Auction-based Order Dispatch and Pricing in Ridesharing

Libin Zheng, Peng Cheng*, Lei Chen
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology, Hong Kong, China
Email: {lzhengab, pchengaa, leichen}@cse.ust.hk

***Abstract*—Ridesharing plays a more and more important role in modern transport. In this paper, we propose solutions for the bonus-offering scenario of ridesharing platforms (service providers). When vehicles are in shortage, requesters are allowed to offer bonus so that their orders can get prioritized in the dispatch process. To enable self-motivated bonus bidding of requesters, we devise an auction mechanism, where requesters are supposed to submit their bids truthfully and the platform conducts order dispatch and pricing. Our goal is to maximize the overall utility of the auction, while ensuring desirable auction properties such as truthfulness and individual rationality. To realize that, we propose a greedy and a ranking approach for order dispatch and their corresponding pricing strategies. Extensive experiments on real data suggest that the ranking approach is both effective and efficient.**

***Index Terms*—ridesharing, car-hailing, auction, order dispatch**

I. Introduction

Ridesharing has gained great popularity over the past few years and becomes more and more common in the transportation market. In Didi Chuxing[1], the largest ridesharing service provider in China, the number of ridesharing orders reaches 2.4 million per day. Besides, among car-hailing service users, more than half of them are willing to share their rides [1].

Vehicle shortage is not uncommon in popular ridesharing platforms like Didi Chuxing and Uber[2]. For example, it can happen during peak ordering times. The shortage of vehicle disables the platform from dispatching all orders immediately, and some requesters would have to wait if their orders are pended. This can further lead to cancellation of orders. According to [2], the order cancellation rate in Didi Chuxing can be more than 20%. In practice, requesters who are anxious and not cost-sensitive, may wish to pay more to get dispatch priority. Therefore, an user interface can be desired by such requesters to announce their bonus offers. With such an interface, the platform can also make more earnings from requesters' mark-up.

In response to the aforementioned scenario, in this work, we propose an auction-based order dispatch and pricing mechanism for ridesharing platforms. In this auction mechanism, the requesters (riders) act as buyers. They submit origins & destinations and bids to the platform server. With bids, the requesters claim the amount of money they would like to pay to take the service. The platform acts as the auctioneer, selling the shard-rides to requesters. With respect to their bids, the platform computes a dispatch, and decides the final payments of dispatched requesters.

Use case 1. *In Didi Chuxing, during peak ordering periods, there is a bonus interface. In this interface, requesters would be asked to pay some bonus if they wish to prioritize their orders. In the current practice, the requesters decide whether to pay the bonus set by the platform. The problem lies in that the platform sets the bonus amount without knowing requesters' valuation in mind. Suppose that for a requester r_j, his valuation of the order prioritization service is z_j, and the bonus amount set by the platform is y. If $y > z_j$, the requester would not offer the bonus because it exceeds his/her valuation, which leads to a monetary loss as z_j. If $z_j > y$, the requester offers bonus y, but still cause a monetary loss as $z_j - y$. This bonus interface can be made more self-motivated and flexible by applying our auction mechanism, where requesters can report their valuation as bids. Each requester can claim his/her own bonus offer instead of simply answering a yes-or-no charge question. Receiving requesters' orders and bids, the platform computes a dispatch where orders with higher bonus should get higher priority. By guaranteeing some properties of the auction mechanism, requesters are supposed to offer bids exactly as z_j's, which would not cause monetary loss.*

To the best of our knowledge, existing works of ridesharing have not yet addressed the aforementioned scenario. Most existing works focus on order dispatch [3]–[11], with the optimization objective to minimize the overall travel distance. Pricing the dispatched orders is not discussed in these works. Fewer research works [12]–[14] have considered the order pricing issue. For example, [12] suggests dividing the travel cost of the shared trip evenly among the collaborative riders. In these works, however, the requesters provide no information on their desired payments. They have no control over their payments, let alone deciding the bonus. Therefore, they cannot be applied to the bonus scenario either.

To realize the auction mechanism, we propose order dispatch and pricing algorithms, which correspond to the winner selection and the payment computation method respectively in a classical auction [15]. Though auction mechanism has been carefully studied in many fields [16]–[18], applying it to the

*Peng Cheng is the corresponding author.
[1]https://www.didiglobal.com/
[2]https://www.uber.com

TABLE I
SYMBOLS AND NOTATIONS

Symbol	Description
$V = \{v_i\}$	Set of vehicles
$R = \{r_j\}$	Set of requesters.
$\widehat{R}$	The set of dispatched requesters.
s_j , e_j	Origin and destination of order r_j.
rev_i	Revenue of v_i, i.e., platform's payment to v_i.
pl_i	Travel plan of v_i.
$cost_{i,j}$	Cost of dispatching r_j to v_i.
$u_{i,j}$	Utility of dispatching r_j to v_i.
$\bar{c}$	Capacity of vehicles.
D_i, T_i	Travel distance, travel time of v_i.
α_d	Travel cost per unit travel distance.
m , n	Number of requesters, number of vehicles.

ridesharing scenario has not yet been explored. The challenge lies in the combinatorial complexity of order dispatch in ridesharing, where different orders can be dispatched to the same vehicle. Besides, when dispatching orders to a vehicle, the utilities of different orders can affect one another depending on the closeness among their origins & destinations. In spite of these challenges, our proposed algorithms manage to maximize the overall utility of the auction, which is comprised of requesters' utility, drivers' utility, and platform's utility. The algorithms are required to guarantee a set of auction properties, including *truthfulness*, *individual rationality*, *profitability*, and *computational efficiency*, which are formally defined in Section II-B. These desired properties ensure that the auction would work well.

In summary, our contributions are listed as follows:

- We systematically introduce the auction mechanism, and the associated problems in Section II.
- We propose the greedy-based order dispatch method and its pricing strategy (GPri) in Section III.
- We propose the ranking-based order dispatch method and its pricing strategy (DnW) in Section IV.
- We conduct experiments on real data to investigate the performance of our proposed algorithms in Section V.

In addition to the contributions listed above, we discuss the related work in Section VI and conclude the paper in Section VII.

II. The Mechanism and Problem

In this section, we formally describe the auction-based ridesharing mechanism. We firstly introduce the participating entities of this mechanism in Section II-A. Then, in Section II-B, we formulate the problems of order dispatch and pricing, for which we need to devise solutions to implement the mechanism. The important notations and symbols in the rest of this paper are summarized in Table I.

A. Entity Models

Following most existing works [7], [8], [19]–[21], we adopt the round-based order dispatch model for our mechanism. In each round, the platform server operates on a set of pended orders and online vehicles to optimize some objectives.

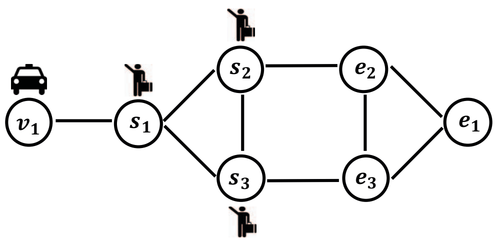


Fig. 1. A toy example.

Requesters act as buyers in the auction mechanism, submitting bids to take the shared rides. Each requester has a value in mind on how much the riding service is worth to him/her. For a requester r_j, we use val_j to denote his/her valuation of the service. Regarding val_j, r_j submits a bid, denoted as bid_j, to the platform, claiming how much he/she would like to pay to take the shared ride.

Definition 1 (Wasted time)**.** *The wasted time of a dispatched requester r_j is $wt_j + dt_j$, where wt_j is his/her waiting time during pick-up, and dt_j is his/her detour time during delivery, i.e., the actual travel time minus the least possible travel time.*

Definition 2 (Requester)**.** *A requester r_j is represented by a tuple: $< s_j, e_j, \theta_j, val_j, bid_j >$, where:*

- *s_j and e_j are the starting and ending geographical locations of its trip;*
- *val_j and bid_j are r_j's valuation of the service and bid to the platform respectively;*
- *θ_j is his/her maximum allowed amount of wasted time during pick-up and delivery.*

In this paper, the concepts of requester and order are used interchangeably.

An example of three requesters $r_{1\sim3}$ is shown in Figure 1, where each node represents a location and all the edge segments are of unit travel distance, requiring an amount of travel time as t_e. Suppose v_1 conducts its travel as $v_1 s_1 s_3 s_2 e_2 e_3 e_1$, where we omit the symbol '$\rightarrow$' between two nodes for simplicity. Then, for r_1, we have $wt_1 = t_e$ (the travel time between v_1 and s_1), and $dt_1 = 2t_e$, which equals the delivery time in the travel ($5t_e$) minus the shortest delivery time ($3t_e$ for $s_1 s_2 e_2 e_1$).

The platform acts as the seller in the mechanism, which provides the riding service to the requesters. It also acts as the auctioneer, who decides the winners among bidders (requesters) and their payments. To dispatch requesters, the platform maintains a set of vehicles.

Definition 3 (Vehicle)**.** *A vehicle v_i is represented by a tuple $< l_i, pl_i >$. l_i is its current location, and pl_i is its travel plan, which consists of the starting and ending locations of requesters dispatched to v_i. Besides, each vehicle has a capacity as $\bar{c}$.*

Definition 4 (Validity of travel plans.)**.** *The travel plan of v_i, pl_i, is valid if it does not violate any of the following*

constraints:
<u>Precedence constraint.</u> $\forall r_j$ assigned to v_i, s_j precedes e_j in pl_i.
<u>Capacity constraint.</u> $\forall v_i$, at any stage, the difference between numbers of finished pick-ups and drop-offs cannot exceed $\bar{c}$.
<u>Time constraint.</u> For any r_j assigned to v_i, his/her amount of wasted time cannot exceed θ_j, i.e., $wt_j + dt_j \leq \theta_j$.

An example of a vehicle can be found in Figure 1. Let us suppose pl_1 as $v_1 s_1 s_3 s_2 e_2 e_3 e_1$. If $\theta_1 = 2t_e$, this would be an invalid plan due to the violation of time constraint. This is because the wasted time of r_1 equals $3t_e$, larger than θ_1. A valid solution is to dispatch r_1 and r_3 to v_1 with pl_1 as $v_1 s_1 s_3 e_3 e_1$.

The platform server dispatches requesters to the vehicles, and makes payments to them. In this work, dispatching r_j to v_i means inserting s_j and e_j into pl_i, without violations of constraints in Definition 4. For payments to vehicles, without loss of generality, we assume platform's payment to a vehicle is proportional to its travel distance after it picks up the first rider. Let rev_i denote the revenue of v_i, then we have

$$rev_i = \beta_d * D_i\,, \tag{1}$$

where β_d is the payment to vehicle per unit travel distance and D_i is the travel distance of v_i during delivery.

B. Problem Formulation

In this section, we formulate the order dispatch problem and the order pricing problem. These two problems correspond to the winner selection problem and payment computation problem respectively in a classical auction [15]. We firstly define the utilities of the auction participants.

Definition 5 (Requester's utility)**.** *According to [22], the utility of a requester (bidder) in an auction, u_j, is equal to*

$$u_j = (val_j - pay_j) \cdot x_j\,,$$

where x_j equals 1 if the order of r_j is dispatched, or 0 otherwise. pay_j is his/her final payment determined by the platform. u_j is the difference between r_j's valuation of service and his/her final payment when $x_j = 1$. Subsequently, the utility of r_j would be negative (or positive) when he/she pays more (or less) than what he/she has expected.

Definition 6 (Platform's utility)**.** *The utility of a platform in an auction, U_{plf} equals*

$$U_{plf} = \sum_{r_j \in \widehat{R}} pay_j - \sum_{v_i} rev_i\,,$$

where $\widehat{R}$ denotes the set of dispatched requesters. Clearly the platform's utility is interpreted as its profit.

Remark. We use $r_j \in \widehat{R}$ and $x_j = 1$ interchangeably in this paper to represent r_j is dispatched.

Definition 7 (Driver's utility)**.** *The utility of a driver (vehicle) in an auction, u_i equals*

$$u_i = rev_i - \alpha_d D_i = (\beta_d - \alpha_d) D_i\,,$$

where α_d measures the labor & fuel cost per unit travel distance.

Definition 8 (Overall utility)**.** *The utility of truthful auction in ridesharing is defined as the sum of requesters' utilities, platform's utility, and drivers' utilities, i.e.,*

$$\begin{aligned} U_{auc} = &\left(\sum_{r_j \in \widehat{R}} (val_j - pay_j)\right) + \left(\sum_{r_j \in \widehat{R}} pay_j - \sum_{v_i} rev_i\right) + \\ &\left(\sum_{v_i} (rev_i - \alpha_d D_i)\right) \\ = &\sum_{r_j \in \widehat{R}} val_j - \alpha_d \sum_{v_i} D_i = \sum_{r_j \in \widehat{R}} bid_j - \alpha_d \sum_{v_i} D_i\,. \end{aligned} \tag{2}$$

We have $bid_j = val_j$ because of auction truthfulness. Truthfulness, which would be formally introduced later, suggests that requesters' best strategy (i.e., the one yielding maximum utility) is to bid their valuations. It is generally assumed in auction works [8], [15], [23] that bidders would bid their valuations if auction truthfulness is satisfied. As what would be presented later, the proposed algorithms satisfy truthfulness, so we replace val_j with bid_j in Equation (2)*.*

Remark. The defined overall utility in this work corresponds to the social welfare of an auction system in existing works [18], [23]–[26]. Pursuing the social welfare defends the benefits of all auction participants, ensuring their willingness in joining the auction. Therefore, maximizing the social welfare is the optimization objective in many existing works [18], [23]–[26].

Next we formally define the problems studied in this paper.

Definition 9 (Order dispatch problem)**.** *Given inputs of requester set R and vehicle set V, the order dispatch problem in auction-based ridesharing is to dispatch requesters $r_j \in R$ to vehicles $v_i \in V$ such that the overall utility (U_{auc} in Definition* (8)*) is maximized.*

The successfully dispatched requesters are the winning bidders in this auction. The losing (un-dispatched) requesters in a round can increase their bids in the next dispatch round. Remember that the system server operates in a round manner. Requesters can make different bids in different rounds.

We formulate the problems on top of bidding behaviors of requesters. However, requesters may not be able to bid a price when they have no knowledge about the common charge of their travel. Therefore, as shown in Use case 1, it is also possible that the platform first displays a base price to requester r_j, on top of which r_j bids the bonus he/she is willing to pay. Note that the proposed algorithms still work after this minor change.

From Equation (2), it can be found that U_{auc} is irrelevant to requesters' actual payments, i.e., pay_j's. However, to make sure that the auction mechanism functions well, we need to properly determine requesters' payments such that the auction possesses a set of desired properties. These properties ensure that *I.* requesters are willing to join the auction (*individual* *rationality*) and bid their valuations (*truthfulness*); *II.* the platform is willing to join the auction (*profitability*); *III.* the auction itself is efficient (*computational efficiency*).

Remark. Drivers' willingness can be easily guaranteed by making $\beta_d \geq \alpha_d$ (see Definition 7), which ensures positive utilities for them. Because $\beta_d \geq \alpha_d$ is generally assured in practice, we do not make it another property to be realized by our algorithms.

Definition 10 (Order pricing problem)**.** *The order pricing problem is to determine the payment for each dispatched requester (i.e., pay_j's) such that the auction mechanism satisfies the following properties: truthfulness, individual rationality, profitability and computational efficiency.*

We define the aforementioned properties in the following.

Definition 11 (Truthfulness)**.** *An auction is truthful if the best strategy (i.e., yields the maximum utility) of any requester is to bid his/her valuation. That is, let pay_j and pay'_j be the payments of r_j from bidding $bid_j = val_j$ and $bid'_j \neq val_j$, respectively. Then,*

$$\forall r_j \forall bid'_j, (val_j - pay_j) \cdot x_j \geq (val_j - pay'_j) \cdot x'_j \,.$$

Definition 12 (Individual rationality)**.** *An auction is individually rational if the utility of any requester is non-negative, i.e.,*

$$\forall r_j, u_j = (val_j - pay_j) \cdot x_j \geq 0 \,.$$

Note that the un-dispatched requesters are naturally individually rational as their utilities equal 0 ($x_j = 0$).

Definition 13 (Profitability)**.** *An auction is profitable if the utility of the platform is non-negative, i.e., $U_{plf} \geq 0$.*

Definition 14 (Computational efficiency)**.** *An auction is computationally efficient if the process of winner selection and payment computation can finish within polynomial time.*

In auction-based ridesharing, Definition 14 means the time complexities of the order dispatch and pricing algorithms are polynomial w.r.t. the input size. However, as what would be presented next, the *order dispatch problem* is NP-hard, which makes it unrealistic to find the optimal dispatch solution while keeping the auction efficient. Therefore, in the rest of this paper, we propose approximation solutions for order dispatch and devise corresponding pricing strategies.

NP-hardness of the problem. We prove the NP-hardness of the *order dispatch problem* with a reduction from the *0-1 knapsack problem* [27].

Theorem II.1. *The order dispatch problem is NP-hard.*

Proof sketch. Given a knapsack problem instance, we construct an order dispatch problem instance with only one vehicle. Besides, we map each item (in the knapsack problem) to an order with the item utility and the weight as the utility and the travel time respectively of delivering the order. Finally, we transfer the budget constraint of the knapsack problem to the wasted time constraint of the orders. Then, delivering an order is analogous to selecting an item in the knapsack problem instance. Please refer to our technical report[3] for the thorough proof. □

III. The Greedy Approach

As shown in Section II, the *order dispatch problem* is NP-hard. To find approximation solutions, we propose a greedy dispatch method in Section III-B. With respect to the greedy dispatch method, we propose a pricing method (GPri), aiming to assure the desired auction properties. We conduct theoretical analyses for these methods in Section III-C.

A. Preliminary of Approximation Methods

During order dispatch, we need to devise the travel plan for a vehicle when there are more than one dispatched requester. Given a set of pick-up and drop-off locations, travel route planning aims to minimize the travel distance of visiting them without violation of constraints in Definition 4.

In this paper, following [4], [10], [20], [21], [28], to dispatch a new order, we insert its locations (pick-up & drop-off) into a vehicle's travel plan such that the increased travel distance is minimized. As in [7], [9]–[11], [14], [20], [21], given two adjacent locations in the plan, we measure the travel distance regarding their shortest path. Then, the overall travel distance equals the sum of adjacent distances. Note that measures other than shortest path distance can also be adopted. For example, the average historical travel distance between the two locations. Our proposed algorithms still work and the theoretical properties still apply. This is because the distances among locations are the inputs to our algorithms, and we do not use assumptions on them in our algorithm design and analyses.

The size of the search space in this insertion-based algorithm is linear to the length of plan, which is much more efficient than enumerating all feasible plans. To dispatch an order, the algorithm has a time complexity of $\mathcal{O}(\bar{c}^2 q)$, where $\mathcal{O}(q)$ is the time cost of a shortest path query. Note that this insertion-based routing algorithm finds a suboptimal solution, but it is a common practice [4], [10], [20], [21], [28] to save computation complexity.

B. Requester Dispatch and Pricing

Greedy-based dispatch. Greedy-based order dispatch proceeds in two steps:

Initialization (Lines 2 $\sim$ 6). It traverses all requester-vehicle pairs to find the valid dispatches, and calculates the corresponding utilities, i.e., $u_{i,j}$'s. Valid dispatch means that pl_i remains valid after accommodating r_j.

One-by-one dispatch (Lines 7 $\sim$ 16). It keeps dispatching (r_{j*}, v_{i*}) which brings the maximum utility among the remaining valid dispatches. The utility $u_{i,j}$ is calculated as follows:

$$u_{i,j} = bid_j - \alpha_d * \Delta D_i(r_j) \,, \tag{3}$$

[3]http://www.cse.ust.hk/%7Elzhengab/auctionRide.pdf

Algorithm 1 Greedy-based order dispatch (Greedy)

```
Input: Requester set $R$, vehicle set $V$.
Output: Updated travel plans of vehicles.
1: $pool \leftarrow \phi$.
2: for all $(r_j, v_i) \in R \times V$ do
3:     if $(r_j, v_i)$ is valid then
4:         Add pair $(r_j, v_i)$ into $pool$ and calculate $u_{i,j}$.
5:     end if
6: end for
7: while $pool \neq \phi$ do
8:     $(r_{j*}, v_{i*}) \leftarrow \arg\max_{(r_j,v_i)\in pool} u_{i,j}$.
9:     If $u_{i,j} < 0$: break.
10:    Dispatch $r_{j*}$ to $v_{i*}$ and update $pl_{i*}$.
11:    $\forall v_i$, if $(r_{j*}, v_i) \in pool$, remove $(r_{j*}, v_i)$.
12:    for all $(r_j, v_{i*}) \in pool$ do
13:        Update $u_{i*,j}$ if it remains valid.
14:        Remove $(r_j, v_{i*})$ from $pool$ otherwise.
15:    end for
16: end while
```

Algorithm 2 Order pricing for greedy dispatch (GPri).

```
Input: $R$, $V$ and $r_h$.
Output: Price for $r_h$, $pay_h$.
1: $R' \leftarrow R \setminus r_h$; $pay_h \leftarrow bid_h$.
2: Obtain $\widehat{R}' = \{r_{j_k}\}$ by running Greedy on $V$ and $R'$.
3: $h\_cost \leftarrow \min_{(r_a=r_h,v_b)\in pool_{j_z}} \alpha_d * \Delta D_b(r_h)$.
4: if $pay_h > h\_cost$ then
5:     $pay_h \leftarrow h\_cost$.
6: end if
7: for all $j_k \in [j_1, j_z]$ do
8:     If $\{(r_a, v_b) \in pool_{j_k} | r_a \text{ is } r_h\} = \phi$: Break.
9:     $h\_cost \leftarrow \min_{(r_a=r_h,v_b)\in pool_{j_k}} \alpha_d * \Delta D_b(r_h)$.
10:    $pay_h \leftarrow \min\{pay_h, bid_{j_k} - cost_{j_k} + h\_cost\}$.
11: end for
12: Return $pay_h$.
```

where $\Delta D_i(r_j)$ is the travel distance increase during delivery, from inserting r_j into v_i's travel plan.

Order pricing for greedy dispatch. In the following, we devise an order pricing algorithm to decide the final payments (prices) of the dispatched requesters. To price r_h, as shown in Algorithm 2, firstly a new dispatch requester set $\widehat{R}'$ is obtained by running Greedy on V and $R \setminus r_h$ (Line 2). $\widehat{R}'$ is represented in a sequence: $r_{j_1}...r_{j_k}...r_{j_z}$, and r_{j_k} refers to the k-th dispatched requester in Greedy. Let V_{j_k} and $pool_{j_k}$ denote the vehicles and the valid requester-vehicle pairs respectively before dispatch of r_{j_k}. Besides, let $cost_{j_k}$ denote the travel cost of dispatching r_{j_k}.

The idea of Algorithm 2 is to compare r_h to the dispatched requesters in $\widehat{R}'$, and finds the minimum bid for r_h to replace one of them. When compared to r_{j_k}, firstly the dispatch cost of r_h at the moment right before dispatching r_{j_k} is calculated, which is denoted by h_cost (Line 9). Then, the minimum bid for r_h to replace r_{j_k} would yield that

$$bid_h - h\_cost = bid_{j_k} - cost_{j_k} .$$

Therefore, the minimum bid for r_h to replace r_{jk} is $bid_{j_k} - cost_{j_k} + h\_cost$ (Line 10).

C. Theoretical Analysis

In this section, we analyze the approximation factor of Greedy, and the auction properties achieved by Greedy & GPri with Theorem III.1 and III.2 respectively.

Approximation factor of Greedy. To analyze the approximation effectiveness of Algorithm 1, we define an auxiliary variable as follows:

$$\beta = \max_{v_i, r_j} \frac{bid_j}{u^0_{i,j}} , \tag{4}$$

where $u^0_{i,j}$'s are the initial utilities before any dispatch, i.e., $u_{i,j}$'s in Lines 2 $\sim$ 6 of Algorithm 1. Note that this variable only depends on the inputs.

Theorem III.1. *If $\beta > 0$, Algorithm 1 achieves an approximation factor of $\frac{1}{n+n\beta(\bar{c}-1)}$.*

Proof sketch. Let u^0_{max} denote the maximum initial utility, i.e., $\max_{i,j} u^0_{i,j}$, and U^* & U_G denote the utilities of the optimal solution and the greedy solution respectively. Obviously we have $u^0_{max} \leq U_G$ and $\forall r_j, bid_j \leq \beta u^0_{max}$. Then, for each vehicle in the optimal solution, we can bound its utility with $u^0_{max} + \beta(\bar{c}-1)u^0_{max}$. Because there are n vehicles,

$$U^* \leq nu^0_{max} + n\beta(\bar{c}-1)u^0_{max} \leq (n + n\beta(\bar{c}-1))u_G .$$

Please refer to our technical report for the thorough proof. □

Theorem III.2. *The auction mechanism implemented by Algorithm 1 and 2 is computationally efficient, truthful and individually rational.*

Computational efficiency. We denote the number of requesters and vehicles as m and n respectively. For Algorithm 1, the time cost of the initialization phase (Lines 2 $\sim$ 6) is $\mathcal{O}(mn\bar{c}^2q)$. For the dispatch phase, in a single loop, the time costs are mainly charged by Line 8 ($\mathcal{O}(mn)$) and Lines 12 $\sim$ 15 ($\mathcal{O}(m\bar{c}^2q)$) respectively. Because there are at most m loops, the time cost of the dispatch phase is $\mathcal{O}\left(m^2(n+\bar{c}^2q)\right)$. Overall, the time complexity of order dispatch (Algorithm 1) is $\mathcal{O}\left(mn\bar{c}^2q + m^2(n+\bar{c}^2q)\right)$. The time complexity of Algorithm 2 is also $\mathcal{O}\left(mn\bar{c}^2q + m^2(n+\bar{c}^2q)\right)$, which results from invoking Algorithm 1 at Line 2. The complexities of the order dispatch and pricing algorithms are polynomial w.r.t. the input size.

Truthfulness. We prove the truthfulness of the mechanism by showing that it satisfies the properties of *monotonicity* and *critical payment*. According to [22], an auction mechanism is truthful if it satisfies conditions of *monotonicity* and *critical payment*.

- *Monotonicity*. It states that if bidder r_j wins the auction by bidding bid_j, then he/she also wins by bidding

$bid'_j \geq bid_j$. This is obviously true in Algorithm 1. Given the sequence of dispatched requesters $\{r_{j_1} ... r_{j_k} ... r_{j_z}\}$, there are two possible cases for r_{j_k} to bid $bid'_{j_k} \geq bid_{j_k}$. In the first case, r_{j_k} is dispatched in an earlier step, replacing r_{j_h} where $h < k$. In the second case, requesters dispatched before the k-th step are not replaced by r_{j_k}. As a result, r_{j_k} is still dispatched at the k-th step.

- *Critical payment*. It states that each bidder pays the critical value. That is, requester r_h would not be dispatched if he/she bids $bid'_h < pay_h$. In algorithm 2, pay_h is set to the minimum among the bid values of r_h to replace a dispatched requester in $\widehat{R}'$. Therefore, pay_h is the critical bid value of r_h.

Individual rationality. Algorithm 2 (Lines 1 and 10) ensures that $pay_h \leq bid_h$. As truthfulness of the auction mechanism is guaranteed, we have $val_h - pay_h = bid_h - pay_h \geq 0$. Therefore, the mechanism is individually rational.

IV. The Ranking Approach

Though the greedy winner selection principle is shown to be effective in existing works [16]–[18], it may not be good for the studied ridesharing problem because of the order dispatch complexity. Ridesharing orders affect one another; their aggregated utility to a vehicle is not simply the sum of their individual utilities. Greedy dispatches orders one by one, locally optimizing each order without considering others. Thus, it is less likely to obtain a globally good dispatch solution. In this section, we propose another algorithm which firstly packs the orders globally, and then dispatches the order packs w.r.t. their utility rankings. Compared with Greedy, it proceeds with the order packs, and thus has a long-sighted view in terms of orders' combination.

We introduce the ranking-based order dispatch and pricing algorithm in Section IV-A. Then, we perform corresponding theoretical analysis in Section IV-B.

A. Ranking-based Order Dispatch and Pricing

Algorithm 3 Ranking-based order dispatch (Rank)

```
Input: Requester set $R$, vehicle set $V$.
Output: Updated travel plans of vehicles.
 1: I. Pack generation.
 2: for all $r_j \in R$ do
 3:     $v_{r_j} \leftarrow$ the nearest vehicle of $r_j$ in $V$.
 4: end for
 5: for all $r_j \in R$ do
 6:     $pack_j \leftarrow \underset{R' \subset R, r_j \in R', \|R'\| \leq \bar{c}}{\arg\max} U(dispatch\ R')$.
 7: end for
 8: II. Pack dispatch.
 9: $Rank \leftarrow$ Rank $pack_j$'s according to their utilities.
10: while $Rank \neq \phi$ do
11:     $pack_{j*} \leftarrow$ the first pack in $Rank$.
12:     Dispatch $pack_{j*}$ to its vehicle $v_{pack_{j*}}$ .
13:     Remove packs in conflict with $pack_{j*}$ from $Rank$.
14: end while
```

1) Order Dispatch: As shown in Algorithm 3, the dispatch algorithm contains two phases. Phase *I* in Algorithm 3 is to generate packs, and Phase *II* is to dispatch the packs.

***I.* Pack generation** (Lines $2 \sim 7$). Firstly, for each r_j, we find its nearest vehicle, v_{r_j}, in V (Lines $2 \sim 4$). Then, for each r_j, we find the optimal pack which contains r_j and has at most $\bar{c}$ requesters, such that dispatching this pack to one of their nearest vehicles yields the maximum utility (Lines $5 \sim 7$). $U(\cdot)$ in Line 6 refers to the maximum utility of dispatching requesters to one of their vehicles. Note that the dispatch of a pack of requesters is conducted w.r.t. their optimal sequence.

The time cost of Lines $2 \sim 4$ is $\mathcal{O}(mnq)$. For Lines $5 \sim 7$, there are totally $m^{\bar{c}}$ packs explored. To calculate the utility of a pack, it explores $\bar{c}!$ orderings of the requesters, where each ordering costs $\mathcal{O}(\bar{c}^3 q)$ to compute the travel cost. Therefore, the time cost of Lines $5 \sim 7$ is $\mathcal{O}(\bar{c}! m^{\bar{c}} \bar{c}^3 q)$. Because generally $\bar{c}! m^{\bar{c}} \bar{c}^3 q > mnq$, the time complexity of pack generation is $\mathcal{O}(\bar{c}! m^{\bar{c}} \bar{c}^3 q)$.

***II.* Pack dispatch** (Lines $9 \sim 14$). The packs are ranked in descending order of their utilities at Line 6. Then, they are dispatched in the order of their rankings. The algorithm firstly retrieves the pack with the highest ranking, $pack_{j*}$, among the remaining packs. Then, it dispatches $pack_{j*}$ to its vehicle $v_{pack_{j*}}$. Finally, it removes the remaining packs whose requesters or vehicles are occupied by $pack_{j*}$. That is,

$$\forall pack_j \in Rank\,, \text{if } pack_{j*} \cap pack_j \neq \phi \text{ or } v_{pack_j} = v_{pack_{j*}}\,,$$
$$\text{remove } pack_j \text{ from } Rank\,.$$

The pack dispatch process terminates when $Rank$ becomes empty.

There are at most m packs to be dispatched, and for each dispatch, at most m packs can be removed. Therefore, the time cost of pack dispatch is $\mathcal{O}(m^2)$.

Time complexity. The time complexity of pack generation and dispatch are $\mathcal{O}(\bar{c}! m^{\bar{c}} \bar{c}^3 q)$ and $\mathcal{O}(m^2)$ respectively. Because $\bar{c} \geq 2$ in ridesharing, the overall time complexity of ranking-based order dispatch is $\mathcal{O}(\bar{c}! m^{\bar{c}} \bar{c}^3 q)$.

2) Order Pricing: Given a dispatched requester, $r_h \in \widehat{R}$, we denote the set of packs containing r_h as S_h, i.e.,

$$S_h = \{pack_j | r_h \in pack_j, j = 1...m\},$$

where $pack_j$'s refer to the packs found at Lines $5 \sim 7$ in Algorithm 3. To ensure truthfulness, we price r_h with its critical payment, i.e., the smallest pay_h ($pay_h \leq bid_h$) such that when $bid_h = pay_h$, one of packs in S_h would still be dispatched. For ease of presentation, let bid^0_h denote the original bid of r_h in the input. bid_h would be used as a variable in our analysis. The difficulty of pricing orders dispatched by Rank lies in the following two aspects.

I. Unlike Greedy where each order acts as a single unit to be dispatched, in ranking-based dispatch, the dispatch unit is a pack. Each requester can be included in multiple packs. The utility of a pack equals

$$\sum_{r_l \in pack} bid_l - cost_{pack}\,, \tag{5}$$

where $cost_{pack}$ denotes the cost of delivering requesters in the pack. For $pack_j \in S_h$, its utility is linearly dependent on bid_h. We denote the smallest bid of r_h for $pack_j$ to be dispatched as r_h's critical bid for $pack_j$. Then, pay_h should be the minimum among r_h's critical bids for packs in S_h. That is,

$$pay_h = \min_{pack_j \in S_h} criticalBid_h(pack_j)\,. \qquad (6)$$

II. Algorithm 3 proceeds w.r.t. the ranking (i.e., $Rank$), however, unlike Greedy where the dispatch units (i..e, requesters) are fixed, the packs in $Rank$ can change regarding different bid_h's. When bid_h decreases from bid_h^0, for $pack_j \in S_h$, it may not be the optimal pack of r_j anymore due to its decrease in utility. For example, let the pack of r_1, $pack_1$, be one of the packs in S_h, and $pack_1 = P_a = \{r_1, r_2, r_h\}$. There is another pack $P_b = \{r_1, r_4, r_6\}$, and $r_h \neq r_6$. Let $\Delta U_{ab} = U(P_a) - U(P_b)$. If bid_h is decreased from bid_h^0 by more than ΔU_{ab}, P_b would be the optimal pack of r_1, $pack_1$, in replace of P_a. This uncertainty of $Rank$ makes it hard to analyze r_h's critical bids for packs in S_h, because packs in $Rank$ may change over the change of bid_h.

Before we introduce our pricing approach, we first describe Lemma IV.1. Given $pack_j \in S_h$, we let p_j^0 denote its original instance which includes r_h, when $bid_h = bid_h^0$. Besides, let p_j' denote the pack with the largest utility among packs including r_j and excluding r_h. Let $f(pack_j)$ denote the smallest bid of r_h for p_j^0 to remain the optimal pack of r_j. Lemma IV.1 indicates that depending on the value of bid_h, there are only two possible instances of $pack_j$, i.e., p_j^0 and p_j'.

Lemma IV.1. *When $bid_h \geq f(pack_j)$, $pack_j = p_j^0$, otherwise $pack_j = p_j'$.*

Proof. Let p'' be any pack which satisfies that $r_h \in p''$ and $r_j \in p''$. From Equation (5), it can be found that a change of bid_h, Δbid_h, incurs exactly the utility change of Δbid_h for all packs including r_h. In other words, the value of bid_h does not affect the ranking among p_j^0 and p''. Therefore, when bid_h is small enough such that p_j^0 is not optimal for r_j anymore, the pack to replace p_j^0 must be the one which has the largest utility among packs excluding r_h, i.e., p_j'. □

In the following, we propose the Divide-and-Walk (DnW) approach to find pay_h. The idea is to firstly divide the domain of bid_h into intervals, according to the pre-computed $f(pack_j)$'s of packs in S_h. Then, it searches pay_h interval-by-interval. For ease of presentation, we denote $f(pack_j)$'s as b_{j_k}'s in ascending order of their values. That is, b_{j_k} denotes the k-th smallest $f(pack_j)$, with the corresponding pack represented as $pack_{j_k}$. As shown in Figure 2, each b_{j_k} on the axis indicates the instance change of $pack_{j_k}$. Subsequently each interval of bid_h, $[b_{j_k}, b_{j_{k+1}})$, indicates a particular instance set of packs in S_h. In other words, each interval yields a particular instance of $Rank$. The DnW approach finds pay_h by exploring these intervals in ascending order.

As shown in Algorithm 4, the inputs are $Rank$, which is the ranking of packs obtained at Line 9 in Algorithm 3, and bid_h^0 which is the original bid of r_h. DnW first finds S_h, and sorts the packs w.r.t. their b_j's (Lines 1 ∼ 3). Then it starts searching the critical bid of r_h from $[b_{j_1}, b_{j_2})$ to $[b_{j_K}, +\infty)$. In the for loop (Lines 5 ∼ 16), the variable k indicates the interval $[b_{j_k}, b_{j_{k+1}})$. Subsequently, for $pack_{j_a}$ where $a \leq k$, their instances should be $p_{j_a}^0$, i.e., the original instances that contain r_h (Line 6). Otherwise, their instances should be p_{j_a}'. By sorting these updated $pack_{j_a}$'s together with packs in $Rank \setminus S_h$, we obtain a new ranking, $Rank$' (Line 7). In $Rank$', r_h would be dispatched if one of packs in $\{pack_{j_a} | a \leq k, pack_{j_a} \in S_h\}$ is dispatched. For each of them, we find r_h's smallest bid for it to be successfully dispatched by Algorithm 3 (Lines 8 ∼ 9). This can be done in a similar manner as Algorithm 2. The found bid value, bid_h^a, would be set to b_{j_k} if it falls behind $[b_{j_k}, b_{j_{k+1}})$ (Line 10). Finally, if the final value of bid_h^a falls inside the interval $[b_{j_k}, b_{j_{k+1}})$ and is smaller than the current minimum bid pay_h, pay_h would be set to bid_h^a (Lines 11 ∼ 13). After it finishes the exploration of one interval, the for loop can be terminated if pay_h has been updated (Line 15). This is because the bids found in later intervals must be larger than the current one. Note that the intervals are explored in strictly ascending order.

Algorithm 4 The Divide-and-Walk approach (DnW)

```
Input: $Rank$, $r_h$, $bid_h^0$.
Output: Price for $r_h$, $pay_h$.
 1: $S_h \leftarrow \{pack_j | pack_j \in Rank, r_h \in pack_j\}$.
 2: $b_j \leftarrow f(pack_j)$ for $pack_j \in S_h$.
 3: $b_{j_1} ... b_{j_K} \leftarrow$ sort $b_j$'s in ascending order.
 4: $pay_h \leftarrow bid_h^0$.
 5: for all $k = 1...K$ do
 6:     $pack_{j_a} \leftarrow p_{j_a}^0$ if $a \leq k$ else $p_{j_a}'$ for $pack_{j_a}$ in $S_h$.
 7:     $Rank' \leftarrow$ the ranking w.r.t. the updated $pack_{j_a}$'s.
 8:     for all $a = 1...k$ do
 9:         $bid_h^a \leftarrow$ smallest bid to dispatch $pack_{j_a}$ by Alg. 3.
10:         $bid_h^a \leftarrow b_{j_k}$ if $bid_h^a < b_{j_k}$.
11:         if $bid_h^a < b_{j_{k+1}}$ then
12:             $pay_h \leftarrow \min\{pay_h, bid_h^a\}$.
13:         end if
14:     end for
15:     If $pay_h \neq bid_h^0$ :   Break.
16: end for
17: return  $pay_h$.
```

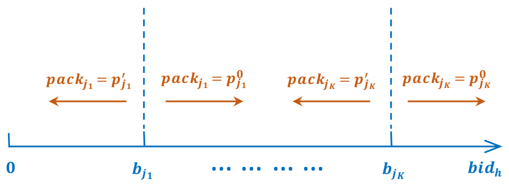


Fig. 2. The Divide-and-Walk (DnW) approach.

Lemma IV.2 (Correctness of Algorithm 4)**.** *Algorithm 4 finds exactly the critical payment of r_h.*

It is not hard to understand the lemma as Algorithm 4 searches the critical payment in the ascending order of intervals. Please refer to our technical report for the formal proof.

Time complexity. The time cost of the initialization at Lines $1 \sim 4$ is $\mathcal{O}(m \log m)$, mainly charged by the sorting. In one loop of the *for* clause, constructing $Rank$' at Lines $6 \sim 7$ takes $\mathcal{O}(m \log m)$ time. Besides, the sub-loop to find the critical bids of packs at Lines $8 \sim 14$ takes $\mathcal{O}(m^2)$ time. Because there are at most m loops of the *for* clause, its time cost is up to $\mathcal{O}(m^2(\log m + m)) \rightarrow \mathcal{O}(m^3)$. Overall, the time complexity of Algorithm 4 is $\mathcal{O}(m^3)$.

B. Theoretical Analysis

In this section, we analyze the approximation factor of Rank, and the auction properties achieved by Rank & DnW with Theorem IV.1 and IV.2 respectively.

Theorem IV.1. *Alg. 3 achieves an approximation factor of* $\frac{1}{m}$.

Proof sketch. The utility sum of packs in $Rank$ is the bound of the utility of any dispatch solution. The utility of the solution obtained by Algorithm 3 is at least $\frac{1}{m}$ of the utility sum of $Rank$, which subsequently indicates it is at least $\frac{1}{m}$ of the optimum. Please refer to our technical report for the formal proof. □

Theorem IV.2. *The auction mechanism implemented by Algorithm 3 and 4 is computationally efficient, truthful and individually rational.*

Computational efficiency. As analyzed in Section IV-A, the time complexities of Algorithm 3 and 4 are $\mathcal{O}(\bar{c}!m^{\bar{c}}\bar{c}^3 q)$ and $\mathcal{O}(m^3)$ respectively. Obviously $\mathcal{O}(m^3)$ is polynomial to input size. For $\mathcal{O}(\bar{c}!m^{\bar{c}}\bar{c}^3 q)$, it can be rather large when $\bar{c}$ is unbounded. However, in practical taxi-sharing platforms like Didi Chuxing, the maximum number of requesters in a vehicle is no larger than 3. Therefore, considering the practical scenario where $\bar{c}$ is often bounded by a small value like 3, we have $\mathcal{O}(\bar{c}!m^{\bar{c}}\bar{c}^3 q) \rightarrow \mathcal{O}(m^3 q)$, which is polynomial to m. Therefore, the implemented auction is computationally efficient in the practical taxi-sharing scenario.

Truthfulness. As mentioned before, an auction mechanism is truthful if its pricing strategy satisfies *monotonicity* and *critical payment*. As shown in Lemma IV.2, Algorithm 4 prices a requester with exactly its critical payment. In the following, we show that Algorithm 4 is monotonic. Let pay_h be the price of r_h determined by Algorithm 4.

Lemma IV.3. *If* $bid_h \geq pay_h$, r_h *will be dispatched by Alg. 3.*

Proof sketch. A bid higher than pay_h can only make r_h dispatched in an earlier step. Please refer to our technical report for the formal proof. □

Individual rationality. From Lines 4 and 12 of Algorithm 4, we know that $pay_h \leq bid_h^0$. Because truthfulness is guaranteed, $bid_h^0 = val_h$. Then,

$$\forall r_h \in \widehat{R}, \quad u_h = val_h - pay_h = bid_h^0 - pay_h \geq 0\,.$$

Thus, the auction mechanism is individually rational.

V. Experiments

In this section, we run experiments to evaluate the proposed methods. We first describe our experimental settings in Section V-A. Then, we present the experimental results of dispatch methods in Section V-B, and the results of pricing methods in Section V-C. Besides, we present the results of increasing requesters' bids in Section V-D, and the scalability results in Section V-E.

A. Experimental Setup

Our experiments are run on the historical order and vehicle data of the ridesharing business in Didi Chuxing, in the urban area of Beijing (the area within the 5th Ring Road whose size is $29.7 \times 29.5\ km^2$) and during the time period of $7:00 \sim 7:30\ am$ of a normal weekday. We target the orders whose starting and ending coordinates both fall in the area, and the vehicles whose online locations fall in the area. $7:00 \sim 7:30\ am$ is one of the busiest periods in a day. The number of orders is around 5000, and the number of vehicles online is around 7000. Our algorithms conduct order dispatch and pricing round by round. The time window of one round would be varied in experiments.

Simulation setup. Every order is issued at the time as recorded in the data, with specified origin and destination. Besides, in the data, each order is associated with a price. This is the upfront price determined by Didi Chuxing with respect to the origin & destination, demand-supply condition, and etc. In our experiment, this price is treated as requester's valuation of the service, i.e., val_j. Because all the proposed methods guarantee truthfulness, we have $bid_j = val_j$. During the simulation, if an order is not dispatched in a round, it would be pended to the next round. An order would be omitted if it has been pended for more than 5 minutes. In practice, such orders can be asked for higher bids to get priority. A vehicle is available during the time when it is recorded as online in the historical data, and becomes unavailable when it turns to offline. A newly online vehicle would be created at the location as recorded in the data. After that if it is not dispatched with orders, it would randomly walk over the road network, otherwise it would move w.r.t. its travel plan.

This work targets taxi-sharing services where the maximum number of orders sharing a vehicle would not be too large in practice. In Didi Chuxing, this number is three, which is also the setting in our experiments.

The travel cost from dispatching orders to a vehicle is computed w.r.t. the origin & destination of the orders and the real-time location of the vehicle, using the route planning method in Section III-A. The road network is fetched from OpenStreetMap[4], which is further preprocessed as in [21].

Control variables. As mentioned before, the algorithms operate round by round. We denote the time duration of one round as t_{rnd}, and vary it with $\{5, 10, 15, 20\}$ in seconds similar to [21]. For α_d, in taxi services of China, the charge of traveling per kilometer normally falls between 2.0 and 3.0 $yuan$ ($yuan$

[4]https://www.openstreetmap.org

TABLE II
EXPERIMENTAL SETTINGS

Variables	Values
t_{rnd}	5 s, 10 s, **15 s**, 20 s
γ	1.2, 1.5, **1.8**, 2.0
α_d	2.5, 3.0, **3.5**, 4.0
charge ratio	**0**, 0.1, 0.2, 0.3, 0.4

is the basic unit of the official currency of China). This charge can be a bit higher when it comes to peak hours. Therefore, we vary α_d with values in $\{2.5, 3.0, 3.5, 4.0\}$. Finally, in Definition 2, each requester r_j specifies θ_j, which is his/her maximum allowed amount of wasted time caused by waiting and detour. In our experiment, θ_j is specified as follows:

$$\theta_j = (\gamma - 1) * t(s_j, e_j)\,,$$

where $t(s_j, e_j)$ denotes the shortest travel time between its origin and destination. γ can be interpreted as the maximum allowed ratio between a requester's actual time cost and the shortest possible time cost. γ can be equivalently interpreted as the detour ratio in [21]. Similar to [21], γ is varied with values in $\{1.2, 1.5, 1.8, 2.0\}$. The variable settings are summarized in Table II, with the default values in bold.

Compared methods. We evaluate the dispatch and pricing methods of Greedy and Rank in Section V-B and V-C respectively.

For comparison with existing works [16]–[18], though their studied problems and settings are different from ours, the greedy-based winner selection principle is universal. Observing that the greedy-based method is shown to be effective in their works, we adapt the greedy winner selection rule to our problem (Alg. 1), develop the corresponding pricing method (Alg. 2), and perform theoretical analysis (Section III-C). The proposed Greedy algorithm in this paper is also the state-of-art method in existing works.

The comparison to the optimal solution on a set of small-scale data and the comparison to Greedy and Rank under the non-auction setting can be found in our technical report[5].

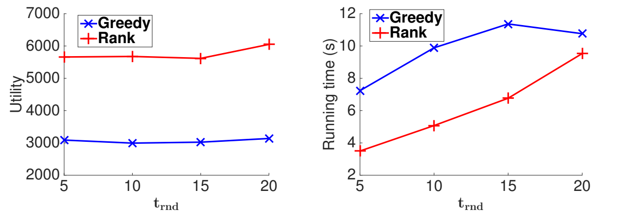
Evaluation metric. We evaluate the dispatch methods in terms of their achieved utilities (defined in Eq. (2)) and running times. Remember that the platform server dispatches requesters in a round manner, so the running time of a dispatch algorithm should not exceed the time duration of one round. Subsequently, the duration of a round would be an important reference to judge the efficiency of the dispatch methods.

For the pricing methods, i.e., GPri for Greedy and DnW for Rank, we have proved that they guarantee individual rationality (non-negative u_{r_j} and U_R). In contrast, for both methods, we fail to prove the platform profitability (Definition 13), i.e., non-negative U_{plf}. Therefore, in the experiments, we would investigate the obtained U_{plf} of the pricing methods.

B. Results of Dispatch Methods

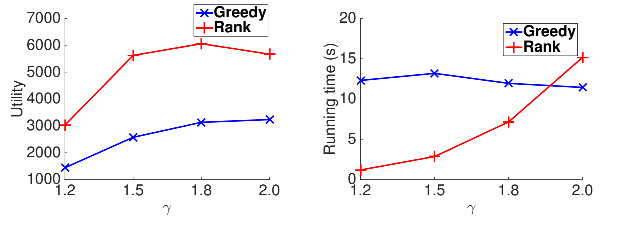
In this section, we compare the order dispatch methods in terms of their achieved overall utilities and running times.

[5]http://www.cse.ust.hk/%7Elzhengab/auctionRide.pdf



(a) Utilities of various t_{rnd}'s (b) Running times of various t_{rnd}'s

Fig. 3. Effect of t_{rnd}.



(a) Utilities of various γ's (b) Running times of various γ's

Fig. 4. Effect of γ.

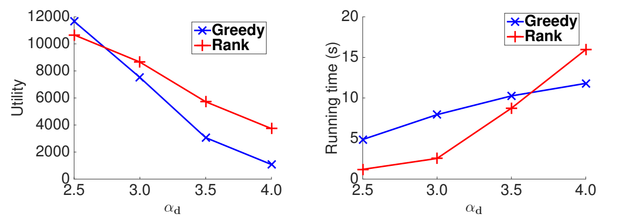
Effect of $\mathbf{t_{rnd}}$. The experimental results of varying t_{rnd} are shown in Figure 3(a) and 3(b). It can be observed that the overall utility of Rank is much higher than that of Greedy, almost doubling it. The superiority of Rank over Greedy comes from its enumeration of order packs. Let us suppose that there are two requesters r_1 & r_2 and a vehicle v_1, and dispatching only one of them brings a negative amount of utility. However, r_1 and r_2 have a long shared travel path, and as a result, dispatching them together to v_1 can bring positive utility. In this case, Greedy fails to dispatch any requester. In contrast, Rank can successfully pack r_1 and r_2 in its packing stage, and then dispatch them to v_1.

It can be found in Figure 3(b) that the running time of Rank is smaller than that of Greedy, but grows faster than it over increasing t_{rnd}. Remember that the time complexities of Greedy and Rank are $\mathcal{O}(mn\bar{c}^2q)$ and $\mathcal{O}(\bar{c}!m^{\bar{c}}\bar{c}^3q)$ respectively. The time complexity of Rank is larger than that of Greedy, which explains its faster growth in running times. Nevertheless, under different values of t_{rnd}, the running time of Rank keeps smaller than it. In contrast, the running time of Greedy can be larger than t_{rnd} when $t_{rnd} = 5\ s$.

Effect of γ. The experimental results of varying γ are shown in Figure 4(a) and Figure 4(b). The achieved utilities of Greedy and Rank both generally go up over increasing γ. This is reasonable because a large γ indicates a large amount of allowed wasted time of requesters. The utility gap between Rank and Greedy stays large over different values of γ, again demonstrating Rank's superiority.

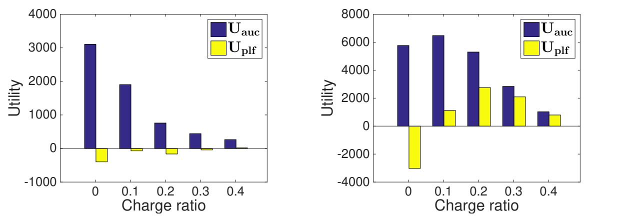
The running time of Greedy stays stably around 12 s over various γ's. In contrast, Rank is more costly given a large γ. However, both methods can finish within t_{rnd}.

Effect of $\mathbf{\alpha_d}$. The experimental results of varying α_d are shown in Figure 5(a) and 5(b). In terms of achieved utility, Rank is superior to Greedy except when $\alpha_d = 2.5$. However,



(a) Utilities of various α_d's (b) Running times of various α_d's

Fig. 5. Effect of α_d



(a) GPri's utilities of various charge ratios (b) DnW's utilities of various charge ratios

Fig. 6. Effect of charge ratio

their gap at $\alpha_d = 2.5$ is not large. Rank is more robust and persistently achieves a competitive amount of utility over various α_d's.

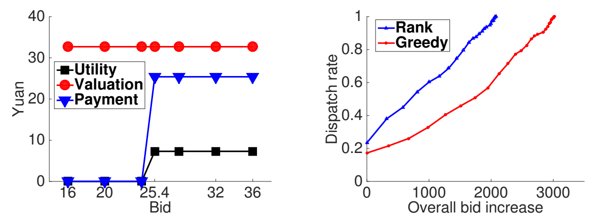
In Figure 5(b), it can be found that the running times of both methods go up over increasing α_d. This is because a large α_d results in few dispatched orders. As a result, the number of pended orders in a round becomes large. The running time of Rank grows more quickly than that of Greedy over increasing α_d, but remains acceptable.

C. Results of Pricing Methods

In this section, we compare the pricing methods of Greedy and Rank, i.e., GPri and DnW. They are invoked to price orders for Greedy and Rank respectively. Using purely these pricing algorithms, our experiments show that the achieved platform utilities can be negative. In pursuit of positive platform utility, we additionally set up a dispatch fee for the platform. That is, for r_j with bid_j, a ratio of bid_j would be compulsorily taken away by the platform, which can be regarded as its charge of being the dispatch agent. We denote this charge ratio as CR. Before the platform invokes the dispatch & pricing algorithms, it firstly deducts requesters' bids with the dispatch fee. The new bids of requesters would be

$$bid'_j = bid_j - bid_j * CR .$$

These deducted bids would be the input to the algorithms. Note that only dispatched requesters would be charged. The dispatch fee $bid_j * CR$ would be returned back to requesters if they are not dispatched. Besides, applying this dispatch fee would not ruin the mechanism's truthfulness, individual rationality and computational efficiency. For both algorithms, the platform profitability can be assured by setting $CR \geq 0.5$. This is because the dispatch cost of r_j cannot exceed $bid'_j = 0.5{*}bid_j$.



(a) Requester utility over various bids (b) Dispatch rate over bid increase

Fig. 7. Results of increasing bids

Then even $pay_j = 0$, the pre-charged $0.5 * bid_j$ can cover the cost.

The achieved utilities of varying CR are shown in Figure 6(a) and 6(b), where U_{auc} and U_{plf} represent the overall utility achieved by Greedy or Rank, and the platform utility determined by GPri or DnW respectively.

For Greedy and its pricing algorithm GPri, in Figure 6(a), the platform utility obtained by GPri is negative in most of cases ($CR \leq 0.3$). Besides, in the only case when the platform utility is positive ($CR = 0.4$), both the platform utility and the overall utility are small.

For Rank and DnW, in Figure 6(b), the platform utility is negative only when $CR = 0$. Generally the platform utility is positive over positive CR's. Furthermore, when $CR = 0.2$, the platform utility is nearly half of the overall utility, which is also comparatively large. We can assure positive platform utility with $CR = 0.5$, however, it is suggested that setting $CR = 0.2$ turns out the best utility results.

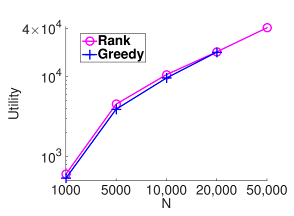
The running time results of GPri and DnW can be found in our technical report. They keep smaller than 0.25 s, and are omitted in this paper. Pricing individual requesters is independent of one another, and in our implementation, we use multiple threads where each one prices one requester. With this speed-up, the pricing process is quite fast.

D. Results of Varying Bids

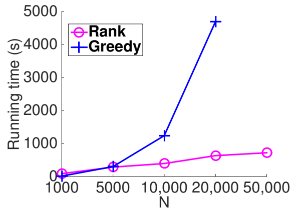
In this section, we increase the bids of requesters to investigate its effects on the requester utilities and the order dispatch rate.

In an auction, a bidder should be able to increase his/her success rate by increasing the bid. We randomly select a requester and investigate his/her dispatch results over different bids, which are presented in Figure 7(a). The critical payment and the service valuation of the requester are 25.4 and 32.7 $yuan$ respectively (the unit $yuan$ would be omitted in later discussion for brevity). We can see that when the requester makes a bid no less than 25.4, he/she would be dispatched with a payment equaling 25.4, and his/her achieved utility is equal to $32.7 - 25.4 = 7.3$. In contrast, when his/her bid is less than 25.4, the requester would not be dispatched ($payment = 0$), and the resultant utility is 0.

To investigate how the increase of bids affects the dispatch rate of orders, we select the orders and vehicles from a 5-minute period and run our dispatch algorithms upon them. For



(a) Utilities of various input sizes



(b) Running times of various input sizes

Fig. 8. Results of scalability

the un-dispatched orders, we increase their bids by 1 and re-run our dispatch algorithms upon them. We repeat this process until all the orders get dispatched. The results are shown in Figure 7(b). It can be found that Rank achieves 100% dispatch rate with the overall bid increase around 2000, which is much less than that of Greedy (around 3000). Besides, under the same amount of bid increase, Rank persistently achieves a dispatch rate much larger than that of Greedy.

E. Results of Scalability

In this section, we synthetically generate various numbers of orders and vehicles to test the scalability of dispatch methods. We denote the number of orders/vehicles as N, and ranges it with values $\{1000,\ 5000,\ 10,000,\ 20,000,\ 50,000\}$. For Rank, we conduct some optimization to save its time cost when $N \geq 5000$. Specifically, before its pack generation phase (see Algorithm 3), we properly cluster the orders w.r.t. their origins/destinations to form $\frac{N}{1000}$ order groups. Then, in the pack generation phase, each order searches the optimal pack within its group, whose size is expected to be around 1000. The shrink of the search space from N to around 1000 greatly reduces the computation cost. Besides, different order groups are processed by different servers parallelly. The packs generated from different groups are finally combined together and processed by the pack dispatch phase. The experimental results are presented in Figure 8. The results of Greedy at $N = 50,000$ are not reported because its running time is unbearable.

Rank persistently achieves a larger amount of utility than Greedy when $N < 20,000$. Note that the utility difference is not apparent in the figure because the Y-axis is in log-scale. In fact, when $N = 5000$, the utilities of Greedy and Rank are 3899 and 4504 respectively, whose ratio is 0.866. Their difference is not significant after $N = 20,000$ because the orders are so dense that it is easy to form good combinations for them.

For the running times, when $N = 5000$, both methods cost hundreds of seconds. However, the running time of Greedy is more than 1200 s at $N = 10,000$, and approaches 5000 s when N reaches $20,000$. In contrast, even when N reaches $50,000$, the running time of Rank remains hundreds of seconds, which benefits from using different servers to process the divided order groups in its pack generation phase.

Summary of experimental results. In summary, in terms of the achieved overall utility, Rank is more effective than Greedy. Besides, Rank & DnW can not only realize the desired auction properties, but also bring a proper amount of platform utility. Its advantage over Greedy comes from packing the orders in advance, which offers a deeper insight to its dispatch process. In terms of running times, both methods are efficient enough regarding the real data. The scalability test suggests that Rank is also able to process large-scale inputs, by clustering the orders in the pack generation phase.

VI. Related Work

The ridesharing problem. The dial-a-ride problem (DARP) [3]–[6] can be seen as the offline version of the ridesharing problem. Given a set of vehicles and riders with origin and destination, DARP is to devise a routing solution such that the overall travel distance of vehicles is minimized while delivering all the riders.

The optimization goal is comparatively diverse in works of ridesharing. Cheng et. al. [20] consider the social affinity among riders. To offer a better riding experience, they manage to maximize the affinity among riders sharing a vehicle. Alonsomora et. al. [8] adopt a score function to linearly combine the travel time delays and the number of un-served orders. They construct a RTV graph which contains orders, order packs and vehicles, based on which they solve the problem via integer programming. Wang et. al. [19] assume that the requesters make travel to take specific activities, based on which they enable multiple destinations of a requester. They study how to group the requesters under such a relaxed setting. Na et. al. [7] formulate the ridesharing problem as a maximum weighted bigraph matching problem, where each vehicle is allowed to take at most one requester. Each vehicle has its original travel plan, and the goal is to maximize the shared travel distance. Qian et. al. [29] study the order grouping problem with the setting that the grouped requesters would walk to a meeting point so that the driver can save some travel cost during pick-up.

Zheng et. al. [21] take the order prices into concern, and aim to maximize the platform profit. The problem setting in [21] is similar to the order dispatch problem in this work. The greedy algorithm is shown to be competitive in [21], which has been developed as a solution for the auction mechanism, accompanied by a pricing method (GPri). In contrast, other proposed algorithms in [21] are not brought for comparison, because so far we are still not able to devise proper pricing algorithms for them to meet the desired auction properties.

Yang et. al. [11] and Ma et. al. [10], [14] study the ridesharing problem in an online setting. In this setting, the platform server responds immediately to each order's arrival, and conducts dispatch for it whenever possible. For each order, it finds the vehicle which owns the least amount of additional travel distance. Chen et. al. [30] dispatch orders in an online setting as well. It differs from [10], [11], [14] in that it finds multiple candidate vehicles for each order.

Foti et. al. [31] and Wolfson et. al. [32] study the fairness issue among requesters in ridesharing. They aim to obtain a grouping solution such that no requesters prefer one another to their current groups.

The auction problem. Among existing auction works, [16]–[18] are most close to us. Jin et. al. [18] propose an auction mechanism for mobile crowd sensing, where both task requesters and workers make bids, and the platform determines the winning requesters/workers and their prices/payments. Wang et. al. [16] enable a worker (mobile device) to bid multiple tasks, where different tasks are offered different bids. Yang et. al. [17] propose an auction model for mobile phone sensing, which differs from [16] in that each worker sets an overall bid for a selected set of tasks. The platform selects workers so that the overall utility is maximized. In aforementioned works, the greedy-based winner selection principle is shown to be effective. Thus, regarding this principle, we propose the Greedy dispatch method, and devise its pricing method (GPri). Besides, we propose another algorithm Rank & DnW, which is shown to be more effective in our experiments.

In [9], for every newly arrived order, each vehicle locally calculates the utility of serving this order, and bids it to the platform. The platform then selects the vehicle which brings the maximum amount of utility.

VII. Conclusion

In this paper, to enable self-motivated bonus offering of requesters, we study the problem of auction-based ridesharing. We devise an auction mechanism which enables requesters to offer bids and platform to dispatch and price orders. We propose two methods to implement the auction mechanism, i.e., the greedy and the ranking based algorithms. Through extensive experiments, the ranking-based algorithm is shown to be both effective and efficient.

VIII. Acknowledgments

The work is partially supported by the Hong Kong RGC GRF Project 16207617, the National Science Foundation of China (NSFC) under Grant No. 61729201, Science and Technology Planning Project of Guangdong Province, China, No. 2015B010110006, Hong Kong ITC ITF grants ITS/391/15FX and ITS/212/16FP, Didi-HKUST joint research lab project, Microsoft Research Asia Collaborative Research Grant and Wechat Research Grant. Data is retrieved from Didi Chuxing.

References

[1] T. News, "The number of ridesharing orders reaches 2.4 million in didi chuxing," 2018. [Online]. Available: http://tech.qq.com/a/20180402/017211.htm

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

[3] J.-F. Cordeau, "A branch-and-cut algorithm for the dial-a-ride problem," *Operations Research*, vol. 54, no. 3, pp. 573–586, 2006.

[4] J.-F. Cordeau and G. Laporte, "A tabu search heuristic for the static multi-vehicle dial-a-ride problem," *Transportation Research Part B: Methodological*, vol. 37, no. 6, pp. 579–594, 2003.

[5] R. W. Calvo and A. Colorni, "An effective and fast heuristic for the dial-a-ride problem," *4OR - A Quarterly Journal of Operations Research*, vol. 5, no. 1, pp. 61–73, 2007.

[6] L. M. Hvattum, A. Løkketangen, and G. Laporte, "A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems," *Networks*, vol. 49, no. 4, pp. 330–340, 2007.

[7] T. Na, G. Li, T. Zhao, J. Feng, H. Ma, and Z. Gong, "An efficient ride-sharing framework for maximizing shared route," *IEEE Transactions on Knowledge & Data Engineering*, vol. PP, no. 99, pp. 1–1, 2017.

[8] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proc. of the NAS*, p. 201611675, 2017.

[9] M. Asghari, D. Deng, C. Shahabi, U. Demiryurek, and Y. Li, "Price-aware real-time ride-sharing at scale: an auction-based approach," in *Proc. of the SIGSPATIAL GIS*, 2016, p. 3.

[10] S. Ma, Y. Zheng, and O. Wolfson, "T-share: A large-scale dynamic taxi ridesharing service," in *Proc. of the ICDE*, 2013, pp. 410–421.

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

[12] R. Geisberger, D. Luxen, S. Neubauer, P. Sanders, and L. Volker, "Fast detour computation for ride sharing," *arXiv preprint arXiv:0907.5269*, 2009.

[13] N. Agatz, A. L. Erera, M. W. Savelsbergh, and X. Wang, "Dynamic ride-sharing: A simulation study in metro atlanta," *Procedia-Social and Behavioral Sciences*, vol. 17, pp. 532–550, 2011.

[14] S. Ma, Y. Zheng, and O. Wolfson, "Real-time city-scale taxi ridesharing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 7, pp. 1782–1795, 2015.

[15] V. Krishna, *Auction theory*. Academic press, 2009.

[16] X. Wang, X. Chen, and W. Wu, "Towards truthful auction mechanisms for task assignment in mobile device clouds," in *Proc. of the INFOCOM*, 2017, pp. 1–9.

[17] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing," in *Proceedings of the MobiCom*, 2012, pp. 173–184.

[18] H. Jin, L. Su, and K. Nahrstedt, "Centurion: Incentivizing multi-requester mobile crowd sensing," in *Proc. of the INFOCOM*, 2017, pp. 1–9.

[19] Y. Wang, R. Kutadinata, and S. Winter, "Activity-based ridesharing: increasing flexibility by time geography," in *Proc. of the SIGSPATIAL GIS*, 2016, p. 1.

[20] P. Cheng, H. Xin, and L. Chen, "Utility-aware ridesharing on road networks," in *Proc. of the SIGMOD*, 2017, pp. 1197–1210.

[21] L. Zheng, L. Chen, and J. Ye, "Order dispatch in price-aware ridesharing," *Proceedings of the VLDB Endowment*, vol. 11, no. 8, 2018.

[22] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge University Press Cambridge, 2007, vol. 1.

[23] P. Lin, X. Feng, and Q. Zhang, *Auction Design for the Wireless Spectrum Market*, 2014.

[24] H. Shah-Mansouri and V. W. Wong, "Profit maximization in mobile crowdsourcing: A truthful auction mechanism," in *Proceedings of the ICC*, 2015, pp. 3216–3221.

[25] Y. Wen, J. Shi, Q. Zhang, X. Tian, Z. Huang, H. Yu, Y. Cheng, and X. Shen, "Quality-driven auction-based incentive mechanism for mobile crowd sensing," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 9, pp. 4203–4214, 2015.

[26] H. Jin, L. Su, D. Chen, K. Nahrstedt, and J. Xu, "Quality of information aware incentive mechanisms for mobile crowd sensing systems," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2015, pp. 167–176.

[27] S. Sahni, "Approximate algorithms for the 0/1 knapsack problem," *Journal of the ACM*, vol. 22, no. 1, pp. 115–124, 1975.

[28] M. E. Horn, "Fleet scheduling and dispatching for demand-responsive passenger services," *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 1, pp. 35–63, 2002.

[29] X. Qian, W. Zhang, S. V. Ukkusuri, and C. Yang, "Optimal assignment and incentive design in the taxi group ride problem," *Transportation Research Part B: Methodological*, vol. 103, pp. 208–226, 2017.

[30] L. Chen, Q. Zhong, X. Xiao, Y. Gao, P. Jin, and C. S. Jensen, "Price-and-time-aware dynamic ridesharing," in *Proc. of the ICDE*, 2018.

[31] L. Foti, J. Lin, O. Wolfson, and N. D. Rishe, "The nash equilibrium among taxi ridesharing partners," in *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2017, p. 72.

[32] O. Wolfson and J. Lin, "Fairness versus optimality in ridesharing," in *2017 18th IEEE International Conference on Mobile Data Management (MDM)*, 2017, pp. 118–123.