Optimizing Order Dispatch for Ride-sharing Systems

Yubin Duan*, Ning Wang† and Jie Wu*
*Department of Computer and Information Sciences, Temple University,Philadelphia, USA
† Department of Computer Science, Rowan University, Glassboro, USA
Email: yubin.duan@temple.edu, wangn@rowan.edu, jiewu@temple.edu

***Abstract*—Ride-sharing companies such as Didi and Uber have served billions of passenger requests from all over the world. The efficiency of the ride-sharing is highly depended on the order dispatch system which assigns passenger requests to idle drivers. However, designing such a dispatch system is challenging because of the spatial-temporal dynamic of passenger requests, and the trade-off between the benefits for passengers and drivers. Existing order dispatch systems use either a system-assigning approach or a driver-grabbing approach. However, either approach has its own flaws. In this paper, we propose to combine the two existing approaches and jointly considers both passengers' and drivers' interest. In our approach, a passenger request is broadcast to the drivers in a dispatch region chosen by the system. The size of the dispatch region could iteratively increase until the request is accepted. We formulate an optimization problem to determine the increase speed of the dispatch region. Drivers' idle driving distances and passengers' waiting time are jointly considered. We propose a dynamic programming algorithm to optimally solve the increase ratio of the size of the dispatch region for a case that different dispatch regions are not overlapped. We further investigate the overlapped case and modify the dynamic programming algorithm correspondingly. We provide a discussion on the effect of the overlapping in a spatial case, where the driver and passenger locations are uniformly distributed. Experiments are conducted based on the synthetic dataset and the real-world dataset from Didi Inc. Results show that our approach can effectively reduce the expected driver pickup distance and keep the dispatching time short, which balances both passengers' and drivers' interests.**

***Index Terms*—dynamic programming, order dispatch, ride sharing, urban computing.**

I. Introduction

Nowadays, ride-sharing companies such as Uber, Lyft, or Didi have rapidly developed worldwide. A report [1] shows that the monthly number of Uber users around the world is forecasted to reach 100 million. With a large number of users, one critical challenge to these companies occurs in the order dispatch. Order dispatch is the process of matching passengers with drivers. By leveraging mobile networks and global positioning systems, the Service Provider (SP) can gather the locations of passengers and drivers, and therefore has a potential ability to optimize matching in a centralized way. An efficient dispatch scheme can not only increase passenger and driver satisfaction, but also raise the revenue of SPs. Therefore, optimizing the order dispatch system is an important problem for SPs.

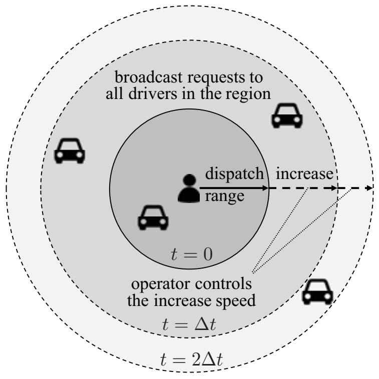

broadcast requests to all drivers in the region
dispatch range
increase
$t = 0$
operator controls the increase speed
$t = \Delta t$
$t = 2\Delta t$

Fig. 1. An illustration of our dispatch scheme.

To the best of our knowledge, two approaches are widely used in dispatch scheme designing. One is system-assigning approach and the other one is the driver-grabbing approach.

The system-assigning approach is widely used by Uber. In this approach, the SP gathers passenger requests and then assigns each request to a specific driver chosen by the system. Specifically, the dispatch process contains three steps [2]. Firstly, the SP grades the match between passengers and drivers based on their locations, history data, profile data, etc. Then, the SP sets weights to optimization factors based on identified user preferences. Finally, the SP can distribute passenger requests to drivers based on optimization scores. Each driver is informed with the pickup location, and the rating of the assigned passenger. The information about order values, such as destinations and travel lengths, would be blocked from drivers. Based on this information, each driver has approximately 20s to decide whether to accept the order. If the driver rejects, the order would be assigned to the next driver chosen by SP. The system-assigning approach tends to benefit passengers. Considering that it is passengers who directly pay SP, the SP usually weighs the optimization factor based on a passenger's preference. In contrast, the interests of drivers are usually ignored. Therefore, the system-assigning approach tends to benefit passengers and might underestimate the importance of drivers.

The driver-grabbing approach was implemented in Didi. In this approach, the SP broadcasts passenger requests (including their pickup locations, ratings, and destinations) and their ratings to nearby drivers, and lets the drivers make decisions. Drivers need to send back their decisions to SP for confirmation. It is possible that several drivers are interested in the same request. In this case, the first driver who sends a positive response (i.e. accepts the request) to the SP wins the order. In this approach, drivers tend to make decisions that benefit them the most. Therefore, the driver-grabbing approach brings more benefits to drivers.

Both the system-assigning and driver-grabbing approaches have weaknesses. In the system-assigning approach, driver preferences are ignored. Consequently, drivers may be not satisfied with the requests assigned to them, which could increase their rejection rate. Also, if a driver rejects an order, the corresponding passenger needs to wait for another driver's decision, which increases the waiting time for the passenger. In the driver-grabbing approach, drivers can simultaneously evaluate their interest in a request. On average, the waiting time for passengers might be reduced. However, some "low-value" requests might take a long time to be accepted. In addition, passengers may be not satisfied with the drivers assigned to them. For example, a driver might be relatively far from the passenger. Therefore, we propose to combine these two approaches and avoid these shortages.

In this paper, we propose an order dispatch scheme that takes advantage of both approaches. After gathering location information from passengers and drivers, the SP first chooses a set of drivers and broadcasts passenger requests to these drivers. Each request contains a pickup location and the rating of the passenger. Information related to the order value is blocked. In this way, multiple drivers can evaluate a request simultaneously, which helps to reduce passenger waiting time. If there is no driver who accepts the order, the system picks another set of drivers and broadcasts the request. This process repeats until the request is accepted by a driver or there are no more available drivers. In our scheme, the driver set is determined based on the distance between the drivers and the pickup location of the passenger. As shown in Fig. 1, all drivers in a circle region make up a driver set. The dispatch range is used to control the size of the circle (also the size of the driver set), and it could iteratively increase until the request is accepted by a driver.

A problem arises in our dispatch scheme when determining the size of the dispatch range. If the initial dispatch range is large, then there will be more drivers who can receive the request information. It increases the probability of the request being accepted and reduces the waiting time for passengers. However, the request might be accepted by the driver who is relatively far from the passenger. It increases the pickup distance between the passenger and the driver. Therefore, there is a trade-off between pickup distance and acceptance probability (or passenger waiting time). We propose a dynamic programming algorithm to adaptively adjust the size of the dispatch range.

Our contributions can be summarized as follows:

- We propose an order dispatch scheme for ride-sharing systems, which combines existing dispatch models and jointly considers benefits for drivers and passengers.
- We observe a trade-off between the pickup distance and passenger awaiting time. We propose to jointly optimize these factors and introduce an utility function.
- We first investigate a case in which different dispatch regions would not overlap, and propose an dynamic programming algorithm to optimize the utility function.
- We further consider the overlapping case, and modify the dynamic programming algorithm. In addition, we investigate the effect of overlapping.

The remainder of the paper is organized as follows. Section II introduces the notations and model used for our dispatch scheme. Section III describes our dynamic programming solution for the non-overlapping scenario. Section IV extends our solution to the overlapping scenario. Section V illustrates the experiment details. Section VI reviews the existing order dispatch schemes and how they are different from our approach. Finally, Section VII concludes the paper.

TABLE I
KEY NOTATIONS

Notations	Description
$U, \|U\| = N$	The passenger set, and its cardinality is N
$V, \|V\| = M$	The driver set, and its cardinality is M
$p_{u,v}$	The pickup probability of driver v to passenger u
Φ	The utility function
φ	The changes of the utility function
d, D	The pickup distance and its limitation
t, T	The dispatching time and its limitation
α	The adjust parameter in the utility function
$dis(\cdot, \cdot)$	The distance function

II. Model

A. Overview of our dispatch scheme

In our order dispatch scheme, the Service Provider (SP) continuously gathers passenger request information and available drivers' locations. In our model, the passenger (or user) set is denoted as $U = \{u_i \mid 1 \leq i \leq N\}$, where N is the total number of passengers. We use V to denote the set of drivers and $V = \{v_j \mid 1 \leq j \leq M\}$, where M is the number of available drivers. The pickup location requested by u_i is denoted as l_i and the corresponding destination is denoted as l'_i. The location of the driver v_j when he/she accepts the order is denoted as o_j. In addition, we define a distance function $dis(\cdot, \cdot) : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}$ that maps a tuple containing two locations into a real number representing the geometric distance between the locations. For example, $dis(o_j, l_i)$ represents the distance between a driver's location o_j and a user's pickup location l_i. The distance between user u_i and driver v_j is denoted by $d_{ij} = dis(u_i, v_j)$.

After acquiring location information, the SP starts the order dispatch process. As a combination of system-assigning and driver-grabbing approaches, the SP chooses a set of drivers and broadcast passenger requests to these drivers. In this way, multiple drivers can evaluate a request based on their preferences at the same time. It also helps to reduce passenger waiting time. Therefore, it benefits both driver and passenger interests. The driver set is determined based on the distances between drivers and the passenger's pickup location. The distance is denoted as *dispatch range* and the corresponding circle area is denoted as *dispatch region*. If the request is not accepted by any driver within a time interval Δt, the system chooses another set of drivers to broadcast the request. This process is denoted as dispatch region *expansion*. The expansion repeats until the request is accepted by a driver or temporal/spatial limitation is reached. As shown in Fig. 2, SP sets a limitation D_u for passenger u's dispatch region to avoid relatively long pickup distances. Specifically, the limitation D_u is proportional to the passenger's travel distance with a factor c set by SP. In addition, the duration of the order dispatch process should not exceed the limitation T, considering passengers have limited patience.

Different from the driver-grabbing approach, information related to order values such as passenger destinations is blocked to drivers. This is to prevent cases in which drivers only choose high-value orders. Although drivers cannot acquire a passenger's intended destination, they can set a service range (e.g. within 10 miles or 20 miles far from current locations). This is used to protect driver interests, considering that some drivers may only interested in short-distance orders.

Notice that even though the order value related information is blocked from drivers, they will still have different preferences for passenger request. For example, some drivers might prefer to pickup passengers at their familiar areas, considering it is easier to find pickup locations. Driver preferences are modeled as possibilities of accepting passenger requests.

Formally, the driver v's pickup preference on request from u is defined as follows:

***Definition** 1 (Pickup preference):* The driver v's pickup preference toward u is defined as the possibility that v will accept the request from passenger u. The probability indicates the driver v's interest toward passenger u's order.

The probability can be learned from historical data by linear regression [3]:

$$p_{u,v} = p(y = 1|\mathbf{x}_{u,v})$$

where $\mathbf{x}_{vu}$ denotes the features that are used in the prediction and y denotes whether accept the order. More details can be found in [3].

B. Problem formulation

The problem arises when the SP ties to determine the size of dispatch region and its expansion speed. On one hand, if the expansion speed is too fast, then a driver who is far from a passenger may get the order. The expected pickup distance (or the driver's idle distance) increases, which decreases the efficiency of the ride-sharing system. On the other hand, if the expansion speed is too slow, then very few drivers are informed in each round of expansion. It might take a long time before a request is accepted. Therefore, there is a trade-off between the driver's idle driving distance and the passenger's waiting time.

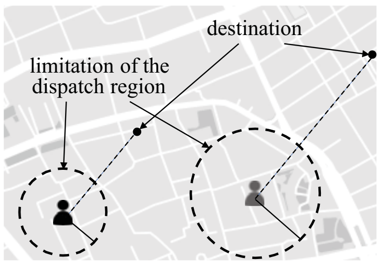


Fig. 2. An illustration of the limitation of dispatch region.

We propose to jointly optimize the idle driving distance and waiting time, and introduce a utility function to quantify the joint benefit.

***Definition** 2 (Utility function):* The cost of dispatching the order for passenger u is quantified by a utility function, which is defined as follows

$$\Phi_u = \frac{\mathrm{E}[d_u]}{D_u} + \alpha\frac{\mathrm{E}[t_u]}{T_u} = \frac{T_u\mathrm{E}[d_u] + \alpha D_u\mathrm{E}[t_u]}{D_u T_u} \tag{1}$$

where the first term is the ratio between expected pickup distances for drivers $\mathrm{E}[d_u]$ and the maximum pickup distance D_u. It represents driver's idle driving distances. The second term is the ratio between dispatch time t_u and user's maximum waiting time T_u, which represents passenger's waiting time. α is the parameter used to adjust the weights between two parts.

For simplicity, we introduce a notation $\varphi(i, j)$ to denote the changes of the utility function when expanding the region from $i \cdot \delta$ to $j \cdot \delta$ in radius. To calculate the expected distance, we define a default sequence of drivers to make a decision when they receive a user's order at the same time. By default, we assume the driver with the higher pickup probability makes a decision first. This is based on the assumption that the driver with the higher pickup probability also has higher interest in the order, and is therefore more likely to make decisions faster.

Let $r_{k,u}$ denote the increase value of u's dispatch range in the k-th expansion. The dispatch range after k expansions is $\sum_{i=1}^{k} r_{k,u}$. The value of $\mathrm{E}[d_u]$ and $\mathrm{E}[t_u]$ are determined by the choices of $r_{k,u}$. Our problem is to find out a set of values for $r_{k,u}$ such that the utility function is maximized. Formally, the problem can be formulated as follows:

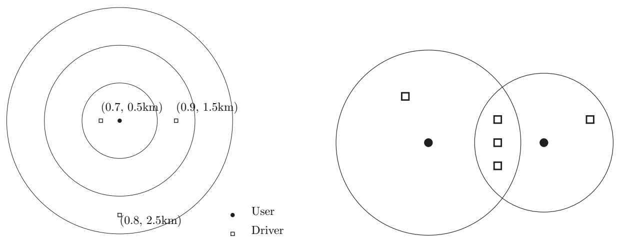
$$\min \sum_u \Phi_u$$

$$\sum_{k=1}^{|R_u|} r_{k,u} \le D_u,\ \forall u \in U \tag{2}$$

$$\Delta t|R_u| \le T_u,\ \forall u \in U \tag{3}$$

$$r_{k,u} \in \{r|r = m\delta, m \in \mathbb{N}\}, 1 \le k \le |R_u|, \forall u \in U \tag{4}$$

The objection is to minimize the overall utility function $\sum_u \Phi_u$. Eq. (2) is the constraint of the maximum dispatch region, i.e, the dispatch range after k expansions should not exceed D_u. Eq. (3) is the constraint on temporal domain, which indicates that the expansion process should be finished



(a) The non-overlapping case (b) The overlapping case

Fig. 3. The non-overlapping case and the overlapping case.

within time T_u. Eq. (4) is the discrete constraint, which indicates that the spatial step length is δ.

III. The Non-overlapping Scenario

We first consider a simple case in which the passengers' maximum dispatch regions do not overlap. In this case, we can focus on the increase ratio of the dispatch region for a single passenger. We propose a dynamic programming solution for increase ratio calculation.

First, we introduce how to calculate $\mathrm{E}(d_u)$. Specifically, $r_{k,u}$ determines drivers' order of making decisions on the request u. If multiple drivers are informed at the same time, the sequence is assigned based on the driver's certainty about his decision. The certainty is quantified by $|p - 0.5|$, where p is the probability of the driver accepting the order. The meaning behind the assumption is that a driver with $p = 0.5$ is more likely to hesitate and is slower to make a decision, while the driver with $p \approx 1$ (or $p \approx 0$) is certain to accept (or reject) the order. Drivers who are informed earlier can make decisions first. We use S_u to denote the sequence of drivers for passenger u. For example, if the three drivers listed in Table II are informed of the request at the same time, the sequence would be $S_u = \{v_2, v_3, v_1\}$. It represents that the driver v_2 first decides whether to pickup u, then the driver v_3 can make a choose, and at last, v_1 can decide. For simplification, let v'_{iu} denote the i-th driver in S_u. In our example, $v'_1 = v_2, v'_2 = v_3$, and $v'_3 = v_1$. Their corresponding preference on u are denoted as $p_{u,i}$. Then the value of $\mathrm{E}[d_u]$ can be calculated by the following equation:

$$\mathrm{E}[d_u] = \sum_{k=1}^{|S_u|} dis(u, v'_k) \prod_{i=1}^{k-1} (1 - p_{u,i}) p_{u,k}$$

We propose a dynamic programming approach to optimally solve the discretized single user's dispatch problem. Firstly, the state of the DP can be defined as $f[i][j]$, which is the optimal utility value that can be achieved if the temporal limitation is $i \cdot \Delta t$ and the spatial limitation is $j \cdot \delta$. For example, $f[6][3]$ is the optimal utility value if the dispatch region can expand 6 times and the maximum radius does not exceed 3δ, where δ represents the granularity of the discretized spatial domain. The state transfer function is defined as:

$$f[i][j] = \min_{1 \le i \le D, 1 \le j \le T} \{f[i-1][j-k] + \varphi(j-k, j), \forall 0 \le k \le j\} \quad (5)$$

Algorithm 1 The Dynamic Programming Solution.

Input: Single user request u_i, available driver set V, driver's pickup probability set P, maximum dispatch time T, expansion interval Δt, maximum dispatch range D, expansion step δ

Output: Dispatch ration increase for each time-slot

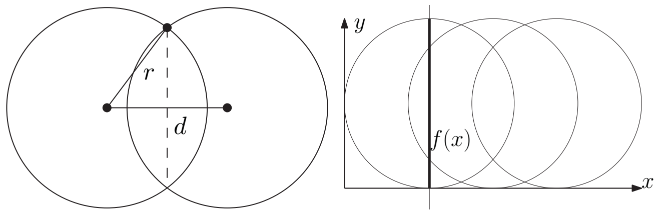
```
1:  $T \leftarrow T/\Delta t$
2:  $D \leftarrow D/\delta$
3:  $f \leftarrow (T+1)$-by-$(D+1)$ zero matrix
4:  for $0 \le i \le T$ do
5:      for $1 \le j \le D$ do
6:          $f[i][j] \leftarrow +\infty$
7:  for $1 \le i \le T$ do
8:      for $1 \le j \le D$ do
9:          for $1 \le k \le j$ do
10:             $f[i][j] = \min\{f[i-1][j-k] + \varphi(j-k, j), f[i][j]\}$
11: return $f[T][D]$
```

TABLE II
Driver information

Driver #	1	2	3
Distance to user	0.5	1.5	2.5
Probability to accept the order	0.7	0.9	0.8

We use the example shown in Fig. 3(a) to illustrate the procedure for the proposed solution. There are 3 drivers in the figure. The 2-tuples on the figure indicate the probability of the driver accepting the order (which can be learned form historical data) and the distance to the user's pickup location. This information is summarized in Table 1. The maximum dispatch range is 3km in radius and the smallest increase step is 1km in radius. The initial dispatch range is 0km in radius, and the range can expand once for each $\Delta t = 20$s. The user's maximum waiting time is 2 min, which means the number of expansions is limited by T =2min/20s = 6 in the temporal domain. In the example, α is set to 1, which means the driver's interests and user's interests are equally important.

Based on the example, we illustrate how to calculate the utility function. Assume we choose to directly expand the dispatch range from 0 to 3km. In this case, the user's order is broadcast to drivers 1, 2, and 3 simultaneously during the first time interval. The expected driver pickup distance $\mathrm{E}[d] = 0.9 \cdot 1.5 + (1-0.9)0.8 \cdot 2.5 + (1-0.9)(1-0.8)0.7 \cdot 0.5 = 1.557$. The first term in the equation means driver 2 has a 0.9 probability to win the order. The second term refers to the expected pickup distance of driver 3, where $(1 - 0.9)$ represents driver 2 rejecting the order and 0.8 represents driver 3 accepting the order. The third term represents driver 1. The maximum pickup distance D is 3. In addition, in this example, we only expand the region once. Therefore $\mathrm{E}[t] = (1 - (1-0.9)(1-0.8)(1-0.7)) \cdot 1 = 0.994$, where $(1-(1-0.9)(1-0.8)(1-0.7))$ is the probability of that the order is accepted. The probability of that the order is rejected by all drivers is constant for every possible dispatch. Therefore, it



(a) The two-passenger case (b) The general case

Fig. 4. Calculating the expected number of drivers.

is not included in the utility function. After plugging these values into the utility function and set $\alpha = 1$, we then have $\Phi = 1.557/3 + 1 \cdot 0.994/6 = 0.685$.

By plugging those values into state transfer equations and setting $alpha = 1$, we can conclude that the optimal strategy is first expanding the radius of the dispatch range by 1km, then expanding the radius again by 2km, and then the range reaches the spatial limitation. In total, there are 2 expansions and the overall utility function is 0.486. As an examination, the values of the overall utility function of other expansion plans are tested. If we expand the radius by 1km 3 times, the utility value is 0.490. If we first expand the radius by 2km and then by 1km, then the utility value is 0.651. Therefore, it can be examined that the DP generates the optimal expansion plan.

The proposed dynamic programming approach can adaptively adjust the expansion ratio based on density distribution of drivers. Our algorithm provides the dispatch system with the ability to address changing environments and varying user densities. In areas where the driver distribution is dense, the passenger request is more likely to be accepted by a driver even if the dispatch region grows slowly. In contrast, in the areas where the number of drivers is relatively small, the dispatch region should grow more aggressively. Otherwise, the request may remain unanswered after several rounds of expansions. Our DP algorithm can adjust the growth rate adaptively based on investigating the trade-off between the expected pickup distance and the probability of the order being accepted.

Theorem 1: The time complexity of the dynamic programming is $O(M^2 n^3)$, where $n = \max\{D, T\}$.

Proof: The state function $f[i][j]$ has two dimensions, which represent temporal and spatial boundaries of dispatch regions. In the state transfer function, the spatial state has to be traversed once in each round of expansion. The reason for this is that the increase ratio of the dispatch region is unlimited. Therefore, at most, $O(n^3)$ rounds are needed to finish the DP process. In addition, in each round, the calculation of $\mathrm{E}(d)$ and $\mathrm{E}(t)$ needs at most $O(M^2)$ times of adds or multiplications. Therefore, the overall time complexity is $O(M^2 n^3)$ ■

IV. The Overlapping Scenario

In this section, we discuss a more realistic scenario in which the maximum dispatch regions of passengers may be overlapped with each other, which brings challenges in pickup distance estimation. We propose to extend the dynamic programming solution and evaluate its differences with the non-overlapping case.

A. The extension of dynamic programming

In this scenario, the dispatch regions of different passengers could be overlapped as shown in Fig.3(b). As shown in Fig.3(b), the dispatch regions of two users are overlapped. The 3 drivers in the overlapped area belong to both dispatch regions of two users. For simplification, we propose to synchronize the expansions of dispatch regions. Specifically, after the system gathers enough passenger requests, it starts the expansion process for these requests simultaneously. In addition, expansions of all requests share the same frequency and increase rate, which aims to reduce the number of decision parameters. The remaining analysis in this section is based on these assumptions.

A challenge arises when calculating drivers' preferences for each request. If a driver is located in the intersection of several dispatch regions, the driver can receive more than one request at the same time. We denote these requests received by the driver as potential requests. The driver can accept at most one potential request. Although the driver's preference toward each potential request is known, it is hard to quantify the joint possibility distribution w.r.t the driver's preference for all potential requests. For example, assume that there are two potential requests, and the driver has a p_1, and p_2 probability to accept each corresponding request if it is evaluated independently. Calculating the possibility of accepting each request is not simple. The driver's decisions on potential requests are not mutually independent since the driver can accept at most one request.

Before extending the dynamic programming solution, we first deal with the challenge of the preference estimation. The reason that we cannot assume the driver's decisions are independent is that the driver could at most accept one request. The probability of rejecting all request should be the same as the case where independence holds. Under this assumption, in the example given above, the probability of rejecting both requests is $(1 - p_1)(1 - p_2)$. It is reasonable since the driver's decision to reject all requests is not affected by the constraint that at most one request can be chosen. As for the possibility of accepting each potential request, we assume they are weighted based on the driver's independent preferences. We use the aforementioned example to illustrate the idea. In the example, the probability of accepting a request is $p_+ = 1 - (1 - p_1)(1 - p_2)$. We assume the probability of accepting the first request is $p_1/(p_1 + p_2) \cdot p_+$ and the probability of accepting the first request is $p_2/(p_1 + p_2) \cdot p_+$. In this way, the sum of the probabilities of all events (accepting the first request, accepting the second request, and rejecting all) is 1. Based on this model, the drivers' preferences for each request can be calculated with linear time complexity.

After establishing the probability model, the DP solution can be extended to the case where overlapping is considered. The skeleton of the DP remains the same, the difference is

Algorithm 2 The Extended DP Solution.

```
Input:  Single user request $u_i$, available driver set $V$, driver's pickup probability set $P$, maximum dispatch time $W$, expansion internal length $\Delta t$, maximum dispatch range $D$
Output: Dispatch ration increase for each time-slot
1: Similar as Algorithm 1, except the line 10:
2: $f[i][j] = \min\{f[i-1][j-k] + \sum_u \varphi_u(j-k, j), f[i][j]\}$
```

that the calculation of the φ function is modified to take multiple passengers into account. Let φ_u denote the utility function of passenger u. Then, we have $\varphi = \sum_u \varphi_u$. With this modification, the DP could be used for the overlapping case.

B. *Evaluating the affect of overlapping*

In the overlapping case, the drivers in the dispatch region might receive multiple passenger requests, and the probability of accepting a specific request is reduced compared to the non-overlapping case. We evaluate the gap between these two cases under an ideal distribution. In the evaluation, we assume locations of drivers and passengers are uniformly distributed and drivers and passengers are homogeneous. Specifically, drivers have the same preferences for passengers, i.e. share the same acceptance probability. In this spatial case, the difference between the two cases can be revealed by the average number of drivers in each dispatch region.

Considering that drivers are uniformly distributed, the number of drivers is proportional to the area of the dispatch region. For the non-overlapping case, the dispatch region is a circle and the area of the circle can be easily calculated. For the overlapping case, the dispatch region is the union of circles. Fig. 4 shows the overlapping of two passengers. For this two-passenger overlapping case, the area of the dispatch region has a closed form expression. Formally, let d denote the distance between two passengers and let r denote the radius of each passenger's dispatch circle. Then, the size of the union of these two circles is:

$$S = (2\pi - \arccos\frac{d}{2r})r^2 + d\sqrt{r^2 - \frac{d^2}{4}} \tag{6}$$

The number of drivers in the region is proportional to the region size. To examine the equation, we compare the numerical results with the Monte Carlo simulation [4]. The Monte Carlo simulation repeats random sampling to obtain numerical results. The simulation results validates the equation.

From the equation, we can find out that $S = O(r^2)$, which is the same order as the area of a circle. This result indicates that the overlapping will not affect the order of the number of drivers in the dispatch region, for the two-passenger case.

If there are more than two passengers, it is hard to have a closed form expression on the combined area of overlapping dispatch regions. Therefore, we use the adaptive Simpson's method [5] to calculate the numerical integration. To use the numerical method, we could build a coordinate system such that the dispatch region lies in the first quadrant, as shown in Fig. 4(b). Let function $f(x_0)$ denote the length of the intersection between $x = x_0$ and the dispatch region. Then, the area of the dispatch region is $R = \int_{-\infty}^{+\infty} f(x)\mathrm{d}x$. Although the closed form expression of R is hard to obtain, we could use the Simpson's rule [6] to calculate the numerical integration. The Simpson's rule can be expressed as following:

$$\int_x^{x+\Delta x} f(x)\mathrm{d}x \approx \frac{\Delta x}{6}\left[f(x) + 4f\left(\frac{2x+\Delta x}{2}\right) + f(x+\Delta x)\right] \tag{7}$$

It can be used to estimate the numerical integration value when Δx is small. The adaptive Simpson's method [5] is based on the same idea, and could adaptively set the value of Δx for speedup.

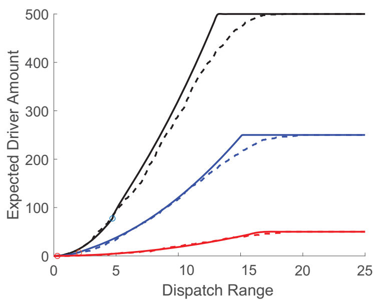


Fig. 5. The non-overlapping case (in solid lines) vs. the overlapping case (in dashed lines).

To evaluate the effect of overlapping, we compare the number of drivers covered in the dispatch region. In Fig. 5, we use solid lines to indicate the number of drivers in the dispatch region when there is no overlapping, and use dashed lines to represent the number of drivers when overlapping occurs. The line color is used to indicate the density of drivers. The black, blue, and red lines represent dense, medium, and sparse driver distributions, respectively. We set the maximum number of drivers in the evaluation. Therefore, all lines are eventually stable. From the figure, we can find out that the overlapping reduces the number of drivers covered in the dispatch region, while the effect is limited. The evolution of the number of drivers in both overlapping and non-overlapping cases has the same trend and increase order.

In summary, we discussed the multiple-passenger case (or the overlapping case) in this section, and proposed a method to extend the dynamic programming solution to this case. In addition, the effect of overlapping is evaluated in a spatial case, where the driver locations and passenger requests are uniformly distributed. By analyzing the expected number of drivers in the dispatch region, we find out that the overlapping will not change the order of drivers covered in the dispatch region.

V. Experiment

A. *Pickup probability prediction*

Predicting the probability that a driver accepts each request is crucial to our scheme. It provides the basis for calculating

TABLE III
THE DIDI DATASET

Data Source	Didi's trajectory data in Chengdu City
Time Span	11/1/2016 - 11/30/2016
Number of orders	150,000
Average travel distance	8.43 km

our utility function. Zhang et. al have studied this prediction problem in [3]. They propose to use either linear regression or a gradient boosted decision tree to build the predictor. Both these two models are attempted in our experiment. However, we use fewer features to train our predictor since some factors they used are not contained in our dataset.

Specifically, the features we used are listed as follows:

Spatial domain features: the driver's pickup distance, the pickup location's POI (business area, airport, shopping mall, etc.), number of drivers and requests in nearby regions, and a driver's historical order acceptance rate in the area.

Temporal domain features: time of a day, day of a week, the driver's recent order acceptance rate, and the driver's long-term order acceptance rate.

B. Dataset setup

The dataset we used in our experiment is obtained from Didi's GAIA open data program [7]. It contains the trajectory data and passenger request data in Chengdu city from Nov. 1 - Nov. 30, 2016. We extract the trajectory and request data of one day to conduct our experiment. The original dataset is large, and we extract partial records from the dataset to build the Didi dataset. The statistics of the data used in the experiment are shown in Table III.

Originally, the dataset did not contain information about each driver's rejected requests. However, it is important ground-truth data used for the predictor. To reveal such information, we build a simulator based on the given dataset. Besides, the maximum acceptable dispatching time is not included in the dataset. We simply assume that each passenger's maximum acceptable dispatching time is proportional to the travel time that is recorded in the dataset. In the experiment, we assume the rate is 0.1. For example, if a passenger's travel time is 30 min, then we assume the maximum acceptable dispatching time of the corresponding passenger is 3 min.

In the simulator, we assume each passenger's order is broadcasted to all idle drivers in the nearby regions. The nearby region used in our experiment consists of the locations within a 5km range of the request's origin location. Since both passenger requests and driver states are contained in the dataset, the information about the requests rejected by a driver can be simulated. Using a simulator to reveal this information is not as precise as using real-world data, so this information is only used to feed the predictor. It can be used to demonstrate the time complexity and optimality of our proposed scheme. All companion algorithms are run on the same dataset. Therefore, the performance companion results are still meaningful. Besides the original dataset, to reduce the density of the drivers, we also randomly choose some driver locations from the Didi dataset. The chosen driver locations form a sparse Didi dataset.

We also set up a synthetic dataset, where the locations of passenger requests and drivers are uniformly distributed. In the dataset, the distribution region of drivers and passenger requests is 0 - 15 miles in both longitude and latitude. The density of drivers in the dataset is controlled by the number of drivers. In the dense and sparse datasets, the ratio between the number of divers and the number of passengers are 15 and 5, respectively.

C. Experiment setup

Unlike directly assigning user requests to drivers or just broadcasting the request to all nearby drivers, our scheme combines these two approaches. To compare our scheme with these approaches, we apply several comparison algorithms in our experiment. To simulate the scheme that directly assigns requests to drivers, we introduce a greedy approach.

In the greedy approach, the dispatch center simply sorts drivers based on their distances to the passenger, and then assigns requests to drivers following the distance sequence, i.e., the nearer driver has a higher priority. Similarly, the driver can choose whether to accept the order. The rejected request is inserted to the request queue and would be assigned in the following round.

In the broadcasting scheme, passenger requests are broadcasted to all idle drivers in the nearby region. The order goes to the first driver who gives a positive response to the request.

To compare these dispatch schemes, we build a test platform. In the platform, the passenger requests are extracted from the Didi dataset. Drivers' preferences for different requests are modeled by the pickup probability learned by the predictor. For assigning approaches, each driver has Δt time interval to make a decision. If the driver does not respond to the request in the given time interval, the order is treated as rejected by default.

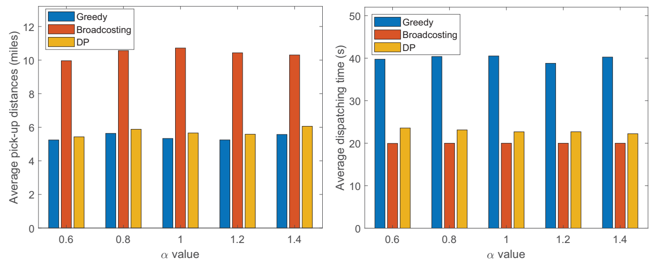
In each set of experiment, we vary the value of $alpha$ from 0.6 to 1.4. As defined in Eq. (1), the α value adjusts the importance between the pickup distance and the dispatching time. When $\alpha = 1$, these two factors share the same weight.

In our experiment, we compare different schemes on average driver pickup distance, average dispatching time and average utility function value. The pickup distance is the distance between each passenger's location and the assigned driver's location. The dispatching time is the time interval between a passenger starting a request and the request being accepted. The utility value is calculated based on Eq. (1).

D. Results

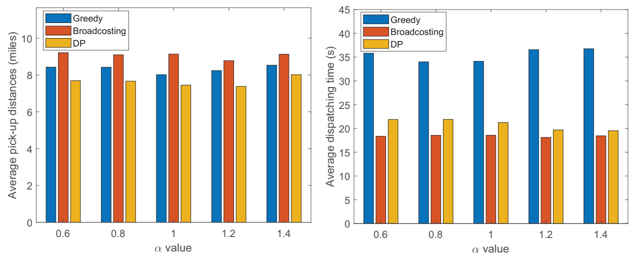
Experiment results are shown in Fig. 6, Fig. 7, and Fig. 8. Each result shown is the average outcome of 50 repeated experiments.

Results in Fig. 6 illustrate the comparison between pickup distance and dispatching time, which represents the interest of passengers and drivers correspondingly. From the results we can determine that the greedy approach can reduce the pickup



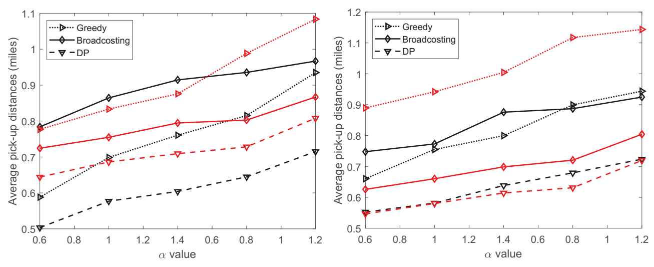
(a) Pickup distance (b) Dispatching time

Fig. 6. Performance comparison on the dense distribution dataset.



(a) Pickup distance (b) Dispatching time

Fig. 7. Performance comparison on the sparse distribution dataset.



(a) On the synthetic dataset (b) On the Didi dataset

Fig. 8. Performance comparison in terms of the utility value.

distance when comparing with the other two approaches. However, the dispatching time is larger than those of other algorithms. The reason is that the greedy approach first assigns orders to nearest drivers, but ignores the pickup probability of these drivers. In contrast, the broadcasting approach achieves the smallest dispatching time, but its performance on pickup distance is worse than others. It is because more drivers can start grabbing orders once the passengers submit their requests. These drivers can make decisions in parallel and therefore have the same dispatching time, while the drivers nearby do not have any priority. Our approach can balance the performance with respect to pickup distance and dispatching time. It can be shown more clearly by comparing the algorithm outcomes on the utility value as shown in Fig. 8. In Fig. 7, we show the comparison result on the sparse distribution dataset, where the ratio between the number of drivers and the number of passengers is reduced to 5. Comparing the result shown in Fig. 6, we can find out that the pickup distances are increased when dispatching with our DP scheme and the greedy scheme. It is because the distance between the passenger and the nearest driver is increased. The pickup distances calculated with the broadcasting scheme remains, since the limitation of the dispatch region is unchanged. Finally, we can also find out that our DP scheme could balance the performance in terms of pickup distance and dispatch time.

Fig. 8 shows the algorithm comparison results on the utility function value. Fig. 8(a) shows the simulation results on the synthetic dataset where the drivers are uniformly distributed. Fig. 8(b) illustrates the simulation results on the Didi dataset. In both figures, lines in black indicate the results of the dense distribution dataset and lines in red indicates the results of the sparse distribution dataset. From both cases, we can find out that our DP scheme achieves the best performances, i,e, the smallest utility values.

Through the experimental results, we can find out that our scheme jointly considers the benefits of passengers and drivers. When measuring the dispatch system performance in terms of the utility function value, our DP algorithm achieves the smallest utility value and outperforms other comparisons.

VI. Related work

Taxi or car-sharing dispatch schemes have been widely studied from different perspectives [3, 8–15]. Some researches start from the passengers' perspective and aim to increase their satisfaction, while some others focus on drivers' interests, and propose to help idle drivers to efficiently find potential passengers. What's more, schemes in [16] propose to jointly consider the interests of both passengers and drivers.

Order dispatch schemes in [3, 8, 10] focus on optimizing customers' interests. Their dispatch scheme aims to increase passengers' satisfaction and minimize their waiting time. A-mong them, [8, 10] propose to minimize a passenger's waiting time. [10] sequentially assigns a passenger's request to the nearest driver based on GPS data. [8] aims to globally minimize the pickup distance. The dispatch scheme in [3] proposes a HillClimbing algorithm to enhance the user's experience by maximizing the global success rate of passenger requests. [15] also aims to improve user satisfaction; it pays more attention to the platforms' incomes by weighing each request by its order value. The long-term gross merchandise volume of the platfrom is maximized by their offline learning and online matching approach.

In addition, schemes in [11–13] mainly consider drivers' benefits. Their schemes guide idle drivers to find new passengers. [11] aims to reduce drivers' idle driving distance and to maintain the quality of service. A receding horizon control framework is proposed. [12] proposes a route recommendation system for taxi drivers, and the goal is to maximize a driver's profit by reducing their driving time before finding a passenger. [13] minimizes the distance to the anticipated passenger through the Monte Carlo tree search. These schemes only consider drivers' benefits and ignores a passenger's waiting time. It may be worthwhile to dispatch a driver to pick up a passenger who is not near to the driver but has waited for a long time. Different from their dispatch schemes, our scheme jointly considers the interests of both passengers and drivers.

Besides considering the benefit of either passengers or drivers, [16] also proposes to jointly consider the interests of two groups. [16] introduces an utility function that contains a passenger's waiting time and a driver's idle drive distance to quantify the joint benefit. The dispatch center has knowledge of both idle driver locations and passenger requests. Based on this information, a weighted bipartite graph between user requests and drivers can be established. The utility function is maximized based on the idea of Kuhn-Munkres algorithm [17]. However, their scheme assigns each passenger's request to a specific driver chosen by the algorithm. Our dispatch scheme broadcasts each request to multiple drivers and the order goes to the driver who accepts the request first. In our scheme, the responding time for each request can be reduced since multiple drivers can make their decisions simultaneously.

Other research topics are related to taxi/car sharing, including passenger request demand prediction [18, 19], routing scheme designs for carpooling [20–24], etc. The efficiency of the order dispatch scheme may be further improved by taking into account request locations and/or carpooling.

VII. Conclusion

In this paper, we focus on the order dispatch problem for ride-sharing systems. The existing driver-grabbing approach may lead to a high un-service ratio for low-profit orders. The system-assigning approach may reduce the time efficiency of the dispatching system, and increase passenger waiting time. In this paper, we propose a new dispatch scheme to overcome the flaws of the two existing approaches. The passenger request is broadcast in the dispatch region, and the size of dispatch region increases in each time slot. In the scheme design, we consider to jointly optimize benefits of passengers and drivers. To quantify the joint benefit, a utility function is proposed. Based on the trade-off between time efficiency and pickup distance, we first propose a dynamic programming algorithm to optimize the utility function for the non-overlapping case. For the overlapping case, we propose a model to evaluate the preference of drivers in the overlapped area, and then extend the DP solution. Experiments on a real-world dataset show that our scheme can efficiently balance the passenger's waiting time for dispatching and the driver's pickup distance. Our DP algorithm outperforms comparison algorithms in terms of utility function values.

Acknowledgement

This research was supported in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS 1651947, and CNS 1564128.

References

[1] Monthly number of uber's active users worldwide from 2016 to 2018. [Online]. Available: https://www.statista.com/statistics/833743/us-users-ride-sharing-services/

[2] M. Truong, D. Purdy, and R. Mawas, "Dispatch system for matching drivers and users," Jan. 12 2017, US Patent App. 14/793,593.

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

[4] D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of monte carlo methods*. John Wiley & Sons, 2013, vol. 706.

[5] G. F. Kuncir, "Algorithm 103: Simpson's rule integrator," *Communications of the ACM*, vol. 5, no. 6, p. 347, 1962.

[6] E. Süli and D. F. Mayers, *An introduction to numerical analysis*. Cambridge university press, 2003.

[7] Data source: Didi chuxing gaia open dataset initiative. [Online]. Available: https://gaia.didichuxing.com

[8] K. T. Seow, N. H. Dang, and D. H. Lee, "A collaborative multi-agent taxi-dispatch system," *IEEE Transactions on Automation Science & Engineering*, vol. 7, no. 3, pp. 607–616, 2010.

[9] Z. Liao, "Real-time taxi dispatching using global positioning systems," *Communications of the ACM*, vol. 46, no. 5, pp. pgs. 81–83, 2003.

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

[11] F. Miao, S. Han, S. Lin, J. A. Stankovic, D. Zhang, S. Munir, H. Huang, T. He, and G. J. Pappas, "Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 463–478, 2016.

[12] M. Qu, H. Zhu, J. Liu, G. Liu, and H. Xiong, "A cost-effective recommender system for taxi drivers," in *ACM SIGKDD*, 2014.

[13] N. Garg and S. Ranu, "Route recommendations for idle taxi drivers: Find me the shortest route to a customer!" in *ACM SIGKDD*, 2018, pp. 1425–1434.

[14] Y. Duan, J. Wu, and H. Zheng, "A greedy approach for carpool scheduling optimisation in smart cities," *International Journal of Parallel, Emergent and Distributed Systems*, pp. 1–15, 2018.

[15] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," *arXiv preprint arXiv:1802.06444*, 2018.

[16] G. Gao, M. Xiao, and Z. Zhao, "Optimal multi-taxi dispatch for mobile taxi-hailing systems," in *IEEE ICPP*, 2016, pp. 294–303.

[17] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of The Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[18] Y. Tong, Y. Chen, Z. Zhou, L. Chen, J. Wang, Q. Yang, J. Ye, and W. Lv, "The simpler the better: A unified approach to predicting original taxi demands based on large-scale online platforms," in *ACM SIGKDD*, 2017, pp. 1653–1662.

[19] L. Moreiramatias, J. Gama, M. Ferreira, J. Mendesmoreira, and L. Damas, "Predicting taxipassenger demand using streaming data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1393–1402, 2013.

[20] 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.

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

[22] I. Jindal, Z. Qin, X. Chen, M. Nokleby, and J. Ye, "Optimizing taxi carpool policies via reinforcement learning and spatio-temporal mining," *arXiv preprint arXiv:1811.04345*, 2018.

[23] D. Zhang, T. He, Y. Liu, S. Lin, and J. A. Stankovic, "A carpooling recommendation system for taxicab services," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 254–266, 2014.

[24] Y. Duan, T. Mosharraf, J. Wu, and H. Zheng, "Optimizing carpool scheduling algorithm through partition merging," in *IEEE ICC*, 2018, pp. 1–6.