A Vehicle Value Based Ride-Hailing Order Matching and Dispatching Algorithm

Shuai Xu[1], Zeheng Zhong[1], Yikai Luo[1], and Bing Shi[1,2(✉)]

[1] School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan, China
{xushuai,lyk,bingshi}@whut.edu.cn
[2] Shenzhen Research Institute of Wuhan University of Technology, Shenzhen, China

Abstract. Online ride-hailing has become one of the most important transportation ways in the modern city. In the ride-hailing system, how to efficiently match passengers (orders) with vehicles and how to dispatch idle vehicles are key issues. In the online ride-hailing system, the ride-hailing platform needs to match riding orders with vehicles and dispatches the idle vehicles efficiently to maximize the social welfare. However, the matching and dispatching decisions at the current round may affect the supply and demand of ride-hailing in the future rounds since they will affect the future vehicle distributions in different geographical zones. In fact, vehicles in different zones at different times may have different values for the matching and dispatching results. In this paper, we use the vehicle value function to characterize the spatio-temporal value of vehicles in each zone and then use it to design the order matching and idle vehicle dispatching algorithm to improve the long-term social welfare. We further run experiments to evaluate the proposed algorithm. The experimental results show that our algorithm can outperform benchmark approaches in terms of the social welfare, and can also achieve effective utilization of idle vehicles and thus improve the service ratio.

Keywords: Ride-hailing · Order matching · Idle vehicle dispatching · Social welfare

1 Introduction

As the quick development of ride-hailing business, various online ride-hailing platforms have emerged, such as DiDi and Uber. In fact, the annual volume of passengers transported by DiDi has exceeded 10 billion.[1] The market value of the global online ride-hailing business is expected to grow to $285 billion by 2030. In such a business, the ride-hailing platform needs to match riding orders with vehicles and dispatch idle vehicles efficiently in order to improve the profit.

Specifically, the ride-hailing platform needs to match vehicles with orders while dispatching idle vehicles to the potential high-demanding zones to avoid

[1] http://news.cctv.com/2020/10/26/ARTIRbGKnKHCeLzgSAltRgwJ201026.shtml.

G. Memmi et al. (Eds.): KSEM 2022, LNAI 13370, pp. 289–301, 2022.
https://doi.org/10.1007/978-3-031-10989-8_23

randomly exploring potential riding orders periodically (i.e., over multiple rounds). The ride-hailing platform should maximize the long-term social welfare[2] (i.e., the sum of social welfare of all rounds), instead of maximizing the social welfare of one round. However, the current matching and dispatching decisions may affect the future vehicle distributions, and thus affect the future matching and dispatching, which may affect the overall social welfare. Therefore, we need to consider the decision of current round on the future impacts when designing the order matching and idle vehicle dispatching algorithm to maximize the long-term social welfare. In more detail, historical information about matching and the corresponding social welfare can provide the implicit information about how much value the vehicle can provide in the spatial-temporal state, which can provide some insights for designing the matching and dispatching algorithm. Therefore, we design a vehicle value function to characterize the future value the vehicle can provide in the spatial-temporal state, and then use this value function to design the order matching and idle vehicle dispatching algorithm. In so doing, our algorithm can take into account the impacts of current decisions on the future rounds, and thus can improve the long-term social welfare.

In more detail, this paper advances the state of the art in the following ways. Firstly, we design a vehicle value function to characterize the future value the vehicle can provide. Then we consider the dispatching of idle vehicle to a zone as a virtual order. In so doing, we combine the order matching problem and idle vehicle dispatching problem as a whole order matching problem. We then convert the order matching problem to a bipartite graph maximum weight matching problem with the vehicle values as the edge weights. In so doing, we can complete the matching and dispatching quickly, and can avoid the issue that vehicles are concentrated in some zones. We run experiments to evaluate the proposed algorithm. The experimental results show that the proposed algorithm can outperform benchmark approaches in terms of the long-term social welfare. It can also improve the service ratio and achieve an effective utilization of idle vehicles.

The rest of this paper is organized as follows. We introduce the related work in Sect. 2. We then describe basic settings in Sect. 3, and introduce the proposed algorithms in Sect. 4. We provide experimental analysis in Sect. 5 and conclude the paper in Sect. 6.

2 Related Work

There exist many works about ride-hailing, especially in the order matching and idle vehicle dispatching issues [9,14]. For the order matching problem, the ride-hailing platform usually matches vehicles with orders to maximize profit, maximize order service ratio or minimize the travel distance. For maximizing profit, Cheng et al. [2] proposed a queueing theory-based order matching framework, while combining demand forecasts with predicted idle time slots to maximize the expected profits of the platform for each round. For maximizing the order

[2] In this paper, we assume that vehicles belong to the ride-hailing platform, and thus the social welfare consists of the profits of the ride-hailing platform and passengers.

service ratio, Garaix et al. [3] proposed an iterative algorithm to solve the order matching problem to maximize the order service ratio. For minimizing vehicle travel distance, Cao et al. [1] proposed a large-scale many-to-many matching algorithm based on spatial pruning techniques in the shared mobility environment to minimize the detour distance of vehicles.

There also exist some works about dispatching idle vehicles. Holler et al. [5] proposed a deep reinforcement learning based approach for solving the order matching and idle vehicle dispatching problems to maximize the profits of all vehicles. Haliem et al. [4] proposed a route planning framework based on demand forecasting and reinforcement learning to dynamically generate optimal routes. Liang et al. [6] integrated both real-time order matching and idle vehicle dispatching within a Markov decision process framework to increase drivers' profits while reducing the waiting time of passengers. Shou et al. [10] used Markov decision process to model an idle vehicle finding passengers and used inverse reinforcement learning to solve the reward function of the model.

However, to the best of our knowledge, existing works usually did not consider the impacts of current decision of order matching and idle vehicle dispatching on the future rounds, and did not maximize the long-term social welfare. Furthermore, existing works usually analyzed the order matching and dispatching problems separately. In this paper, we address the above issues by taking the spatio-temporal value of vehicles into account, and consider the order matching and idle vehicle dispatching as a whole to maximize the long-term social welfare.

3 Basic Settings

In this paper, we assume that all vehicles are managed by the online ride-hailing platform. In the ride-hailing system, firstly, passengers submit riding orders to the platform. Then the platform matches orders with available vehicles, and provide dispatching suggestions for idle vehicles.

Furthermore, we divide the entire time into T time steps (i.e., rounds) $\mathcal{T} = \{1, 2, \cdots T\}$. The geographical zone where passengers and vehicles are located is constructed as a road network, which is defined as follows:

Definition 1. ***Road Network.*** *The road network is defined as a weighted graph $G = (L, E)$ where L is the set of nodes and E is the set of edges on the road network. We use $dis\,(l_i, l_j)$ to represent the shortest path from node l_i to l_j. $dis\,(l_i, l_j)$ is also used to denote the weight of edge $\langle l_i, l_j \rangle$.*

The riding order is defined as follows:

Definition 2. ***Order.*** *An order $o \in \mathcal{O}$ is defined as a tuple $(l_o^p, l_o^d, t_o^r, t_o^w, val_o)$, where l_o^p, l_o^d is the pick-up and drop-off locations of order o respectively, t_o^r is the time when the order o is raised, t_o^w is the maximum time that a passenger in order o is willing to wait for the riding service, val_o is the highest price the passenger is willing to pay for the service, which can be regarded as the real value of this order for the passenger.*

Noted that in the realistic scenarios, passengers do not need to express the above value information when submitting orders. However, such information will be used to maximize social welfare. Therefore, similar to existing work [15,16], we assume that the passenger is required to submit this value for the riding service. Note that passengers may not reveal this information truthfully in order to obtain more profits. How to prevent passengers from dishonestly revealing their values is beyond the scope of this paper. In addition, we assume that when the order o is not matched within t_o^w time, the passenger is not willing to wait and the order will be cancelled.

Definition 3. ***Vehicle.*** *A vehicle $v \in \mathcal{V}$ is defined as a tuple (l_v, c_v), where l_v is the current position and c_v is the unit travel cost.*

Note that different types of vehicles may have different unit travel costs. When the platform matches orders with vehicles, it can only match the order with feasible vehicle, which is defined as follows.

Definition 4. ***Feasible Vehicle.*** *For an order o, a feasible vehicle v must serve a passenger before the maximum waiting time t_o^w, that is, $dis(l_v, l_o^p) \leq V_{avg} \cdot t_o^w$, where $dis\,(l_v, l_o^p)$ is the distance from the current position l_v of the vehicle to the pick-up position l_o^p of order o, and V_{avg} is the average speed of the vehicle.*

We now introduce the social welfare of the ride-hailing system, which consists of the profits of the passengers and the platform. The passenger's profit u_o is the passenger's true value for order o minus its payment for the riding service:

$$u_o = \begin{cases} val_o - p_o, & o \in O^w \\ 0, & o \notin O^w \end{cases} \tag{1}$$

where p_o is the price paid to the platform and $\mathcal{O}^w$ is the set of matched orders. When order o is not matched, the passenger's profit $u_o = 0$. The platform's profit is the sum of passengers' payments for the matched orders minus the costs of the corresponding vehicles to complete these orders over all time steps: $U_p = \sum_{t=1}^{T} \sum_{o \in \mathcal{O}_t^w} (p_o - C^o_{\Theta_t(o)})$, where $\mathcal{O}_t^w$ is the set of matched orders at time step t, Θ_t denotes the matching results at time step t, and $\Theta_t(o) = v$ means that order o is matched with vehicle v. $C_{\Theta_t(o)}$ is the cost of vehicle $\Theta_t(o)$ completing order o, which is:$C^o_{\Theta_t(o)} = \left(dis\left(l_{\Theta_t(o)}, l_o^p\right) + dis\left(l_o^p, l_o^d\right)\right) \cdot c_{\Theta_t(o)}$.

We now give the definition of social welfare. Note that in this paper, we consider the long-term social welfare, which is the sum of the profits of all participants over the whole time steps, i.e., the summary profits of the platform and passengers, which is:

$$\begin{aligned} SW &= \sum\nolimits_{t=1}^{T} \left(\sum\nolimits_{o \in \mathcal{O}_t^w} (val_o - p_o) + \sum\nolimits_{o \in \mathcal{O}_t^w} \left(p_o - C^o_{\theta_t(o)}\right) \right) \\ &= \sum\nolimits_{t=1}^{T} \sum\nolimits_{o \in \mathcal{O}_t^w} \left(val_o - C^o_{\Theta_t(o)}\right) \end{aligned} \tag{2}$$

4 The Algorithm

In this paper, we intend to maximize the long-term social welfare over all time steps. Therefore, we need to consider the impacts of current decision on the future matching. We design a vehicle state value function, which implies the ability of vehicles to make social welfare in different spatio-temporal states. Then based on the vehicle value function, we design the order matching and idle vehicle dispatching algorithm.

4.1 Vehicle Value Function

The vehicle value function shows the potential social welfare the vehicle can make in the future in the current spatial-temporal state, which is:

Definition 5. ***Vehicle Value Function.*** *The vehicle value function is* $V(t, g, c)$*, where* $t \in \mathcal{T}$ *is the time step,* $g \in G$ *is the zone index at which the vehicle is located, and* c *is the vehicle unit travel cost.*

At each time step, the platform collects order information and makes decisions based on the current vehicle states, including whether the vehicle is matched with the order, whether the vehicle is stationary, or whether the vehicle is idle and dispatched to some place. Then the platform computes the social welfare of the current time step and enter into the next step. In such a multi-round matching process, we can capture how the current vehicle state can affect the future social welfare, i.e., the vehicle value function. This process is a sequential decision process, and thus we can model it as a Markov Decision Process (MDP) [11], and then compute the state value function by value iteration.

In the following, we give the description of MDP $M = \langle S, A, P, r, \gamma \rangle$.

State: The state of each vehicle is defined as a tuple $s = (t, g, c) \in S$, which is the vehicle value function.

Action: The action is $a \in A = \{a_1, a_2, a_3\}$, where a_1 means that the platform matches an order with a vehicle, a_2 means that the vehicle is stationary, and a_3 means that the platform dispatches an idle vehicle to an adjacent zone.

Reward: The reward r is the profit of the passengers and the platform when the action is taken. Note that the reward is 0 when the vehicle's action is stationary and negative (caused by the vehicle's travel cost) when the vehicle's action is dispatched. The reward value r is calculated as: $r = val_o - C_{\Theta(o)}$, where val_o is the passenger's value for an order o and $C_{\Theta(o)}$ is the cost required for the vehicle $\Theta(o)$ to complete the order o. For an order which lasts for T time steps, the cumulative reward R_γ is: $R_\gamma = \sum_{t=0}^{T-1} \gamma^t \frac{r}{T}$, where γ is a discount factor that decreases the impact of the past rewards, and is set to 0.9.

In this paper, we solve the MDP by using value iteration. The platform collects historical matching data to construct a historical state transition tuple $D = \{(s_i, a_i, r_i, s_i')\}$, which means that the agent acts a_i in the state s_i to

obtain an instant reward r_i and transfer to the next state $s_i{}'$. Since different types of vehicles may have different unit travel cost, we use cost c to represent the type information of the vehicle, and therefore the state transition information of the same type of vehicle constitutes a value function data set. Referring to existing work [13], we assume that the online policy generating the state transfer data remains constant during the phase of learning the value function. In the following, we will omit the policy parameter π. State transition involves three actions, which are matching orders, stationery and idle vehicle dispatching.

When the action is to match an order, the vehicle receives an immediate reward R_γ and makes a state transfer. The Temporal difference (TD) update rule is:

$$V(s) = V(s) + \alpha \left[R_\gamma + \gamma V(s') - V(s)\right] \tag{3}$$

where $s = (t_0, g, c)$ is the state of the vehicle at the current time step, $s' = (t_3, g_{ld}, c)$ is the state of the vehicle after completing the matching order, in which t_3 is the time step when the passenger reaches the destination and g_{ld} is the index of the order destination zone.

When the action is being stationary, the immediate reward of the agent is 0. The TD update rule is:

$$V(s) = V(s) + \alpha \left[0 + \gamma V(s'') - V(s)\right] \tag{4}$$

Since the vehicle takes a stationary action, the position of the vehicle does not change, i.e., $s'' = (t_1, g, c)$.

When the action is to dispatch idle vehicle, we construct a virtual order where the value is 0, the origin of the order is g, and the destination of the order is one of the neighboring zones of g. The TD update rule is:

$$V(s) = V(s) + \alpha \left[R_\gamma{}' + \gamma V(s''') - V(s)\right] \tag{5}$$

where $s''' = (t_2, g''', c)$ is the state of the vehicle after the dispatching is completed, in which t_2 is the time step when the idle vehicle is dispatched to the destination, $g''' \in g_{near}$ is a neighboring zone of g.

Next, we describe how to compute the vehicle value function V. The platform first collects historical state transfer data, and then uses a dynamic programming based value iteration algorithm to backward recursively calculate value $V(s_i)$ in each state to obtain the vehicle value function $V(s)$. The details are shown in Algorithm 1.

4.2 Order Matching and Idle Vehicle Dispatching Algorithm Based on Value Function

After obtaining the vehicle value function, we now describe how to use this function to design the order matching and idle vehicle dispatching algorithm. The order matching and idle vehicle dispatching problem to maximize the social welfare is actually a bipartite graph maximum weight matching problem. At each time step t, the set of orders O_t and the set of vehicles V_t are two disjoint sets

Algorithm 1: Dynamic Programming based Value Iteration Algorithm(DPVI).

Input: History state transfer tuple $D = \{(s_i, a_i, r_i, s_i')\}$, where each state $s_i = (t_i, g_i, c)$ consists of the current time step, geographic zone index and cost of the vehicle.

Output: Vehicle value function V

1 **Initialize:** $\forall s, V(s) \leftarrow 0, N(s) \leftarrow 0$;
2 **for** $t = T - 1$ *to* 0 **do**
3 $\quad D_t \leftarrow \{(s_i, a_i, r_i, s_i') | \forall s_i = (t_i, g_i, c),\ t_i = t\}$;
4 $\quad$ **foreach** $(s_i, a_i, r_i, s_i') \in D_t$ **do**
5 $\quad\quad N(s_i) \leftarrow N(s_i) + 1$;
6 $\quad\quad V(s_i) \leftarrow V(s_i) + \frac{1}{N(s_i)} \left(\gamma^{\Delta t(a_i)} V(s_i') + R_\gamma(a_i) - V(s_i)\right)$
7 $\quad$ **end**
8 **end**
9 **return** V

of vertices of the bipartite graph. The weight of edge $\langle o, v \rangle$ is the difference ΔV of the value of vehicle v after completing order o, which is: $\Delta V = \gamma^{\Delta t_{o,v}} V(s') - V(s) + R_\gamma$, where s is the state when the vehicle v is matched with the order o and s' is the state when the vehicle v delivers the passenger corresponding to the order o to the destination, $\Delta t_{o,v}$ is the time required for vehicle v to complete this trip, and R_γ is the cumulative reward. The details of the value function-based order matching and idle vehicle dispatching algorithm are shown in Algorithm 2, which is named VFOMIVD for short.

In Algorithm 2, line 1 initializes the set of matched orders $\mathcal{O}^w$, the matching result Θ, the social welfare SW. For each time step of order matching and idle vehicle dispatching, lines 3 and 4 initialize the set of orders $\mathcal{O}_t$ and vehicle set $\mathcal{V}_t$. The order collection $\mathcal{O}_t$ contains orders submitted by passengers at the current time step and orders that have not been matched in previous steps and are still within the maximum waiting time. The vehicle set $\mathcal{V}_t$ includes vehicles that are not serving orders at the current step, i.e., idle vehicles. Lines 5 to 17 construct the bipartite graph. For each matching pair $\langle o, v \rangle$, the platform calculates the difference ΔV and use it as the weights of the edges of the bipartite graph.

Note that only the matching pair $\langle o, v \rangle$ with $\Delta V > 0$ is inserted into the bipartite graph, while if vehicle v cannot serve the passenger corresponding to order o within the maximum waiting time, its corresponding $\Delta V = 0$. For each idle vehicle v, the platform fictitiously creates several virtual orders o' from the current location of vehicle v to its neighboring zone $g \in g_{near}$. In so doing, the algorithm combines the order matching and idle vehicle dispatching as a whole. In the order matching, the platform calculates the difference $\Delta V'$ of its corresponding state value and insert it into the bipartite diagram. In lines 18, the platform solves the bipartite graph using the Kuhn-Munkres algorithm [8]. Line 19 calculates the social welfare for the current time step, and lines 20 to 22 record the results for the current time step. Finally, the platform updates the

Algorithm 2: Value Function based Order Matching and Idle Vehicle Dispatching Algorithm(VFOMIVD).

Input: Iterate through the set of orders $\mathcal{O}$, the set of vehicles $\mathcal{V}$, and the vehicle state value function V.
Output: The set of matched orders $\mathcal{O}^w$, matching result Θ, social welfare SW.

Initialize: $\mathcal{O}^w \leftarrow \emptyset$, $\Theta \leftarrow \emptyset$, $SW = 0$;
for $t = 0$ *to* T **do**
  $\mathcal{O}_t \leftarrow \{o \in \mathcal{O} | t_o^r + t_o^w \leq t + \Delta t\}$;
  $\mathcal{V}_t \leftarrow select\_empty\_vehicles\,(\mathcal{V}, t)$;
  Initialize the bipartite graph $G = (\mathcal{O}_t, \mathcal{V}_t, E)$;
  foreach $\langle o, v \rangle \in \mathcal{O}_t \times \mathcal{V}_t$ **do**
    calculate the state value difference ΔV corresponding to $\langle o, v \rangle$;
    if $\Delta V > 0$ **then**
      assign the weights of edges $\langle o, v \rangle$ to ΔV and insert them into the bipartite graph;
  end
  foreach $v \in \mathcal{V}_t$ **do**
    foreach $g \in g_{near}$ **do**
      calculate the difference $\Delta V'$ of the state value of the vehicle v from the current position to the neighboring zone g ;
      if $\Delta V' > 0$ **then**
        assign the weights of edges $\langle o', v \rangle$ to $\Delta V'$ and insert them into the bipartite graph G;
    end
  end
  $\mathcal{O}_t^w, \Theta_t, V_t \leftarrow KM\,(G)$;
  $SW_t \leftarrow calculate\_social\_welfare\,(\mathcal{O}_t^w, \Theta_t)$;
  $\mathcal{O}^w \leftarrow \mathcal{O}^w \cup \mathcal{O}_t^w$;
  $\Theta \leftarrow \Theta \cup \Theta_t$;
  $SW \leftarrow SW + SW_t$;
  Update vehicle trip information in Θ_t;
  $\mathcal{O} \leftarrow \mathcal{O} \backslash \mathcal{O}_t^w$;
end
return $\mathcal{O}^w, \Theta, SW$

trips of the vehicles that have been matched with orders and eliminates the set of orders $\mathcal{O}_t^w$ that have been matched from the set of orders $\mathcal{O}$.

5 Experimental Analysis

In this section, we run experiments to evaluate the proposed algorithm based on the real taxi order data in New York city, which has been used by a large number of related works [9,13,15].

(1) Order Data. We collect taxi order data on Manhattan Island in June 2019 from New York City Taxi and Limousine Commission (TLC).[3] Each taxi order data contains departure and destination locations, order starting time, trip fare and trip mileage.
(2) Map Data: We use Manhattan taxi zone map provided by TLC as the map data, and we number each zone.
(3) Fuel Consumption Data. We cannot find the travel cost data of New York taxis. Instead, we collect the urban vehicle fuel consumption data of type M1 and M2 from China Automobile Fuel Consumption Query System[4] of the Ministry of Public Information of China to compute the vehicle cost in the below.
(4) The Shortest Path Cache. To ensure that the distance between any two nodes can be quickly queried during experiments, we pre-build the cache of the shortest path matrix and the shortest path distance matrix.
(5) Order Data Processing. We remove some orders with noises (i.e. orders with invalid fares, zero trip milage and so on). We eliminate the order data in those isolated zones. We count the number of orders per hour each day and we find that the number of orders on weekdays and weekends varies greatly over time. For consistency, we use the order data of weekdays (20 days in total) in the evaluation.

5.1 Experimental Settings

The number of orders for each hour of a weekday is also significantly different, and the performance of the algorithm in the peak period is more important. Furthermore, a large number of order data in the peak time period is also helpful to generate the vehicle value function. Therefore, we choose the time period (19:00 to 21:00) in the weekday for the evaluation. We now compute the average number of order data from 19:00 to 21:00 in these 20 days, and randomly choose the average number of one day as the order input. For other parameters, we set the length of each time step as 60 s. The maximum waiting time for passengers is chosen randomly from {3 min, 4 min, 5 min, 6 min, 7 min, 8 min}. The average vehicle travel speed V_{avg} is set to 7.2 mph. For each vehicle, the unit travel cost is randomly selected from $\{6, 8, 10\} \times 2.5/6.8/1.6$\$/km.[5] The initial location of the vehicle is randomly selected in Manhattan taxi zone map. In the experiments, we try different numbers of vehicles, which is increased from 1500 to 3000. For each experiment, we repeat it for 10 times and compute the average result.

[3] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.
[4] https://yhgscx.miit.gov.cn/fuel-consumption-web/mainPage.
[5] The basic fare of New York taxi is 2.5$ per mile, the average fuel consumption is 6.8 L per one-hundred kilometres according to the above fuel consumption data, and 1.6 is the converting factor between mile and kilometre.

5.2 Evaluation of Order Matching and Idle Vehicle Dispatching Algorithm

In this section, we evaluate the proposed VFOMIVD algorithm against some benchmark algorithms.

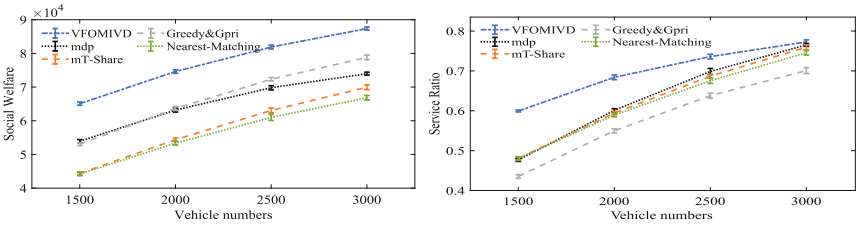
Benchmark Algorithms and Metrics

(1) **mdp** [13]. The mdp algorithm also utilizes the vehicle value function to guide the order matching, but its vehicle state does not consider the variability of vehicles in terms of cost and also does not consider the dispatching of idle vehicles.
(2) **mT-Share** [7]. The mT-Share algorithm intends to minimize the vehicle travel cost. The order matching in the mT-Share algorithm uses a greedy algorithm to match orders with vehicles with the least extra cost.
(3) **Nearest-Matching**. The Nearest-Matching algorithm is widely used in industry (i.e., Uber), where the platform matches orders with the nearest vehicles.
(4) **Greedy&GPri** [16]. The Greedy&GPri algorithm greedily matches orders with vehicles with the highest social welfare iteratively and adopts a critical value-based pricing algorithm. The reason of choosing this greedy method is that existing research has showed that greedy method can perform well in the crowdsourcing tasks [12].

In terms of the evaluation metrics, in addition to the social welfare, we also investigate service ratio, which is the ratio of the number of matched orders to the total number of orders submitted by passengers.

Analysis of Experimental Results

The experimental results are shown in Fig. 1(a). We find that as the number of vehicles increases, the social welfare increases since more orders are served. We find that VFOMIVD algorithm achieves the highest social welfare. We also find that greedy based method Greedy&Gpri can perform well. We also look into the service ratio in Fig. 1(b). We find that the VFOMIVD algorithm achieves the maximum service ratio. Although the service ratio of algorithms such as mdp is close to that of the VFOMIVD algorithm when the number of vehicles increases, the social welfare of the VFOMIVD algorithm is still the largest. This may imply in our algorithm, vehicles are more likely to converge to zones where more social welfare is generated.



(a) Social welfare. (b) Service ratio.

Fig. 1. Experiments of order matching and idle vehicle dispatching algorithm.

5.3 Evaluation of Idle Vehicle Dispatching

In this section we further analyze the effectiveness of the idle vehicle dispatching algorithm. In order to evaluate the performance of the idle vehicle dispatching algorithm, we combine different benchmark dispatching algorithms with the order matching module of VFOMIVD to generate the benchmark algorithms.

Benchmark Algorithms and Metrics

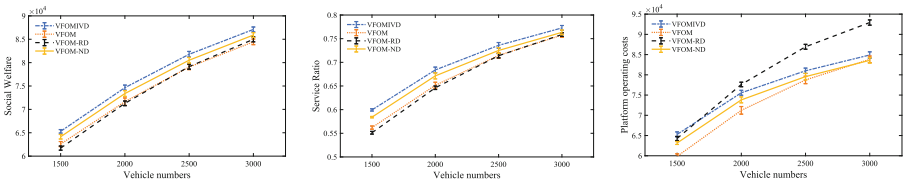
1. **VFOM**. We remove the idle vehicle dispatching module of VFOMIVD algorithm (lines 12 to 19 in Algorithm 2) and keep the order matching module.
2. **VFOM-RD**. This algorithm adds random dispatching into VFOM algorithm, which randomly dispatches idle vehicles to their neighboring zones. Random dispatching has been used in related works [16].
3. **VFOM-ND**. It adds the nearest dispatching to the VFOM algorithm, which is a common dispatching algorithm used by companies (e.g., Uber) to dispatch idle vehicles to the nearest neighboring zone [3].

In addition to evaluating the performance on social welfare and service ratio, we consider one more metric to evaluate the idle vehicle dispatching algorithm, which is the platform operating cost, consisting of the costs of all served orders and idle vehicles travelling to dispatched zones over the whole time steps.

Analysis of Experimental Results

The experimental results are shown in Fig. 2(a). We find that VFOMIVD algorithm achieves the largest social welfare. As the number of vehicles increases, the social welfare obtained by all algorithms increases. We also find that the social welfare of VFOM algorithm (where no dispatching algorithm is used) and VFOM-RD is similar. This may imply that random dispatching is not beneficial for the utilization of idle vehicles. From Fig. 2(b), we find that the VFOMIVD algorithm achieves the maximum service ratio. This means that after using the proposed dispatching algorithm, the platform can serve more orders, and thus can achieve the maximum social welfare. From Fig. 2(c), we find that the platform operating cost of the VFOMIVD algorithm is higher than the VFOM algorithm. However, from Figs. 2(a) and 2(b) we find that the social welfare and

service ratio of the VFOMIVD algorithm are higher. This may imply that the increased operating cost of our algorithm is caused by dispatching idle vehicles to zones with more riding demands, and thus can serve more orders and bring more social welfare.



(a) Social welfare. (b) Service ratio. (c) Platform operating cost.

Fig. 2. Experiments of idle vehicle dispatching.

In summary, we find that because VFOMIVD algorithm takes into account the spatio-temporal value of vehicles and dispatches vehicles to zones where more vehicles are needed, it can utilize idle vehicles to serve more riding orders, and thus can increase social welfare.

6 Conclusion

In this paper, we proposed an order matching and idle vehicle dispatching algorithm to maximize the long-term social welfare in the ride-hailing system. By considering the impacts of current order matching and idle vehicle dispatching decisions on the future rounds, we design a vehicle value function, which can characterize the ability of the vehicle to make social welfare in the future spatio-temporal state. Based on the vehicle value function, we design the order matching and idle vehicle dispatching algorithm. Finally, we run extensive experiments to evaluate the proposed algorithm. The experimental results show that our algorithm can help online ride-hailing platforms to dispatch idle vehicles efficiently to improve the utilization of idle vehicles, and thus can increase the service ratio and social welfare.

Acknowledgement. This paper was funded by the Shenzhen Fundamental Research Program (Grant No. JCYJ20190809175613332), the Humanity and Social Science Youth Research Foundation of Ministry of Education (Grant No. 19YJC790111), the Philosophy and Social Science Post-Foundation of Ministry of Education (Grant No. 18JHQ060) and the Fundamental Research Funds for the Central Universities (WUT:2022IVB004).

References

1. Cao, B., Hong, F., Wang, K., Xu, J., Zhao, L., Fan, J.: Uroad: an efficient method for large-scale many to many ride sharing matching. J. Comput. Res. Dev. **56**(4), 866 (2019)

2. Cheng, P., Feng, C., Chen, L., Wang, Z.: A queueing-theoretic framework for vehicle dispatching in dynamic car-hailing. In: 35th International Conference on Data Engineering, pp. 1622–1625 (2019)
3. Garaix, T., Artigues, C., Feillet, D., Josselin, D.: Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation. Comput. Oper. Res. **38**(10), 1435–1442 (2011)
4. Haliem, M., Mani, G., Aggarwal, V., Bhargava, B.: A distributed model-free ride-sharing approach for joint matching, pricing, and dispatching using deep reinforcement learning. IEEE Trans. Intell. Transp. Syst. **22**, 1–12 (2021)
5. Holler, J., et al.: Deep reinforcement learning for multi-driver vehicle dispatching and repositioning problem. In: 2019 IEEE International Conference on Data Mining, pp. 1090–1095 (2019)
6. Liang, E., Wen, K., Lam, W.H., Sumalee, A., Zhong, R.: An integrated reinforcement learning and centralized programming approach for online taxi dispatching. IEEE Trans. Neural Netw. Learn. Syst. 1 (2021)
7. Liu, Z., Gong, Z., Li, J., Wu, K.: Mobility-aware dynamic taxi ridesharing. In: 36th International Conference on Data Engineering, pp. 961–972 (2020)
8. Munkres, J.: Algorithms for the assignment and transportation problems. J. Soc. Ind. Appl. Math. **5**(1), 32–38 (1957)
9. Sharma, S.K., Routroy, S., Yadav, U.: Vehicle routing problem: recent literature review of its variants. Int. J. Oper. Res. **33**(1), 1–31 (2018)
10. Shou, Z., Di, X., Ye, J., Zhu, H., Zhang, H., Hampshire, R.: Optimal passenger-seeking policies on e-hailing platforms using Markov decision process and imitation learning. Transp. Res. Part C Emerg. Technol. **111**, 91–113 (2020)
11. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2018)
12. Tong, Y., She, J., Ding, B., Chen, L., Wo, T., Xu, K.: Online minimum matching in real-time spatial data: experiments and analysis. Proc. VLDB Endow. **9**(12), 1053–1064 (2016)
13. Xu, Z., et al.: Large-scale order dispatch in on-demand ride-hailing platforms: a learning and planning approach. In: 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 905–913 (2018)
14. Yi, X., Yongxin, T., Wei, L.: Recent progress in large-scale ridesharing algorithms. J. Comput. Res. Dev. **57**(1), 32 (2020)
15. Zhao, H., Xiao, M., Wu, J., Liu, A., An, B.: Reverse-auction-based competitive order assignment for mobile taxi-hailing systems. In: 2019 Database Systems for Advanced Applications, pp. 660–677 (2019)
16. Zheng, L., Cheng, P., Chen, L.: Auction-based order dispatch and pricing in ridesharing. In: 35th International Conference on Data Engineering, pp. 1034–1045 (2019)