close to MNFBA, but due to its computation overhead, it is not practical for large-scale online vehicle dispatching. We can see from the average running time (RT) that OLSTM takes more than two minutes to return a solution. When the number of vehicles increases, the RT of OLSTM increases severely as the decision space of variables becomes larger, while the RT of MNFBA keeps low. To conclude, the experiment results show that MNFBA can not only satisfy real-time performance but also greatly improve the efficiency of vehicle dispatch.

1) Influence of Different Parameters: In this part, we evaluate the influence of different parameters on the performance of our algorithm. MNFBA has two main parameters: sample size N and sample length L. We first fix the sample length to 5 and observe the change in RSR as the sample size increases. In Figure 8(c), when the sample size increases, the RSR of different days has different trends which indicates increasing the sample size can not guarantee an increase in RSR. However, when the sample size is large enough, e.g. 10, the performance of RSR is usually better. This means if we take enough samples into account, we can uncover potential patterns of future demand.

Then, we fix the sample size to 10 and observe the change in RSR as the sample length increases. As it is shown in Figure 8(d), RSRs on different days have the same trend and higher RSR is achieved when increasing the sample length.

2) Scalability: In this part, we show that MNFBA has good scalability when the sample size increases. Figure 8(e) shows the effect of sample size on RT. In this figure, the RT of OLSTM increases dramatically when the sample size increases. This is because increasing the sample size also

Authorized licensed use limited to: ShanghaiTech University. Downloaded on December 29,2024 at 04:35:46 UTC from IEEE Xplore. Restrictions apply.

greatly increases the number of decision variables for the Integral Programming (IP) solver. In contrast, when the number of samples increases, the RT of MNFBA always remains below 10 seconds. Figure 8(f) shows the effect of sample length on RT. Similarly, we can find that the RT of our algorithm MNFBA grows slowly when increasing the sample length, which validates the good scalability of MNFBA.

VII. Conclusion

In this paper, we focus on optimizing the vehicle dispatch for city-scale RHSs. To improve the efficiency of RHSs, we propose novel network flow-based algorithms to solve the vehicle dispatch problem in both offline and online scenarios. For offline scenarios, the provided algorithm NFBA can compute the optimal vehicle dispatch solution by constructing a vehicle-shareability network and transform the original problem into an equivalent min-cost flow problem. For online scenarios, we propose a multi-sample multi-network flow-based algorithm MNFBA. To generate a solution in real-time while efficiently improve the request service rate, MNFBA takes samples from historical data as possible samples of future requests and use each sample to construct a zone-centric vehicle dispatching graph. Then, the sampling probability matrix is obtained by normalizing the results returned by solving the multiple min-cost flow problems defined on these graphs. The final online vehicle dispatch solution for each vehicle is generated by sampling its destination zone according to the sampling probability matrix.

Although experiments on a real-world dataset show that our online algorithm significantly outperforms baseline algorithms in terms of the metrics of interest, it still has some limitations. Two basic assumptions that enable MNFBA to work are 1) the quantity and distribution of requests are essentially stable, and 2) the number of vehicles and the number of requests is independent of each other. In future work, we are going to consider the cases where these two assumptions are not satisfied separately. If the first assumption does not hold, we need to improve the sampling method that measures the similarity between historical and future scenarios and assigns different weights to different samples. If the second assumption is not valid, a prediction model that captures the quantitative relationship between vehicles and requests should be provided to forecast the number of requests under different fleets. Additionally, our results also lead to several promising directions for future work. An open question is how to dispatch vehicles when the service time of the request is more flexible, i.e., each request has a service time window within which the system must decide whether to accept it. Another interesting direction is how to dispatch vehicles under capacity constraints when carpooling is allowed.

References

[1] D. Zhao and M. Chen, "Ex-ante versus ex-post destination information model for on-demand service ride-sharing platform," *Ann. Oper. Res.*, vol. 279, nos. 1–2, pp. 301–341, Aug. 2019.

[2] M. Hu, *Sharing Economy: Making Supply Meet Demand*, vol. 6. New York, NY, USA: Springer, 2019.

[3] M. Truong, D. Purdy, and R. Mawas, "Dispatch system for matching drivers and users," uS U.S. Patent 14 793 593, Jan. 12, 2017.

[4] X. Zhan, X. Qian, and S. V. Ukkusuri, "A graph-based approach to measuring the efficiency of an urban taxi service system," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 9, pp. 2479–2489, Sep. 2016.

[5] 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, p. 534, 2018.

[6] L. Zhang *et al.*, "A taxi order dispatch model based on combinatorial optimization," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 2151–2159.

[7] Z. Xu *et al.*, "Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 905–913.

[8] H. Zheng and J. Wu, "Online to offline business: Urban taxi dispatching with passenger-driver matching stability," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 816–825.

[9] Y. Zhong, L. Gao, T. Wang, S. Gong, B. Zou, and D. Yu, "Achieving stable and optimal passenger-driver matching in ride-sharing system," in *Proc. IEEE 15th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Oct. 2018, pp. 125–133.

[10] J. P. Dickerson, K. A. Sankararaman, A. Srinivasan, and P. Xu, "Allocation problems in ride-sharing platforms: Online matching with offline reusable resources," in *Proc. AAAI*, 2018, pp. 1–5.

[11] D. H. Lee, H. Wang, R. L. Cheu, and S. H. Teo, "Taxi dispatch system based on current demands and real-time traffic conditions," *Transp. Res. Rec.*, vol. 1882, no. 1, pp. 193–200, 2004.

[12] S. C. Chadwick and C. Baron, "Context-aware distributive taxi cab dispatching," U.S. Patent 14 125 549, Mar. 19, 2015.

[13] Y. Tong *et al.*, "Flexible online task assignment in real-time spatial data," *Proc. VLDB Endowment*, vol. 10, no. 11, pp. 1334–1345, Aug. 2017.

[14] Y. Liu, W. Skinner, and C. Xiang, "Globally-optimized realtime supply-demand matching in on-demand ridesharing," in *Proc. World Wide Web Conf.*, 2019, pp. 3034–3040.

[15] M. Lowalekar, P. Varakantham, and P. Jaillet, "Online spatio-temporal matching in stochastic and dynamic domains," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 3271–3277.

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

[17] Z. Liao, "Real-time taxi dispatching using global positioning systems," *Commun. ACM*, vol. 46, no. 5, pp. 81–83, May 2003.

[18] M. Meghjani and K. Marczuk, "A hybrid approach to matching taxis and customers," in *Proc. IEEE Region 10 Conf. (TENCON)*, Nov. 2016, pp. 167–169.

[19] Y. Duan, N. Wang, and J. Wu, "Optimizing order dispatch for ride-sharing systems," in *Proc. 28th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2019, pp. 1–9.

[20] F. Miao *et al.*, "Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 463–478, Apr. 2016.

[21] F. Miao *et al.*, "Data-driven robust taxi dispatch under demand uncertainties," *IEEE Trans. Control Syst. Technol.*, vol. 27, no. 1, pp. 175–191, Jan. 2019.

[22] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Predicting Taxi–Passenger demand using streaming data," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1393–1402, Sep. 2013.

[23] J. Xu, R. Rahmatizadeh, L. Boloni, and D. Turgut, "Real-time prediction of taxi demand using recurrent neural networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 8, pp. 2572–2581, Aug. 2018.

[24] D. Lee, S. Jung, Y. Cheon, D. Kim, and S. You, "Demand forecasting from spatiotemporal data with graph networks and temporal-guided embedding," 2019, *arXiv:1905.10709*. [Online]. Available: http://arxiv.org/abs/1905.10709

[25] X. Ning, L. Yao, X. Wang, B. Benatallah, F. Salim, and P. D. Haghighi, "Predicting citywide passenger demand via reinforcement learning from spatio-temporal dynamics," in *Proc. 15th EAI Int. Conf. Mobile Ubiquitous Systems: Comput., Netw. Services*, Nov. 2018, pp. 19–28.

[26] J. B. Orlin, "A polynomial time primal network simplex algorithm for minimum cost flows," *Math. Program.*, vol. 78, no. 2, pp. 109–129, Aug. 1997.

[27] (2016). *Taxi and Limousine Commission (TLC) Trip Record Data*. [Online]. Available: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[28] (2018). *H3: Uber's Hexagonal Hierarchical Spatial Index*. [Online]. Available: https://eng.uber.com/h3/

Authorized licensed use limited to: ShanghaiTech University. Downloaded on December 29,2024 at 04:35:46 UTC from IEEE Xplore. Restrictions apply.

[29] (2020). *Openstreetmap*. [Online]. Available: https://www.openstreetmap.org/
[30] (2020). *Singapore Taxi Fare*. [Online]. Available: https://www.lta.gov.sg/content/ltaweb/en/public-transport/taxis/fares-a%nd-payment-methods.html

Yuhang Xu received the B.Sc. degree from Soochow University. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Southeast University. His main research interests include combinatorial optimization, scheduling algorithm design, multi-agent systems, and reinforcement learning.

Wanyuan Wang received the Ph.D. degree in computer science from Southeast University in 2016. He is currently an Assistant Professor with the School of Computer Science and Engineering, Southeast University, Nanjing, China. He has published several articles in refereed journals and conference proceedings, such as the IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON CYBERNETICS, AAAI, and AAMAS. He won the best student paper award from ICTAI14. His main research interests include artificial intelligence, multiagent systems, and game theory.

Guangwei Xiong is currently pursuing the M.E. degree with the School of Computer Science and Engineering, Southeast University. His main research interests include reinforcement learning and optimization.

Xiang Liu received the B.Sc. degree from the Dalian University of Technology. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Southeast University. He has published several articles in refereed international conference proceedings, such as International Joint Conference on Artificial Intelligence, International Conference on Autonomous Agents and Multi-agent Systems. His main research interests include algorithmic game theory, mechanism design, and reinforcement learning.

Weiwei Wu (Member, IEEE) received the B.Sc. degree from the South China University of Technology and the Ph.D. degree from the Department of Computer Science, City University of Hong Kong (CityU), and University of Science and Technology of China (USTC) in 2011. He is currently a Professor with the School of Computer Science and Engineering, Southeast University, China. He went to Nanyang Technological University (NTU, Mathematical Division, Singapore, for post-doctoral research in 2012. He has published over 50 peer-reviewed papers in international conferences/journals, and serves as TPCs and reviewers for several top international journals and conferences. His research interests include optimizations and algorithm analysis, wireless communications, crowdsourcing, cloud computing, reinforcement learning, game theory, and network economics.

Kai Liu (Senior Member, IEEE) received the Ph.D. degree in computer science from the City University of Hong Kong in 2011. He is currently an Assistant Professor with the College of Computer Science, Chongqing University, China. From 2010 to 2011, he was a Visiting Scholar with the Department of Computer Science, University of Virginia, Charlottesville, VA, USA. From 2011 to 2014, he was a Postdoctoral Fellow with Nanyang Technological University, Singapore, City University of Hong Kong, and Hong Kong Baptist University, Hong Kong. His research interests include mobile computing, pervasive computing, intelligent transportation systems and the Internet of Vehicles.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on December 29,2024 at 04:35:46 UTC from IEEE Xplore. Restrictions apply.