This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Utility-Based Matching of Vehicles and Hybrid Requests on Rider Demand Responsive Systems

Yongxuan Lai, Shipeng Yang, Anshu Xiong, Fan Yang, Lei Li, and Xiaofang Zhou, *Fellow, IEEE*

***Abstract*— In rider demand responsive systems riders submit requests to demand transit services and the incoming requests and vehicles are matched by the system. This demand-responsive transport problem is viewed by existing research as a kind of spatial matching problem between vehicles and riders. However, existing schemes mainly focus on maximizing the number of matching pairs. It neglects other factors like the length of pickup trajectories and riders' waiting time. And the matching is not revocable, which loses the chance for further optimizations. Moreover, there is still not much work on handling and matching the appointment-based requests that play a key role in the demand-responsive transport market. In this article, we propose an algorithm called BMCF (*Bipartite Minimal-Cost Flow*) to solve the taxi-rider matching problem with appointment-based rider requests on a time-dependent road network. Unlike existing solutions that map the problem into the max flow problem, we transform the optimal matching to a *minimal cost flow problem* which aims to maximize a utility value and could be solved efficiently. Riders and vehicles are modeled as vertices in a bipartite graph, and factors like the length of pickup trajectories, riders' waiting time, etc. are abstracted as utilities and denoted as weights of the edges which are taken into account during the matching process. To the best of our knowledge, the proposed scheme is the first to efficiently integrate and process both the real-time and appointment-based requests within the same framework, and the assignments between vehicles and requests could be dynamically adjusted and revoked to maximise the overall utility. Experimental results show that the proposed algorithm can effectively increase the appointment-based matching ratio and decrease the riders' waiting time (> 10.4%) and vacant vehicles' picking up time (> 60.9%) compared with existing schemes, while at the cost of acceptable increase on the running time.**

***Index Terms*— Appointment-based request, vehicle-rider matching, minimal cost flow, rider demand responsive system.**

Manuscript received December 10, 2019; revised July 5, 2020; accepted August 19, 2020. This work was supported in part by the Natural Science Foundation of China under Grant 61672441 and Grant 61673324; in part by the Natural Science Foundation of Fujian under Grant 2018J01097; in part by the Shenzhen Basic Research Program under Grant JCYJ20170818141325209 and Grant JCYJ20190809161603551; and in part by the Special Fund for Basic Scientific Research Operation Fees of Central Universities under Grant 20720200031. The work of Xiaofang Zhou and Lei Li was supported in part by the Australian Research Council under Grant IC200100022, Grant DP200103650, and Grant LP180100018. The Associate Editor for this article was S. C. Wong. *(Corresponding authors: Yongxuan Lai; Fan Yang.)*

Yongxuan Lai, Shipeng Yang, and Anshu Xiong are with the Department of Software Engineering, School of Informatics, Xiamen University, Xiamen 361005, China, and also with the Shenzhen Research Institute, Xiamen University, Shenzhen 518000, China (e-mail: laiyx@xmu.edu.cn; wydn12345@qq.com; 837412842@qq.com).

Fan Yang is with the Department of Automation, Shenzhen Research Institute, Xiamen University, Xiamen 361005, China, and also with the Xiamen Key Laboratory of Big Data Intelligent Analysis and Decision-Marking, Xiamen University, Xiamen 361005, China (e-mail: yang@xmu.edu.cn).

Lei Li and Xiaofang Zhou are with the School of Information Technology and Electrical Engineering, University of Queensland, Brisbane, QLD 4072, Australia (e-mail: l.li3@uq.edu.au; zxf@itee.uq.edu.au).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/TITS.2020.3019791

I. Introduction

With the development of mobile-oriented cloud architectures and technologies, in recent years there has been increasing concern about the Rider Demand Responsive Systems (RDRS) [1]–[4]. Platforms such as Uber and Didi have brought great convenience to the daily lives of urban citizens. Users or riders submit requests to RDRS to demand transit services; RDRS then makes a match between the incoming requests and vacant vehicles, and guides the vehicles to pick up the riders. Large number of requests arise on these platforms dynamically and need to be assigned to vacant vehicles in real-time based on given objectives, where their spatial adjacency is of the most concern. In this sense, the vehicle-rider matching could be viewed as a kind of dynamic and real-time spatial matching problem that aims to match two sets of objects. Existing research [5]–[9] maps it to a max flow problem, which is to be solved by following some heuristic rules or through some classic algorithms, e.g. the Ford-Fulkerson algorithm [10].

However, there are still drawbacks in existing schemes that hinder the effectiveness of the matching. First, existing schemes only aim to maximize the number of total assigned pairs, and their most-concerning factor is spatial adjacency, which determines whether a vehicle could match a request. But other factors, like trajectory length and pickup time, are neglected. Second, once a pair of vehicle and request is matched, they are not revocable and disappear from the system. This is mainly due to the efficiency and performance consideration, yet it also loses optimization opportunities when a better matching appears. Third, although appointment-based taxi service has been a kind of business of taxi companies for long, existing works and systems only process the real-time requests and the appointment-based requests are not well supported. Appointment-based rider service benefits both the riders and taxi companies in several aspects: (1) Riders make appointments of requests to have their activities well organised and scheduled as they wish. People tend to arrange important activities, e.g. taking a train or airplane, in a priori and make appointments to ask for a trip service. (2) Appointment-based

1524-9050 © 2020 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: Xiamen University. Downloaded on September 26,2020 at 04:33:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

requests make the demand known to the taxi companies, and hence the companies have the opportunity to optimise the vehicle scheduling and dispatching. In other words, ride sharing and carpooling might be easier when requests are known beforehand. (3) Riders usually show willingness to pay more fare for the reserved trip service compared with the ad-hoc service. So taxi companies and drivers would increase their income by providing the appointment-based services.

Albeit the afore-mentioned advantages, the appointment-based requests are not well studied and integrated into the existing RDRS platform. The appointment-based requests are usually processed separately in a semi-manual approach; and in some taxi-hailing platforms the service is not provided. Although there are various factors for the success of the appointment-based trip service, the punctuality when picking up riders and the integration to real-time requests play a very important role from the aspect of the RDRS. On one hand, riders' user experience would degrade sharply with the growth of the waiting time when the scheduled pickup is late. However, as the road networks are affected by various factors [11], it is still hard to incorporate the changing traveling time information for the match of appointment-based requests, and none of the existing work considers this. On the other hand, vehicles should be transparent to the types of requests, although a dominating portion of the requests in the RDRS system are real-time requests. For the appointment-based requests, the system should be flexible enough to dynamically adjust the matching pairs when better matching opportunities emerge in realtime, and drivers only need to follow the guidance of the scheduling system. This requires a seamless integration when handling the real-time or appointment-based requests dynamically. All these factors make providing appointment-based service a challenging issue in RDRS.

In this article, we propose an efficient utility-based vehicle-rider matching algorithm that integrates the processing of real-time and appointment-based requests for RDRS. Factors related to the trajectory, service quality, and profitability within time-dependent road-network are taken into account and embedded into the utility calculating. The proposed algorithm assumes a time-dependent road network model for the estimation of the traveling cost, and models the taxi-rider match as a maximal utility calculation problem. Efficient solutions are adopted to find feasible matches between the vehicles and riders. The major contributions of this article are as follows:

- We propose an algorithm called BMCF (Bipartite Minimal-Cost Flow) to solve the vehicle-rider matching problem, where riders and vehicles are modeled as vertices in a bipartite graph. Unlike existing solutions [5]–[8] that map the matching into the max flow problem, we transform the maximal utility calculation to a *minimal cost flow problem* (MCFP) [12] that takes more factors into consideration for the matching. The proposed algorithm assigns both the real-time and appointment-based requests within the same framework, and the assignments between vehicles and requests are set in a flexible way to be adjusted and revokable to maximise the overall utility.
- We extend the algorithm for the dynamic maximal match utility problem, which handles the dynamics and changes of the vehicles and requests. An incremental match calculation algorithm that adopts grid-based index on the vertices is proposed to facilitate effective pruning of unmatched pairs and to avoid redundant calculation.
- We conduct experiments on real-world road network and datasets to verify the effectiveness of the proposed methods. Experimental results show that the proposed algorithm could effectively increase the success ratio of appointment-based requests and decrease the riders' waiting time ($> 10.4\%$) and vacant vehicles' picking up time ($> 60.9\%$) compared to existing schemes, while at the cost of acceptable increase on the running time.

To the best of our knowledge, the proposed scheme is the first to efficiently integrate the processing of both the real-time and the appointment-based requests within the same framework. The remaining of this article is organized as follows. Section II presents the related work of this article. Section III gives some preliminaries and problem definitions. Section IV presents the detailed description of the BMCF algorithm for the maximal match utility problem. Section V extends the algorithm to handle the dynamics of the maximal match utility problem. Section VI presents the experimental studies and analysis. Finally, section VII concludes the paper and presents some future directions.

II. Related Work

RDRS provides alternative transit service to traditional transportation methods. According to whether requests are predefined or dynamically added, the problem could be classified into the *static* and the *dynamic* RDRS. And recent advancement of shared transit service, e.g. carpool or vanpool [13], [14] all have RDRS integrated as the sub-component.

A. Static RDRS Problem

In RDRS a number of customer requests need to be served door-to-door or curb-to-curb by a set of vehicles. The static RDRS problem could be viewed as a special member of the general class of the Dial-a-Ride Problem (DARP) [3], [15], also known as vehicle routing problem with time windows. DARP is essentially a constraint satisfaction problem, i.e., planning schedules for vehicles, subject to the time constraints on pickup and delivery events. The static version of RDRS corresponds to the static DARP where all customer/rider queries are known in advance, on which existing works on the DARP have primarily focused. Also, as the general DARP is NP-hard, only small instances that involve a few cars and dozens of requests can be solved optimally, usually by resorting to integer linear programming techniques.

B. Dynamic RDRS Problem

In dynamic RDRS problem, requests are popped up in real-time. In real scenarios, e.g. Uber and Didi, there are large number of vehicles and riders, and the most important issue is to quickly match vehicles to the incoming requests.

Authorized licensed use limited to: Xiamen University. Downloaded on September 26,2020 at 04:33:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Tong *et al.* [7] viewed the RDRS as the online minimum bipartite matching problem in real-time spatial data. The authors evaluated four representative online algorithms, and argued that greedy algorithms significantly outperform the other algorithms in almost all practical cases. Tong *et al.* [8] developed a two-step framework that integrates offline prediction for the flexible two-sided online task assignment. Idle workers are allowed to move around if no task is assigned, so the scheme could increase the total number of assigned worker-task pairs. The problem of task assignment in spatial data is also called spatial matching problem [5], [6], which aims to match two sets of objects with optimization goals based on their spatial locations. Yiu *et al.* [6] proposed an algorithm of edge-pruning strategies based on the spatial properties of the problem for optimal assignment. They also proposed an approximate solution that provides a trade-off between result accuracy and computation cost. Hassan and Curry [16] proposed a framework that formulates the online spatial task assignment as the multi-armed bandit problem. They adapted a contextual bandit algorithm to assign a worker based on the spatial features of tasks and workers. The algorithm simultaneously adapts the worker assignment strategy based on the observed task acceptance behavior of workers. Most of the existing approaches for the dynamic RDRS problem only handle the real-time tasks/requests, and the pair of assignments could not be revoked once they are matched. In this article we consider both the real-time and appointment-based requests within the same framework, and the vehicles and requests are set in a flexible way to be revokable to maximise the overall utility. Also, different from the previous approaches that take little consideration on the road conditions, in this article we assume a time-dependent road-network model, which is more consistent with the real-world scenarios.

C. Shared Transit Services

In recent years there are also some research on the shared transit services which allow customers to share vehicles with other customers so as to reduce the travel cost and environmental pollution [14], [17]–[19]. Huang *et al.* [14] proposed algorithms to dynamically match real-time trip requests to servers in a road network to allow ridesharing. A kinetic tree algorithm was developed to efficiently schedule the dynamic requests and adjust the routes on-the-fly. Chen *et al.* [18] considered both the pick-up time and the price in dynamic ridesharing. It used an indexing for the road network and the trip schedules of vehicles, and adopted pruning heuristics to improve the efficiency of the ridesharing. Cheng *et al.* [19] formulated the utility-aware ridesharing problem on road networks. The algorithm assigns time-constrained riders to capacity-constrained and dynamically moving vehicles to maximize the entire utility value, which includes the vehicle-related utility, the riders-related utility, and the trajectory-related utility. These schemes could be viewed as the shared version of the dynamic RDRS problem. Our approach also adopts the utilities as the optimisation objectives, yet the main focus of this article is the processing and integration of appointment-based requests in the nonshared scenarios.

III. Preliminaries and Problem Definition

In this section, we introduce some preliminaries on the road network, rider requests, and utilities. We also present a formal definition of the *maximal utility vehicle-rider matching problem*. Requests of riders dynamically pop up to be assigned to riders under the constraints of valid time windows to achieve the maximal utility of the overall matching.

A. Road Network

The road network is represented by a directed graph $G_r = (V_r, E_r)$, where V_r is a set of vertices and $E_r \subseteq V_r \times V_r$ is a set of ordered pairs of vertices, with a weight function $w : (E_r, t) \rightarrow \mathbb{R}$ mapping edges to time-dependent real-valued weights. The weight of an edge $e(u, v) \in E_r$ at time t is $w(u, v, t)$, which represents the cost of time required to reach v starting from u at time t. It is calculated as the length of edge divided by the speed of edge at time slot that t belongs to:

$$w(u, v, t) = \frac{e.length}{e.speed(slot(t))} \tag{1}$$

Here we assume there is a speed profile that records the speeds in G_r, where series of snapshots of the network speeds are recorded and each edge has an array that records the speeds of time slots. If there is no edge from u to v, then we define $w(u, v, t) = \infty$. Given two vertices x_1, x_k, we also denote $w(x_1, x_k, t)$ as the minimal accumulated weight from node x_1 to x_k if there is a path $\{x_1 \rightarrow x_2 \ldots \rightarrow x_k\}$. $w(x_1, x_k, t)$ is defined as follows:

$$w(x_1, x_k, t) = min(p_k - p_1) \tag{2}$$

where p_i is the time when x_i is visited, which is defined in a recursive way as follows:

$$p_1 = t, \quad p_i = p_{i-1} + w(x_{i-1}, x_i, p_{i-1}), \quad i \in [2, k] \tag{3}$$

If there is no path from x_1 to x_k, the weight is set ∞. Table I summarizes the main notations and their meanings in this article.

B. Vehicle, Rider and Query Set

Vehicles, e.g. cabs, are deployed in the road network to fulfil the requests of riders. The set of vehicles is denoted by C, and each vehicle c is attached with a current location and capacity.

Riders submit requests to RDRS server asking for a service. Each request, q, is associated with a submission timestamp t_0, an origin location o, a destination location d, and a preferred time window $[t_1, t_2]$. The requests could be categorised into two types: the real-time requests, and the appointment-based requests, which are denoted as Q_r, Q_a respectively:

$$\begin{cases} q \in Q_r, & if \ q.t_1 \in [t_0, ct + T_{min}] \\ q \in Q_a, & if \ q.t_1 \in (ct + T_{min}, \infty) \end{cases} \tag{4}$$

where ct is the current time, T_{min} is the minimal time gap for making appointments. The system also sets a time point as the beginning of taking appointment-based requests into the

Authorized licensed use limited to: Xiamen University. Downloaded on September 26,2020 at 04:33:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE I
MAIN NOTATION AND THEIR MEANINGS

Notation	Meaning	Notation	Meaning
$G_r(V_r, E_r)$	road network with vertex set V_r and edge set E_r	$e(u, v)$	directed edge from u to v
$w(u, v, t)$	the weight of an edge $e(u, v)$ at time t	C, Q	set of vehicles and requests
$q.t_0$	the submission timestamp of request q	$q.o, q.d$	the origin and destination location of request q
$q.[t_1, t_2]$	the preferred time window of request q	$G_b(V_b, E_b)$	bipartite graph with vertex set V_b and edge set E_b
T_{min}	minimal time gap for making appointments	T_{ready}	maximal time gap for appointment-based requests to be put to ready set
Q_r, Q_a	set of real-time requests and set of appointment-based requests	Q_a^1, Q_a^2	ready set and waiting set in Q_a
Q_+, Q_-	set of new requests and set of old requests	$\mathcal{M}^*{}_t$	matching that has maximal utility at time t
α, β	trade-off parameter for the calculation of utility	$u(c, q, t)$	the utility of an assignment
$\Delta(c, q, t)$	length of time when the pickup is ahead or behind preferred pickup window	$pt(c, q, t)$	the expected time point of picking up q by vehicle c given time t
$\mathbb{C}$	set of clusters of drop-off locations	K	amplification factor for appointment-based requests
g_x	a grid in G_r	v_q	number of vehicles that can pick up q

matching procedure. So $T_{ready} \geq T_{min}$ is introduced to further divide the appointment set Q_a into two parts:

$$\begin{cases} q \in Q_a^1, & if\ q \in Q_a,\ q.t_1 \in (ct + T_{min}, ct + T_{ready}] \\ q \in Q_a^2, & if\ q \in Q_a,\ q.t_1 \in (ct + T_{ready}, \infty) \end{cases} \tag{5}$$

Then the *ready set* of requests is denoted as Q and defined as follows:

$$Q = Q_r \cup Q_a^1 \tag{6}$$

Set Q_a^2 is called the *waiting set*. Requests in the waiting set do not need to be put into the matching procedure, so it decreases the scale of the matching problem and avoids premature matching pairs. There are two cases for a request in the waiting set to be transferred to the ready set: 1) its pickup time meets the definition in Eq. 5 as time advances; 2) it is explicitly added to Q by the matching system by checking available vehicles nearby. Fig. 1 depicts the type of requests along the time axis, where $Q_r = \{q_1\}$, $Q_a = \{q_2, q_3, q_4\}$, the ready set Q is $\{q_1, q_2, q_3\}$, and the waiting set is $\{q_4\}$. Request q_4 could be transferred to the ready set Q as the current time ct advances. Only requests in the ready set are to be considered for the matching assignments with the vehicles.

C. Utility of Assignment

When a vacant vehicle c is assigned to a request q at time t, an assignment (c, q, t) is generated in the RDRS system.

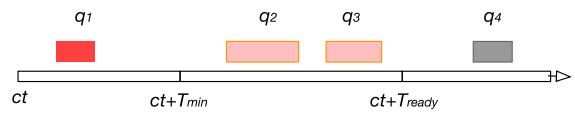


Fig. 1. Illustration of requests along the time axis. $Q_r = \{q_1\}$, $Q_a = \{q_2, q_3, q_4\}$, the ready set Q is $\{q_1, q_2, q_3\}$, and the waiting set is $\{q_4\}$.

RDRS aims to achieve better matches between the vehicles and riders efficiently, and various metrics have been proposed to define the utility of assignments.

The utility of an assignment, denoted as $u(c, q, t)$, consists of three aspects: the trajectory-related utility $trac(c, q, t)$, the service-related utility $serv(c, q, t)$, and the profit-related utility $prof(c, q, t)$. The relative importance of these utilities are controlled by two trade-off parameters, i.e. $\alpha, \beta \in [0, 1]$, as follows:

$$\begin{aligned} u(c, q, t) = \alpha * trac(c, q, t) + \beta * serv(c, q, t) \\ +(1 - \alpha - \beta) * prof(c, q, t), \quad \alpha + \beta \leq 1 \end{aligned} \tag{7}$$

When α is set to 1, only trajectory related factors are considered for the utility calculation, which is the case described in the maximal online matching problem [7].

1) Trajectory-Related Utility: The trajectory-related utility includes the pickup trajectory of vehicles, the energy consumption of pickups, and any other accumulated cost to pick up a rider. The length of trajectory and energy cost increase as the traveling time grows. So without the loss of generality we use the cost of traveling time as an example of the trajectory-related factor, where the trajectory-related utility is defined as follows:

$$trac(c, q, t) = \frac{t_sum - w(c, q, t)}{(k - 1) * t_sum} \tag{8}$$

$$t_sum = \sum_{w(c', q', t) \neq \infty} w(c', q', t) \tag{9}$$

where $w(c, q, t)$ is the cost of traveling time for c to pickup q, i.e. $w(c, q, t) = w(l_c, q.o, t)$. Here l_c is the current location of vehicle c, $w(l_c, q.o, t)$ is the minimal accumulated weight from l_c to $q.o$ at time t. t_sum is defined as the sum of weights of all feasible matches among the vehicles and riders at time t, and k is the number of feasible matches. We could see that the sum of the trajectory utility equals to 1, i.e. $\sum_{w(c', q', t) \neq \infty} trac(c', q', t) = 1$; and Eq. 8 is a normalization of the time cost of picking up riders, where the time cost is inversely linear to the trajectory utility.

2) Service-Related Utility: The service-related utility includes the rider's satisfaction when consuming the RDRS service. It usually depends on the riders' subjective feelings. For example, the friendliness of user interface, comfort, the waiting time, the cleanness of the vehicle, or even whether the driver is hospital, all have impact on this utility [20]. We could combine all these factors into the utility calculation when the data is available and proper quantification could be done on these factors. Without the loss of generality, in this research we use the promptness and punctuality of vehicles as the main factor for the service-related utility. We denote $\Delta(c, q, t)$ as the length of time when the pickup is ahead or behind the

Authorized licensed use limited to: Xiamen University. Downloaded on September 26,2020 at 04:33:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

preferred pickup window, which is further defined as follows:

$$\Delta(c,q,t)=\begin{cases} q.t_1 - pt(c,q,t), & pt(c,q,t) < q.t_1 \\ 0, & q.t_1 \le pt(c,q,t) \le q.t_2 \\ pt(c,q,t) - t_2, & pt(c,q,t) > q.t_2 \end{cases} \tag{10}$$

where $pt(c,q,t)$ is the expected time point of picking up q by vehicle c, i.e. $pt(c,q,t) = t + w(c,q,t)$. The service-related utility is then defined as:

$$serv(c,q,t) = \frac{s_sum - \Delta(c,q,t)}{(k-1) * s_sum} \tag{11}$$

$$s_sum = \sum_{\Delta(c',q',t)\neq\infty} \Delta(c',q',t) \tag{12}$$

Similar to Eq. 8, Eq. 11 is the normalization of $\Delta(c,q,t)$ for all feasible matches. s_sum is defined as the sum of Δ for all potential matches among the vehicles and riders at time t, and k is the number of feasible matches.

3) Profit-Related Utility: The profit-related utility includes the profit made by the taxi company or drivers when providing a successful delivery service. While the pricing policy, the operation cost of the RDRS, and etc. all play an important role for the profit, in most cases a driver would earn more if he/she could pick up more riders. From our study on the predication of rider demands based on the origin-destination dataset [21], we found that there is great difference on the number of requests among the regions and time intervals. The profitability would be affected by the future riding demands. In this research, we use the "heat" of rider demands to set an example for the profit-related utility, while the rider demand is calculated by clustering the origin-destination points. More specifically, the locations of drop-offs within time slots are clustered by K-Means clustering algorithm [22], so that the whole city map is partitioned into K clusters. Based on this, the profit-related utility is defined as follows:

$$prof(c,q,t) = \frac{heat(cluster(q.d), slot(t_q))}{h_sum} \tag{13}$$

$$t_q = t + w(q.o, q.d, t) \tag{14}$$

$$h_sum = \sum_{r\in\mathbb{C}} heat(r, slot(t_q)) \tag{15}$$

where $cluster(q.d)$ is the cluster where the destination $q.d$ belongs to, $heat(r, slot(t_q))$ is the "heat" of riding demands at time interval that t_q belongs to. Here $heat$ is calculated based on time intervals (e.g. 15 minutes), and $slot(t_q)$ is the time slot t_q belongs to. t_q is the estimated arriving time of q, which is calculated based on the accumulated weight $w(q.o, q.d, t)$ defined at Eq. 2. Similarly, the profit related utility, i.e. $heat$, is normalized using h_sum which is the sum of heat for all the clusters which is denoted by $\mathbb{C}$. And the heat of cluster r at slot ts is calculated as follows:

$$heat(r, ts) = \frac{n_trac(r, ts)}{n_area(r)} \tag{16}$$

where $n_trac(r, ts)$ is the average number of trajectories that starts at the region of r, $n_area(r)$ is the number of road network vertexes that are covered by the region of cluster r.

4) Utility of Appointment-Based Request: For an appointment-based request that transits to the ready set, i.e. $q(o, d, [t_1, t_2]) \in Q_2^1$, its utility is defined similar to the real-time request except an amplifying factor:

$$u(c,q,t) = K * u'(c,q,t), \quad q(o,d,[t_1,t_2]) \in Q_2^1 \tag{17}$$

where $u'(c,q,t)$ is the utility if the request is treated as a real-time request, $K \ge 1.0$ is a parameter that amplifies the utility of serving appointment requests compared to real-time requests. Larger K gives larger weight to the appointment based requests, and the RDRS system would assign vehicles to them with greater chance.

For the sake of concise, the utilities are also denoted as $u(c,q)$, $serv(c,q)$, $trac(c,q)$, and $prof(c,q)$ respectively if time t is known in the context.

D. Maximal Utility Vehicle-Rider Matching Problem

From the view of the whole road network G_r, given a set of requests Q, a set of vehicles C, and time t, the overall utility of a match $\mathcal{M}$ between Q and C is:

$$u(G_r, Q, C, \mathcal{M})_t = \sum_{(c,q,t)\in\mathcal{M}} u(c,q,t) \tag{18}$$

where with assignment (c,q,t) vehicle c could only serve request q during the trip. Suppose $\mathbb{M}$ is the set of matches, the goal of the RDRS system is to find a match that maximises $u(G_r, Q, C, \mathcal{M})_t$:

$$\mathcal{M}^*{}_t = \underset{\mathcal{M}}{\operatorname{argmax}} \{u(G_r, Q, C, \mathcal{M})_t : \mathcal{M} \in \mathbb{M}\} \tag{19}$$

The maximal utility changes with time t because the vehicles, requests, and road network are dynamic in nature in real scenarios. When splitting the time domain into slots by a system defined parameter, e.g. 10 seconds or 1 minute per slot, it generates a time sequence $t, t+1, .., t+k$. Given this sequence, the RDRS system would find a sequence of maximal matches $\mathcal{M}^*{}_t, \mathcal{M}^*{}_{t+1}, \ldots, \mathcal{M}^*{}_{t+k}$ that correspond to the time sequence. It is possible that assignments are revoked in the following matches to maximise the overall utility.

IV. Matching Riders and Vehicles

In this section we present the scheme called BMCF (*Bipartite Minimal-Cost Flow*) for the *maximal utility vehicle-rider matching problem*. Riders and vehicles are modeled as vertices in a bipartite graph, and the maximal utility calculation is transformed to the *Minimal Cost Flow* Problem (MCFP) [12] that could be solved accordingly in polynomial time.

A. Transform to Minimum-Cost Flow Problem

A bipartite graph is firstly built based on sets of vehicles and requests. As illustrated in Fig. 2, the set of vehicles are positioned on the left (C), and the set of requests are positioned on the right (Q). Two special nodes denoted by $Source$ and $Sink$ are also added, where $Source$ connects to each of the vehicles and $Sink$ connects to each of the requests. Every edge

Authorized licensed use limited to: Xiamen University. Downloaded on September 26,2020 at 04:33:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

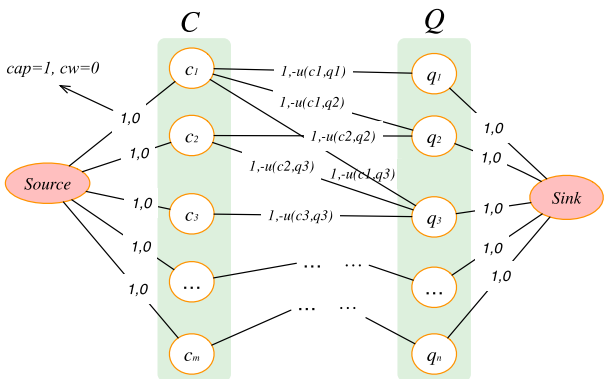


Fig. 2. An example of graph G_b that illustrates the possible match between vehicles (C) and requests (Q). Each edge in the graph is attached with a capacity cap and cost weight cw, which are defined in Eq. 20 and Eq. 21 respectively.

(v_i, v_j) has a *capacity* and a *cost weight* attached to it, which are defined as follows:

$$cap(v_i, v_j) = 1, \quad (v_i, v_j) \in E_b \tag{20}$$

$$cw(v_i, v_j) = \begin{cases} -u(v_i, v_j), & v_i \in C \;\; v_j \in Q \\ 0, & v_i = Source \;||\; v_j = Sink \end{cases} \tag{21}$$

where E_b is the set of edges in G_b.

Given the set of vertices $V_b = C \cup Q \cup \{Source, Sink\}$ and the set of edges E_b, the graph generated is denoted by $G_b(V_b, E_b)$. Each pair of vehicle and request is connected by an edge if the vehicle is able to pick up the rider before the end of the desired time window, i.e. $pt(c, q, t) \leq q.t_2$ for vehicle c and request q. The maximal match utility problem defined at Eq. 19 is then transformed to the following *Minimum-Cost Flow Problem* (MCFP) [12]:

$$\text{minimize}: \sum_{(u,v)\in E_b} cw(u, v) * f(u, v) \tag{22}$$

with the constraints:

$$Capacity\ constraints: \quad f(u, v) \leq capacity(u, v) \tag{23}$$

$$Skew\ symmetry: \quad f(u, v) = -f(v, u) \tag{24}$$

$$Flow\ conservation: \quad \sum_{w\in V_b} f(w, u) + \sum_{w\in V_b} f(u, w) = 0, \tag{25}$$

$$\text{for all } u \neq Source, Sink\ Required\ flow: \quad \sum_{w\in V_b} f(Source, w) = d \text{ and } \sum_{w\in V_b} f(w, Sink) = d \tag{26}$$

where $f(u, v)$ is the value of flow that goes through edge (u, v), d is the number of assignments in the match scheduling $\mathcal{M}$. $f(u, v) = 1$ means there is an assignment (u, v) in the matching. The match $\mathcal{M}^*$ defined by Eq. 22 is denoted as $\mathcal{M}^* \leftarrow MCFP(G_b, Source, Sink)$. When Eq. 22 has the minimal cost, $\mathcal{M}^*$ has the largest overall utility defined at Eq. 19 because the cost weight is set to zero or negative as defined in Eq. 21.

Algorithm 1: Bipartite Match Algorithm

$d = min(|C - \Delta C|, |Q - \Delta Q|)$;
while $d > 0$ **do**
  $\mathcal{M} \leftarrow MCFP(G_b, Source, Sink)$;
  if $\mathcal{M}$ *is a feasible match* **then** break ;
  else $d = d - 1$; ;
return $\mathcal{M}$;

Algorithm 2: Build Bipartite Match Graph G_b

1 create new nodes: $Source$ and $Sink$;
2 **for** *all* $q \in Q$ **do**
3 add node q to graph G_b;
4 add edge $(q, Sink)$ to G_b;
5 $cap(q, Sink)$ =1, $cw(q, Sink)$=0;
6 add node q to V_r;
7 **for** $(u, v) \in \{e|q.o \in e\}$ **do**
8 add edge $(u, q.o)$, $(q.o, v)$ to E_r;
9 **for** *all* $c \in C$ **do**
10 add node c to graph G_b;
11 add edge $(Source, c)$ to G_b;
12 $cap(Source, c)$ =1, $cw(Source, c)$=0;
13 **for** *all* $q \in Q$ **do**
14 call Algorithm 2 to calculate $w(c, q)$;
  /* ct is current time */
15 **if** $ct + w(c, q) \leq q.t_2$ **then**
16 add edge (c, q) to G_b;
17 $cap(c, q) = 1$, $cw(c, q) = -util(c, q)$;
18 **return** *graph* $G_b(V_b, E_b)$;

Note that the value of flow, i.e. d, is equal to or less than the maximal number of assignments, which could be formulated as:

$$d \leq min(|C - \Delta C|, |Q - \Delta Q|) \tag{27}$$

where ΔC are the set of vehicles that has no edge connecting to Q in graph G_b, and ΔQ is the set of requests that has no edge connecting to C. The BMCF algorithm would decrease d one by one until it finds a feasible matching that satisfies all the constraints. Algorithm 1 is the pseudocode of the procedures searching the maximal flow and maximal matching utility.

B. Build Bipartite Graph

Each vehicle c searches its reachable requests, records the utility values and adds edges to the graph. This is actually the graph building process.

Algorithm 2 presents the pseudocode that builds the bipartite graph. It first generates two extra nodes (line 1). Every request q is added to the bipartite match graph G_b, and its edge to $Sink$ is added (line 3-5). The origin location of request $q.o$ is temporally added to the road network G_r as a vertex, and new edges that connect $q.o$ to its nearby vertices are also added (line 6-8). In line 7, $\{e|q.o \in e\}$ is the set of edges that

Authorized licensed use limited to: Xiamen University. Downloaded on September 26,2020 at 04:33:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

$q.o$ overlaps. Every vehicle c is also added to G_b (line 10-12). Then for every request q, Algorithm 3 is called to calculate the minimal weight to pick up riders for c (line 14). If the weight satisfies the time constraint, an edge from c to q is added to G_b, and the capacity and cost weight are set accordingly (line 15-17).

Algorithm 3 is the pseudocode for calculating the shortest path from c to request q based on A* algorithm [23] and some pruning heuristics. It first calls a $apply_pruning$ function to determine whether the vehicle is definitely not reachable from c. If it returns $true$, the algorithm terminates, which avoids further computing (lines 1-2). We would further discuss some pruning heuristics, e.g. grid-based indexing, in section V-B. The locations of c and q are firstly mapped to vertexes in the road network (line 3). The algorithm maintains two sets, i.e. $openSet$ and $visitedSet$, to store vertices that are open for exploration and that have been visited. These sets are initialized empty, and s is added to the $openSet$ (lines 4-5). The algorithm then goes into a loop based on the A* algorithm. While the $openSet$ is not empty, it pops the node with the smallest $fScore$, and add this node v to the $visitedSet$ (line 7). If v is the destination, it returns by reconstructing a path $path(c, q)$ and calculating the weight $w(c, q)$. If v is not the destination, the algorithm loops through the neighbouring nodes of v to update the scores and variables (lines 13-29). The neighbors already visited are skipped (line 14). For each node, e.g. v, its $gScore$ indicates the movement cost to move from the s to v, following the path generated to get there. $tempG$ is a temporal variable storing the $gScore$ of u if its predecessor is v (line 16). Here $w(v, u, ct + v.gScore)$ is defined at Eq. 1, which is the time-dependent traveling cost beginning at $ct + v.gScore$ based on the speed profile of road network G_r. If the neighbour is already explored, i.e. already in the $openSet$, and has a smaller $gScore$ through v, its $gScore$ and $fScore$ are updated and v is set as the predecessor of u (lines 17-22). Here $u.fScore$ is the estimated total cost from s to t through u, which is calculated as $u.gScore + u.H$. $u.H$ is the estimated cost from u to t. In our case, H is calculated as $d(u, t)/max_s$, where $d(u, t)$ is the Euclidean distance from u to t, and max_s is the maximal speed of road segments. If u has not been explored, i.e. it is not in the $openSet$ or $visitedSet$, and its temporary $gScore$ is smaller than maximal time interval of waiting time of request q, the algorithm calculates the scores, sets v as u's predecessor, and add u to the $OpenSet$ (lines 24-29). When $openSet$ is empty, i.e. all nodes are visited yet t is not found, an empty set ϕ and a weight of ∞ are returned (line 30).

Having the graph G_b built, BMCF transforms the problem into the MCFP problem, which is to be solved by calling algorithms such as the Network Simplex algorithm [24], [25] which could be efficiently solved in polynomial time. Due to the limited space of this article we suggest interested readers refer to [25] for more details of the algorithm.

Algorithm 3: Time-Dependent Shortest Path Algorithm

```
   /* Given road network G_r with speed
      profile and current time ct,
      calculate shortest path of traveling
      time from vehicle c to request q.
      It returns (path(c,q), w(c,q)) if q is
      reachable, returns (φ, ∞) otherwise.
      */
1  if apply_pruning(c,q) = true then
2      return φ, ∞;
   /* map locations of c and q to s and t
      */
3  s ← c.loc, t ← q.o;
4  openSet = φ, visitedSet = φ;
5  openSet.add(s);
6  while openSet is not empty do
7      v = openSet.popMinfScore(); visitedSet.add(v);
8      if v == t /* find the destination */
9      then
10         path(c,q) = reconstruct_path(s,t);
11         calculate weight w(c,q) based on path(s,t);
12         return path(c,q), w(c,q);
13     for u ∈ neighbor(v, G_r) do
14         if u is in visitedSet then
15             continue;
16         tempG = v.gScore + w(v,u,ct + v.gScore)
17         if u is in openSet /* u is explored,
               update the scores */
18         then
19             if tempG < u.gScore then
20                 u.gScore = tempG;
21                 u.fScore = u.gScore + u.H;
22                 u.pre = v;
23         else
               /* u is unexplored, calculate the
                  scores */
24             if tempG < q.t_2 − ct then
25                 u.gScore = tempG;
26                 u.H = huristic(u,t);
27                 u.fScore = u.gScore + u.H;
28                 u.pre = v;
29                 openSet.add(u);
30 return φ, ∞;
```

C. Complexity Analysis

The construction of graph G_b consists of $O(|Q| + |C|)$ operations to connect nodes to the $Source$ and $Sink$. Also for every vehicle in C it searches the reachable requests in road network $G_r(V_r, E_r)$. The complexity of the reachability operation depends on the implementation. For example Dijkstra's algorithm based on a binary heap gives an overall running time $O(log|V_r| * (|V_r| + |E_r|))$. There are $|C|$ vehicles, so the complexity of constructing the graph is $O(|C| * log|V_r| * (|V_r| + |E_r|))$.

The maximal utility matching is transformed into the MCFP problem. Its time complexity depends on the implementation of the MCFP algorithm. An implementation of the Network

Authorized licensed use limited to: Xiamen University. Downloaded on September 26,2020 at 04:33:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Simplex algorithm [25] takes $O(|V_b| * |E_b| * \log|V_b| * \log(|V_b|Z))$, where $V_b = Q \bigcup C$ and Z is maximum weight of any edges.

V. Dynamic Matching of Vehicles and Riders

Existing approaches assume that once the vehicle and request pairs are assigned, they can not be revoked. However, the road network, vehicles and requests would dynamically change in real situations. The traveling cost varies at different time periods, vehicles become vacant when the rider gets off, and new requests might pop up. The matching system should update its optimal match M^* to achieve larger overall utility according to the new set of requests and vehicles. This is especially true for the appointment-based requests. When there is still much time for the a request to be served, the assigned vehicle could be changed or be able to serve some real-time requests before picking up the rider of the appointed request. In this section, we present a method to solve the *dynamic maximal match utility problem*, which handles the dynamics and changes of the vehicles and requests.

A. Change of Sets

We denote the set of requests and vehicles as Q' and C' respectively at time $t' = t + ut$, where ut is the time interval the matching system would make the assignments. During the period of time ut, some vehicles become vacant when riders get off, some new requests pop up and are added into the system, and some riders are picked up so they could be removed from the matching system.

The set of requests Q' are composed of two parts, i.e. $Q' = Q_- \cup Q_+$, where Q_- is the set of requests left from previous matching epoch, and Q_+ is the set of requests that are newly added. Similarly, the set of vehicles are updated as $C' = C_- \cup C_+$, where C_- is the set of left vacant vehicles from previous matching epoch, and C_+ is the set of vehicles that are occupied previously yet become vacant at current epoch. Objects (vehicles or requests) in the set $Q - Q_-$ and $C - C_-$ are handled and removed from the match system. There are two cases that objects could be removed from the system: 1) a request q and a vehicle c are matched, and $w(c, q) < \tau$, where τ is a predefined threshold. This happens when the pickup time is approaching and the rider is to be picked up by vehicle c. The assignment pair (c, q) is finally determined, so both objects c and q are removed from the match system; 2) request q matches no vehicle and $q.t_2 < ct$. This happens when the request q is expired, so the object is also removed from the system.

B. Match Recalculation Over Time

The *dynamic maximal match utility problem* could be solved by iteratively conducting the BMCF algorithm. For each time unit, the procedure is called over the road network G'_r to calculate a new match $\mathcal{M}'$. This method is straightforward but expensive, especially when the unit of time ut is small. Here we introduce an efficient match algorithm that reuses the previous calculation results and does the calculation incrementally.

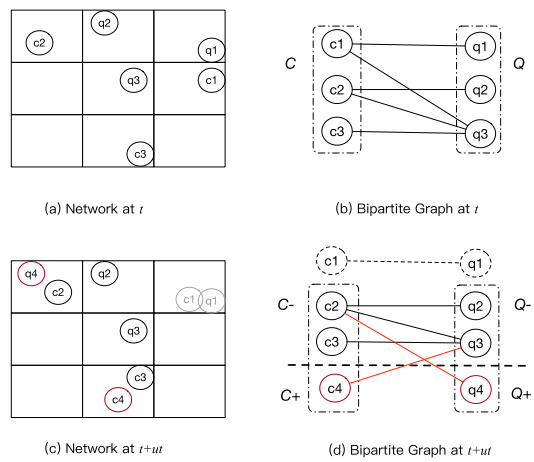


Fig. 3. Illustration of dynamic match of riders and vehicles: (a) vehicles and requests in the filed which is partitioned into grids at time t, (b) corresponding bipartite graph at t, (c) new vehicles c_4 and new request q_4 are added at time $t + ut$, (d) corresponding bipartite graph where match (c_1, q_1) is fixed and removed from the graph. The newly increased calculation is denoted by red lines.

BMCF adopts a cost matrix for optimizing the matching algorithm. The matrix is pre-computed and could be used to approximate the distance of the shortest paths. In more detail, we adopt an approach similar to reference [1] to split and maintain the grid cells of the road network. The minimal cost of time from any two grids are calculated and stored at a matrix L. Each element, e.g. $L(g_c, g_q)$, stores the minimal time traveling from any vertex in grid g_c to any vertex in g_q, where g_c, g_q denote the grids that vehicle c and request q currently belong to. If $L(g_c, g_q)$ is larger than the maximal time allowance of pickup, i.e. $L(g_c, g_q) > q.t_2 - ct$, then q is unreachable for the vehicle. So it does not need to calculate the time cost among the pair, which speeds up the computation. BMCF avoids unnecessay shortest path calculation. In function $apply_pruning$ in Algorithm 3 it prunes out vehicle and request pairs that definitely are not reachable given the pickup time constraint. Since the number of grids is much smaller than that of the vertices in the road network, it is feasible to maintain a time dependent cost matrix for all the grids. We take series of snapshots derived from the historical data and build a matrix for each of them.

Fig. 3 illustrates the dynamic matching of riders and vehicles. The road network is partitioned into grids. Fig. 3(b) is the bipartite graph of vehicles and riders in Fig. 3(a). At time $t + ut$, the match (c_1, q_1) is fixed and should be removed from the graph, and a vacant vehicle c_4 and a new request q_4 emerge and are added to the graph. The newly increased calculation is denoted by red lines. According to the cost matrix, the calculation of cost weight from c_4 to q_2 and q_4 could be avoided because the time cost between the grid which contains c_4 to the grid which contains q_2 and q_4, is larger than the maximal allowed pickup intervals of the requests. The cost of shortest path for pairs (c_2, q_2), (c_2, q_3) and (c_3, q_3) could be approximated by Eq. 28 given that only a small interval of time, e.g. ut, elapses.

$$w(c, q) \simeq \begin{cases} w(c, q) - ut, & c \text{ is along } Path(c, q) \\ w(c, q) + ut, & c \text{ is not along } Path(c, q) \end{cases} \tag{28}$$

Authorized licensed use limited to: Xiamen University. Downloaded on September 26,2020 at 04:33:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Here $Path(c, q)$ is the path with minimal weight from vehicle c to request q. When the weight is continuously approximated by a predefined maximal unit of time, a new weight calculation would be triggered to keep the weight up to date.

Due to the limited space, we leave the description of the incremental match calculation to Appendix I, where Algorithm 4 presents the pseudocode of the algorithm.

C. Early Binding of Appointment Requests

As discussed previously at section III-B, a request is added to the ready set Q if its pickup time is smaller than T. The request then begins to be considered for the match assignments. The amplifying parameter K gives the request a larger utility, so the appointment requests are preferentially assigned to vehicles and could be successfully picked up.

To have greater guarantee on the success of appointment-based requests, another mechanism called "early binding" could be adopted. In early binding, the match system would make predictions on whether the request would be picked up on time even when the pickup time of an appointment-based request is larger than T. BMCF maintains a variable ν_q which indicates the number of vehicles that could travel to the origin of request q within time interval T. If ν_q is zero for a period of time, e.g. $T/3$, then it is added to the ready set Q and the BMCF algorithm would assign a vehicle to it. This procedure is denoted by $checkBind(Q, Q_a)$ in line 1 Algorithm 4 (in Appendix). We also index the vehicles through the same grid cells on the road network G_r, and the variable ν_q could be calculated through the following formula:

$$\nu_q = \sum_{dist(g_q, g_x) < T} count(g_x) \tag{29}$$

where $count(g_x)$ returns the number of vehicles in grid g_x. When a request is from an area that has fewer vehicles, early binding would greatly increase the chance of an appointment request being picked up on time.

VI. Experimental Study

A. Environment Setup

We conduct experiments with real-world road network and taxi trajectory datasets to verify the performance of the proposed algorithm. The schemes are implemented in Java 1.8 and experiments are run on a notebook computer with Intel Core(TM) i7-6700HQ CPU, 2.60GHz, 8 G RAM under Windows 10.

1) Road Networks: The road network of the Xiamen City, Fujian Province, China is used for the simulation, which contains 24750 road vertices and 32364 road segments. We get the map file ([118.0660E,118.1980E]×[24.4240N,24.5600N]) of the road networks from OpenStreetMap [26] and load the graph into the memory using the JGraphT framework[1] 1.3.0 so as to make efficient shortest path queries on the road network.

[1] https://jgrapht.org/

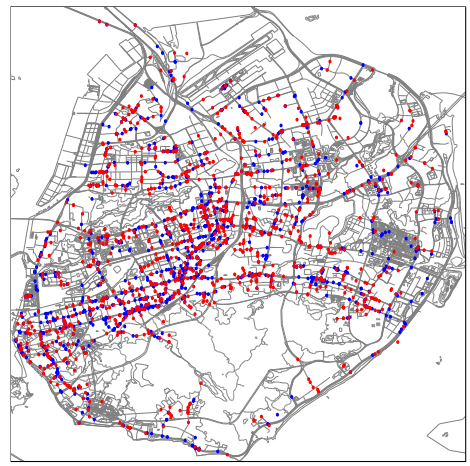


Fig. 4. Distributions of origins (in red) and destinations (in blue) of trips on the Xiamen Island, Fujian Province, China.

2) Dataset: The Xiamen Taxi Trip Dataset [27] is used for the simulation. The datasets consist one-month trajectory data of near 5,000 taxicabs in Xiamen city, China during July 2014, totally about 220 million GPS position records and 8 million live trips. On the time aspect, the dataset covered all the daytime and nighttime on both weekdays and weekends, being able to reflect mobility patterns in heavy, moderate, and light traffic conditions. The trajectory reporting frequency is 1-3 times per minute. So we first extract the speed profile of roads from the dataset. The traveling time and driving speed are time-dependent parameters. It is extracted from the above-mentioned historical trip dataset. More specificly, each road segment has a speed value at an interval of every 10 minutes, and has 144 values (slots) in total within a day. For the road segments that have no GPS records, their speed are estimated from the nearby road segments.

3) Requests and Vehicles: Origin-destination pairs of trips are extracted from taxi operating table of the Xiamen Trip Dataset. These pairs are used as the requests' origin locations and destination locations. Due to the large number of vehicles and riders within the whole city, in the experiment we select 25% of the requests in random from the dataset whose origins begin from 7:00 to 8:00 at $1st$, July 2014. There are 2704 requests in total in the simulation, and each request is assigned a preferred pickup window $[t_1, t_2]$. According to the appointment ratio, a portion of the requests are selected and set as appointment-based requests. Vehicles are deployed randomly distributed within the road networks before the simulation.

4) Compared Algorithms: To the best of our knowledge, there are few research that integrates the real-time and appointment-base requests. To study the performances, we also conduct other three algorithms besides the proposed BMCF scheme.

- Distance first match (*dis-first*): matches a request to the nearest taxi and the matched taxi-request pair is irrevocable. It serves as the baseline of the schemes.
- Utility & grouping (*util-group*): is adopted based on [19] which is utility-aware. It groups the riders and prioritizes the matching of long-distance requests, and requests at

Authorized licensed use limited to: Xiamen University. Downloaded on September 26,2020 at 04:33:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE II
OVERALL PERFORMANCE OF THE SCHEMES

Schemes/ Metrics	Overall Success Ratio	Real-Time Success Ratio	Appointment Success Ratio	Matching Time (s)	Rider Waiting Time (s)	Vehicle Pickup Time (s)	Average Income (¥)
dis-first	0.893	0.897	0.868	0.049	139.682	355.896	53.908
util-group	0.891	0.886	0.924	0.042	121.855	307.278	54.313
max-flow	0.895	0.890	0.925	0.062	121.511	306.583	54.402
max-util	0.895	0.889	0.926	0.063	124.411	313.792	54.345
bmcf-matrix	0.895	0.889	0.934	0.121	99.410	123.228	54.746
BMCF	**0.914**	**0.903**	**0.954**	**0.137**	**108.848**	**119.756**	**55.465**

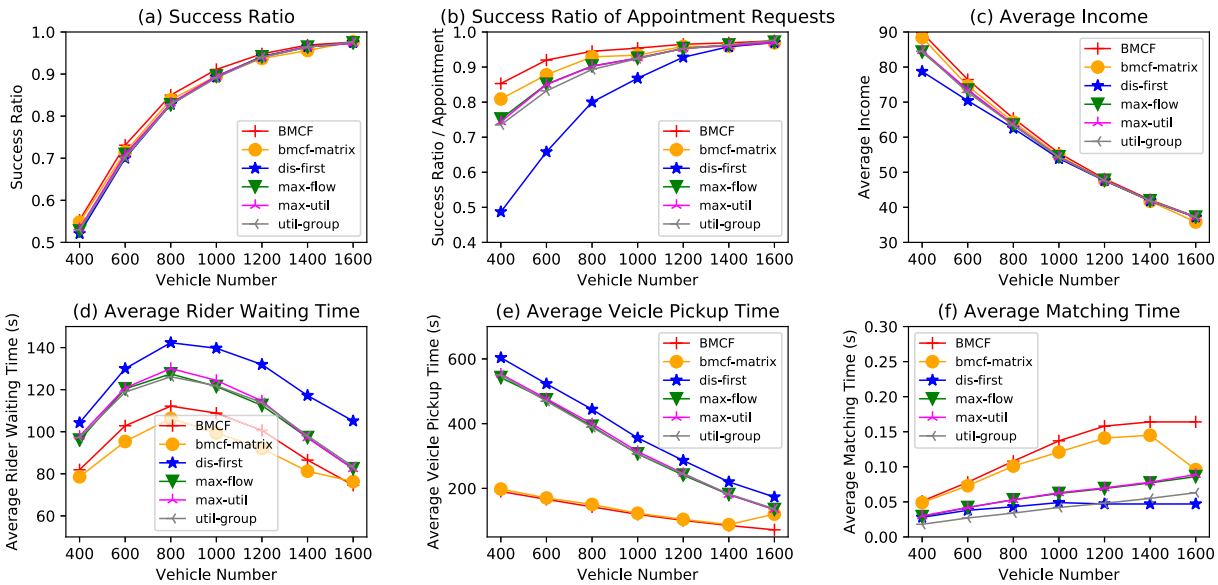


Fig. 5. Impact of number of vehicles on (a) the overall success ratio, (b) success ratio of appointment requests, (c) average income, (d) riders' average waiting time, (e) vehicles' average pickup time, and (f) the average matching time.

areas with a large number of requests will be matched first.

- Maximum flow match (*max-flow*): is adopted based on [7], [8], which uses bipartite graph to match taxis and requests as more as possible.
- Maximum utility match (*max-util*): uses bipartite graph to match taxis and requests based on the utility calculation similar to BMCF, but the matches are not revokable.

Also, the performance of a variant of the proposed BMCF scheme is presented. The *bmcf-matrix* scheme maintains a cost matrix to speed up the calculation and dynamically calculates and approximates the match over time, which is discussed at section V. Six factors are adopted as the main performance metrics, including 1) the ratio of successful match, 2) the average time of matching, 3) the average waiting time of riders that are successfully picked up,[2] 4) the average time of vacant vehicles travel to pick up the riders, 5) the average income of drivers, and 6) the success ratio of appointment-based requests.

B. Result Analysis

1) Overall Performance: By default, there are 1000 vehicles deployed in the field with random initial locations. The length of the pickup window is 300 seconds, the percentage of appointment-based requests are 15%, the interval of fixed matches is 60 seconds, the balance factor α and β are both set to 0.4, and the amplification factor for the appointment-based requests K is set to 1.5. The minimal time gap for making appointments T_{min} and the time gap for setting a ready request T_{ready} are both set to 15 minutes. The requests and vehicles are refreshed for the match calculation every 5 seconds by default.

Table II displays the overall performance of different approaches. BMCF has the highest overall success ratio of requests at 0.914, which is about 2% higher than those of other schemes. The ratios of real-time requests and appointment-based requests are also presented. BMCF has the appointment success ratio at 0.954, while those of the *dis-first*, *util-group* and *max-util* schemes are less than 0.93. This is because BMCF specially treats the appointment-based requests with larger utility, while other schemes treat all requests equally.

The proposed BMCF scheme, as well as its variant, has the shortest waiting time, either for the riders' waiting time or the vehicles' pickup time. It achieves a reduction of more than 10.4% (15 seconds) in the riders' waiting time and gains at least 60.9% of reduction on the average time when vacant vehicles pick up riders. This is largely because in BMCF the

[2]waiting time of a request q is calculated as the pickup time minus $q.t_1$.

Authorized licensed use limited to: Xiamen University. Downloaded on September 26,2020 at 04:33:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

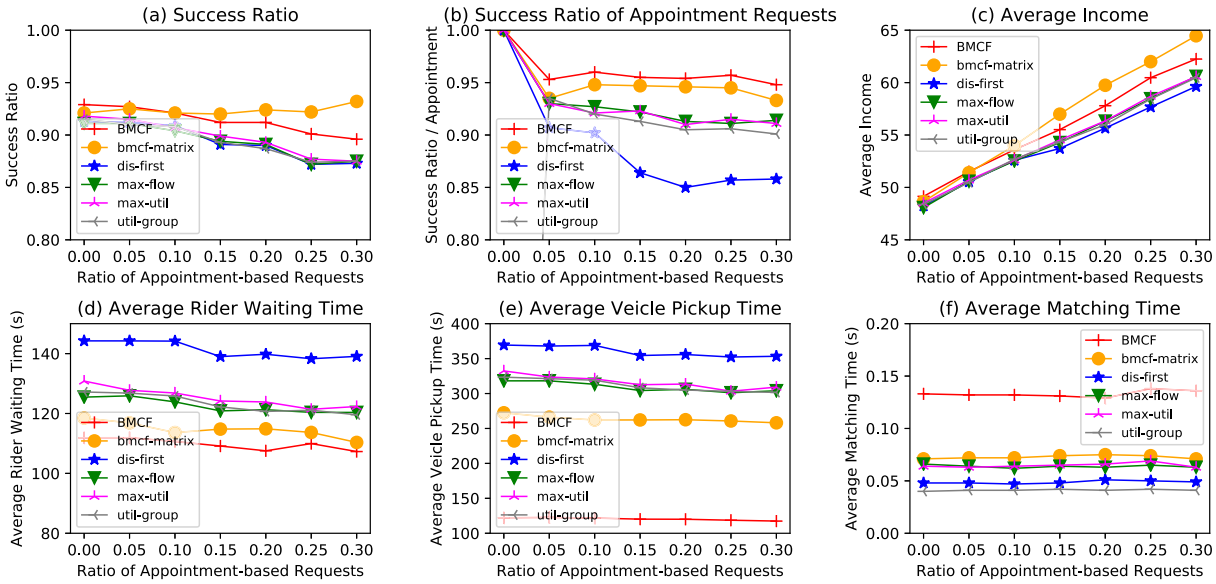


Fig. 6. Impact of ratio of appointment-based requests on (a) the overall success ratio, (b) success ratio of appointment requests, (c) average income, (d) riders' average waiting time, (e) vehicles' average pickup time, and (f) the average matching time.

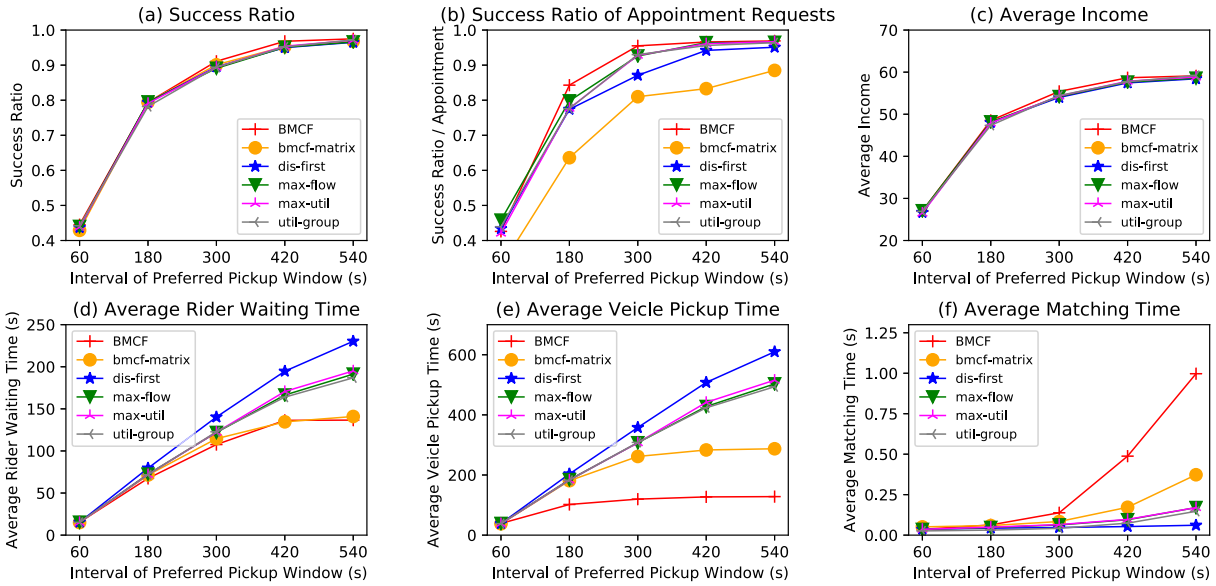


Fig. 7. Impact of preferred pickup window on (a) the overall success ratio, (b) success ratio of appointment requests, (c) average income, (d) riders' average waiting time, (e) vehicles' average pickup time, and (f) the average matching time.

matched requests are flexible and revocable so that vacant vehicles with smaller distance could be matched to achieve larger utility. The performance of *max-util* is similar with BMCF. But as the assignments are not revokable, taxis might have to fulfill determined requests even when better matches appear. Compare to other schemes, BMCF schemes have larger matching time per epoch. The matching time is more than 0.121 seconds, while other schemes is about 0.049∼0.063 seconds. This is because of the extra complexity solving the MCFP problem. However, this increased matching time is acceptable for the vehicle-rider matching which has a period of 5 seconds for accumulating the requests and vehicles. Moreover, in this experiment we only uses an notebook PC with ordinary hardware for all the experiments. Smaller matching time would be achieved if we adopt more advanced hardwares.

The overall income of drivers are also showed in Table II. We follow a general and common pricing strategy that follows the following rules of fees: 1) 13 RMB for the trip within 3 km long, if the length of trajectory is larger than 3 km, charge 2 RMB per km for the rest of the trip; 2) charge 20 RMB additional service fee for the appointment-based requests. As depicted, the proposed BMCF scheme has the highest average income per taxi at 55.465 RMB, while *dis-first* has the lowest income at 53.908 and other schemes are in the middle. *dis-first* has the lowest average income mainly because it has the lowest satisfaction ratio, which indicates a large

Authorized licensed use limited to: Xiamen University. Downloaded on September 26,2020 at 04:33:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

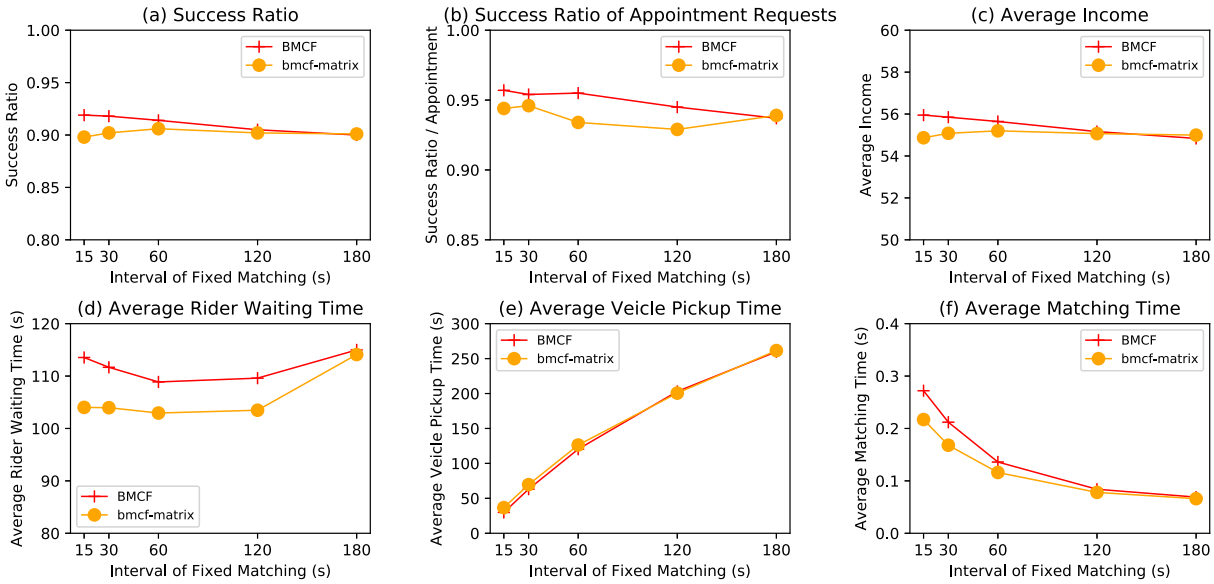


Fig. 8. Impact of interval for fixed matching on (a) the overall success ratio, (b) success ratio of appointment requests, (c) average income, (d) riders' average waiting time, (e) vehicles' average pickup time, and (f) the average matching time.

portion of requests are failed to be satisfied. Combined with the cost of time and petrol, the proposed BMCF actually has larger economic gain compared with other schemes because it has more than 60.9% reduction on the time of interval when vehicles are going to pick up the riders.

2) Impact Factors: We also vary other parameters, i.e., the number of taxis, the length of preferred pickup window, and the percentage of appointment-based requests to study their impact on the schemes.

Fig. 5 depicts the impact of the number of vehicles. When there are more vehicles, the success ratio increases and the income per driver decreases for all the schemes. By default there are 4974 vehicles in the dataset, yet only 25% of the requests are included in the simulation. So there are roughly fewer than 1200 vehicles, which means there would be a lack of vehicles for the requests. BMCF has the best performance on the success ratio of appointment request, especially when there are not enough vehicles. BMCF achieves a gain of about 10% on the success ratio compared to other schemes. This is because the proposed scheme gives higher weight to the appointment-based requests, and these requests are more likely to be served in the matching process. The riders' waiting time increases as the vehicle number increases, and then decreases as the vehicle number grows larger. We only calculate the waiting time of successful pickups, so when there are fewer vehicles and smaller success ratio, the riders waiting time is also smaller, e.g. between 80 to 110 seconds for all the schemes. The riders' waiting time increases when there are more vehicles and larger success ratio, and the waiting time begins to decrease when more than 800 vehicles are deployed. Also, the average time for vehicles to pick up riders decreases and the matching time increases as the vehicles number grows.

Fig. 6 depicts the impact of the ratio of appointment-based requests that varies from 0.05 to 0.30. From the figure, we could see there is a trend that the overall success ratio decreases as the appointment ratio increases, yet the proposed

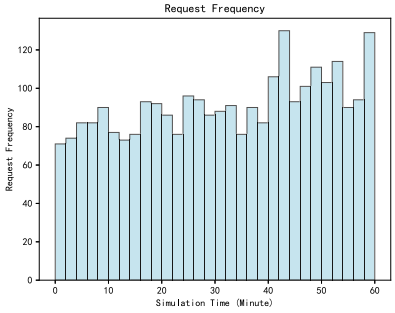


Fig. 9. Histogram of the distribution of the requests, where y-axis denotes the number of requests in an interval of every two minutes.

BMCF has higher overall success ratio and appointment-based success ratio. Also, based on the current pricing policy that has additional service fee for appointment-based requests, the income of drivers increases from about 47 to 63 when the ratio of appointment-based requests grows from 0 to 0.3. The appointment ratio has relatively less impact on the riders' waiting time, vehicles' pickup time and the matching time.

Fig. 7 illustrates the impact of the preferred pickup window of requests, which varies from 60 to 540 seconds. The success ratio and the drivers' income all increase as the preferred pickup window grows for all the schemes. The success ratio grows to near 0.97 when the window is 540 seconds. Larger window means more time tolerance and flexibility when picking up riders, hence it is easier to pick up riders before the requests are outdated. Also, larger pickup window interval indicates the riders could wait more time and vacant vehicles could travel more to pick up the riders, so the riders' waiting time and vehicles' pickup time all increases as well for all schemes. But for the proposed BMCF scheme, the increase rate is much smaller than other schemes. This is because our scheme takes the waiting time into part of the utility calculation, so the final matching would prefer a match that has

Authorized licensed use limited to: Xiamen University. Downloaded on September 26,2020 at 04:33:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

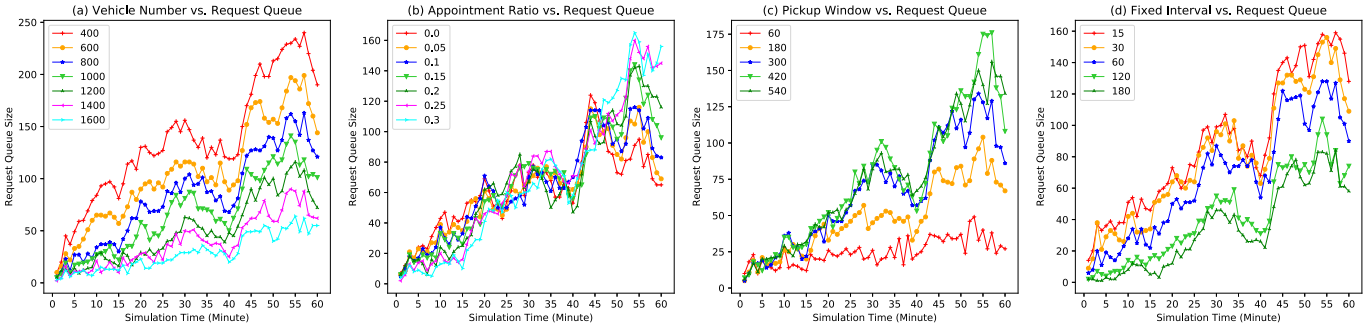


Fig. 10. Change of matching queue size with the simulation time, when given different number of vehicles (a), appointment ratio (b), pickup window size (c) and interval for fixed matching (d).

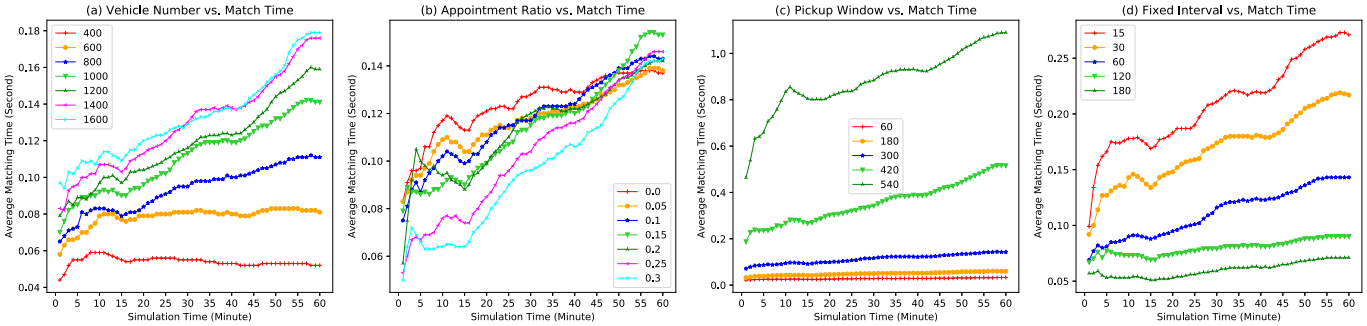


Fig. 11. Change of average matching time with the simulation time, when given different number of vehicles (a), appointment ratio (b), pickup window size (c) and interval for fixed matching (d).

the waiting time as small as possible. In Fig. 7(e), the matching time grows as the interval of pickup window increases, yet BMCF schemes have a larger growing rate. This is because the queue of matches becomes larger when the pickup window expands. We would further discuss at the next subsection.

3) More Analysis on BMCF: We also conduct experiments and calibrate other impact factors that only belong to the proposed BMCF scheme.

Given a time interval ti, a match is called "fixed match" if the remaining time of a request approaching to $q.t_1$ is smaller than ti, i.e. $q.t_1 - ct \leq ti$. When a match is fixed, its rider and vehicle can not be revoked. So the time point of fixed match plays an important role. Fig. 8 depicts the impact of the interval for fixed matches. When the interval of fixed match increases, the success ratio, and the average income decreases accordingly, yet the vehicles' pickup time increases. This is because larger interval means fewer riders and vehicles could not be revoked to make matchings to achieve higher utility. The riders' waiting time first decreases and then increases with the interval. BMCF and *bmcf-matrix* have the smallest waiting time when the interval is 120 or 60 seconds respectively. The *bmcf-matrix* scheme is similar to BMCF but adopts cost matrix to estimate the distance of shortest paths. Its performance is close to that of *bmcf*, yet achieves a speedup for the matching calculation. About 17% percent of CPU time is saved when the interval is about 15 seconds, yet the performance gap on CPU decreases and becomes trivial when the interval increases.

Fig. 9 depicts a histogram of requests coming in with time. The y-axis denotes the number of requests in an interval of every two minutes. The main time cost of the matching time of the BMCF scheme lies on solving the *minimal cost flow problem* (MCFP) whose complexity is directly related to the number of vertexes of the bipartite graph. In the algorithm implementation, we store the unmatched requests and matched yet revokable requests into a queue. Fig. 10 and Fig. 11 depict the change of queue size and the matching time with the simulation time given different parameter settings. The riders and vehicles are matched every 5 seconds by solving a MCFP problem based on the bipartite graph. Some matches are fixed and some matches could still be revoked to have better matching based on the utility calculation. The matching queue has a trend to grow with time, so is the cost of the matching time. Also, larger vehicle number, larger appointment ratio, larger pickup window and smaller interval of fixed match means larger queue sizes, so it takes larger matching time as depicts in Fig. 11(a)∼(d). When concerning the factor of fixed interval, smaller fixed interval means more requests are accumulated at the queue, and hence more time are needed for solving the minimal cost bipartite matching. As depicted in Fig. 9 there is a fall on the number of incoming requests from12 to 16 minutes, so there is also a concave on the line of the matching time at 16 minutes at Fig. 11(d). By default the fixed interval is set to 5 minutes, and the average matching time is about 0.15 seconds. In the implementation we adopt the Shortest Path Faster Algorithm (SPFA) [28] for the minimal weighted shortest paths when solving the MCFP problem. The worst-case running time of the SPFA algorithm is $O(|V| \cdot |E|)$, which is like the standard Bellman-Ford algorithm [29], but experiments suggest that the average running time is $O(|E|)$. Also in the implementation, we adopt

Authorized licensed use limited to: Xiamen University. Downloaded on September 26,2020 at 04:33:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

multi-threading programming model with 4 threads to calculate the shortest paths, which speeds up the overall matching of vehicles and riders in the bipartite graph. As depicted in Fig. 11, for most scenarios the average matching time is less than 0.25 seconds, given that a notebook PC with commodity hardware is used for all the experiments.

We also varied the balance utility factors in Eq. 7 to study their impacts. Due to the limited space, the detailed experimental results and discussions are presented in Appendix II.

VII. Conclusion

We have proposed an algorithm called BMCF to solve the vehicle-rider match scheduling problem on the time-dependent road network. The trajectory and service related utilities are defined, base on which the maximal utility calculation is transformed to the minimal cost flow problem (MCFP) that could be solved efficiently. The scheme integrates and processes both the real-time and appointment-based requests, and requests could be revoked to maximise the overall utility. Experimental results show that the proposed scheme can effectively increase the appointment-based matching ratio and decrease the riders' waiting time and vacant vehicles' picking up time, while at the cost of acceptable increase on the running time.

For the future work, we are to investigate other factors, e.g. desirability and variability of requests / trips and propose efficient algorithms to further optimize the vehicle-rider matching. Also, we are to study the problem of appointment-based car pooling or sharing systems under the similar matching framework proposed at this article.

References

[1] S. Ma, Y. Zheng, and O. Wolfson, "T-share: A large-scale dynamic taxi ridesharing service," in *Proc. IEEE 29th Int. Conf. Data Eng. (ICDE)*, Apr. 2013, pp. 410–421.

[2] S. Ma, Y. Zheng, and O. Wolfson, "Real-time city-scale taxi ridesharing," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 7, pp. 1782–1795, Jul. 2015.

[3] A. Attanasio, J.-F. Cordeau, G. Ghiani, and G. Laporte, "Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem," *Parallel Comput.*, vol. 30, no. 3, pp. 377–387, Mar. 2004.

[4] Y. Lai, F. Yang, L. Zhang, and Z. Lin, "Distributed public vehicle system based on fog nodes and vehicular sensing," *IEEE Access*, vol. 6, pp. 22011–22024, 2018.

[5] R. C.-W. Wong, Y. Tao, A. W.-C. Fu, and X. Xiao, "On efficient spatial matching," in *Proc. 33rd Int. Conf. Very Large Data Bases*, VLDB Endowment, 2007, pp. 579–590.

[6] L. H. U, M. L. Yiu, K. Mouratidis, and N. Mamoulis, "Capacity constrained assignment in spatial databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2008, pp. 15–28.

[7] Y. Tong, J. She, B. Ding, L. Chen, T. Wo, and K. Xu, "Online minimum matching in real-time spatial data: Experiments and analysis," *Proc. VLDB Endowment*, vol. 9, no. 12, pp. 1053–1064, Aug. 2016.

[8] 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.

[9] P. Cheng, C. Feng, L. Chen, and Z. Wang, "A queueing-theoretic framework for vehicle dispatching in dynamic car-hailing," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 1622–1625.

[10] U. Zwick, "The smallest networks on which the ford-fulkerson maximum flow procedure may fail to terminate," *Theor. Comput. Sci.*, vol. 148, no. 1, pp. 165–170, Aug. 1995.

[11] A. Abadi, T. Rajabioun, and P. A. Ioannou, "Traffic flow prediction for road transportation networks with limited traffic data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 653–662, Apr. 2015.

[12] A. V. Goldberg and R. E. Tarjan, "Finding minimum-cost circulations by canceling negative cycles," *J. ACM*, vol. 36, no. 4, pp. 873–886, Oct. 1989.

[13] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti, "Quantifying the benefits of vehicle pooling with shareability networks," *Proc. Nat. Acad. Sci. USA*, vol. 111, no. 37, pp. 13290–13294, Sep. 2014.

[14] Y. Huang, F. Bastani, R. Jin, and X. S. Wang, "Large scale real-time ridesharing with service guarantee on road networks," *Proc. VLDB Endowment*, vol. 7, no. 14, pp. 2017–2028, Oct. 2014.

[15] J.-F. Cordeau and G. Laporte, "The dial-a-ride problem: Models and algorithms," *Ann. Oper. Res.*, vol. 153, no. 1, pp. 29–46, 2007.

[16] U. U. Hassan and E. Curry, "A multi-armed bandit approach to online spatial task assignment," in *Proc. 11th Int. Conf. Autonomic Trusted Comput. Ubiquitous Intell. Comput.*, Dec. 2014, pp. 212–219.

[17] N. Ta, G. Li, T. Zhao, J. Feng, H. Ma, and Z. Gong, "An efficient ride-sharing framework for maximizing shared route," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 2, pp. 219–233, Feb. 2018.

[18] L. Chen, Q. Zhong, X. Xiao, Y. Gao, P. Jin, and C. S. Jensen, "Price-and-time-aware dynamic ridesharing," in *Proc. IEEE 34th Int. Conf. Data Eng. (ICDE)*, Apr. 2018, pp. 1061–1072.

[19] P. Cheng, H. Xin, and L. Chen, "Utility-aware ridesharing on road networks," in *Proc. ACM Int. Conf. Manage. Data (SIGMOD)*, 2017, pp. 1197–1210.

[20] I. Balachandran and I. B. Hamzah, "The influence of customer satisfaction on ride-sharing services in Malaysia," *Int. J. Accounting Bus. Manage.*, vol. 5, no. 2, pp. 184–196, 2017.

[21] Y. Lai, K. Zhang, J. Lin, F. Yang, and Y. Fan, "Taxi demand prediction with LSTM-based combination model," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Dec. 2019, pp. 944–950.

[22] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Appl. Statist.*, vol. 28, no. 1, pp. 100–108, 1979.

[23] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[24] 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.

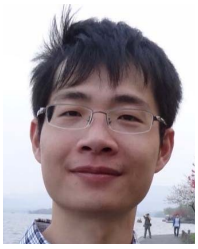
[25] R. E. Tarjan, "Dynamic trees as search trees via euler tours, applied to the network simplex algorithm," *Math. Program.*, vol. 78, no. 2, pp. 169–177, Aug. 1997.

[26] M. Haklay and P. Weber, "OpenStreetMap: User-generated street maps," *IEEE Pervas. Comput.*, vol. 7, no. 4, pp. 12–18, Oct. 2008.

[27] Y. Lai, Z. Lv, K.-C. Li, and M. Liao, "Urban traffic Coulomb's law: A new approach for taxi route recommendation," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 8, pp. 3024–3037, Aug. 2019.

[28] E. F. Moore, "The shortest path through a maze," in *Proc. Int. Symp. Switching Theory*, 1959, pp. 285–292.

[29] R. Bellman, "On a routing problem," *Quart. Appl. Math.*, vol. 16, no. 1, pp. 87–90, 1958.



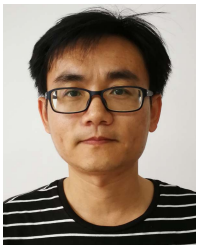
Yongxuan Lai received the Ph.D. degree in computer science from the Renmin University of China in 2009. He was a Visiting Scholar with the University of Queensland, Australia, from September 2017 to September 2018. He is currently a Professor with the School of Informatics, Xiamen University, China. His research interests include network data management, vehicular ad-hoc networks, and big data management and analysis.

Shipeng Yang received the B.S. degree from the Department of Software Engineering, Xiamen University, in 2017. He is currently pursuing the master's degree with the School of Informatics, Xiamen University. His research interests include algorithms, data mining, and intelligent transportation systems.

Authorized licensed use limited to: Xiamen University. Downloaded on September 26,2020 at 04:33:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Anshu Xiong is currently pursuing the bachelor's degree majored in software engineering with the School of Informatics, Xiamen University. His research interests include algorithms and data mining.



Fan Yang received the Ph.D. degree in control theory and control engineering from Xiamen University in 2009. He is currently an Associate Professor with the Department of Automation, Xiamen University. His research interests include feature selection, ensemble learning, and intelligent transportation systems.

Lei Li received the Ph.D. degree from the Data and Knowledge Engineering (DKE) Group, School of Information Technology and Electrical Engineering, University of Queensland, Australia, in September 2018. He is currently a Post-Doctoral Research Fellow with the Data Science Group, University of Queensland, and a Data Scientist with Redback Technologies. His research interests include spatial temporal data, trajectory data, graph, and distributed database.

Xiaofang Zhou (Fellow, IEEE) received the bachelor's and master's degrees in computer science from Nanjing University in 1984 and 1987, respectively, and the Ph.D. degree in computer science from the University of Queensland in 1994. He is currently a Professor of computer science with the University of Queensland and also the Head of the Data Engineering and Pattern Recognition Research Division, UQ. He is also a specially appointed Adjunct Professor of computer science with Soochow University, China. His research interests include spatial and multimedia databases, high-performance query processing, web information systems, data mining, and data quality management.

Authorized licensed use limited to: Xiamen University. Downloaded on September 26,2020 at 04:33:58 UTC from IEEE Xplore. Restrictions apply.