A domain-of-influence based pricing strategy for task assignment in crowdsourcing package delivery

Zhifeng Zhou  |  Rong Chen  |  Shikai Guo

College of Information Science and Technology, Dalian Maritime University, Dalian, China

Correspondence
Rong Chen, College of Information Science and Technology, Dalian Maritime University, No.1 Linghai Road, Dalian 116026, China.
Email: rchen@dlmu.edu.cn

Funding information
National Natural Science Foundation of China, Grant/Award Numbers: 61672122, 61902050, 61906027; Fundamental Research Funds for the Central Universities, Grant/Award Number: 3132019355; Next-Generation Internet Innovation Project of CERNET, Grant/Award Number: NGII20181205; China Postdoctoral Science Foundation Funded Project, Grant/Award Number: 2019M661080

Abstract
Crowdsourced package delivery has gained great interest from the logistics industry and academia due to its significant economic and environmental impact. However, there are few research achievements about incentive mechanism to motivate people to participate. A novel domain-of-influence based pricing strategy for crowdsourced delivery is proposed. The three-stage package delivery framework is extended with the proposed pricing algorithm, which can iteratively figure out the price that represents a state of balance between the package demand and driver supply. To create better matching, even hyperbolic temporal discounting function is employed to estimate the driver's perceived reward to accept the package. The performance is evaluated using the Jinan dataset and real delivery data. Results show that economic utility and stable assignment rate have been increased by over 9% and over 6%, respectively, while the average delivery time and average delivery price have also been improved.

1 INTRODUCTION

Advancements in the size and popularity of the e-commerce industry imply that working intelligently is more crucial than ever to determine business success of the supply chain and logistics [1–3]. Crowdsourced package delivery (CPD) is thus emerging as the intelligent technology that allows drivers to provide on-the-way delivery by using the excess capacity of their vehicles. Crowdsourcing upgrades the operations and logistics among companies such as Amazon, DHL, SF, BestBuy and Macy, to the state of fast-paced delivery services and reduced transportation costs.

A surge of research into crowdsourced delivery service has been developed. Most studies concerned how dynamic requests can be better integrated into delivery routes while minimizing delivery time cost or maximizing the number of transshipments [4–6]. Arslan et al. [7] presented a rolling horizon framework and developed an exact solution approach to repeatedly solve the various versions of the offline problem with ad-hoc drivers. Chen et al. [8,9] proposed two baseline algorithms and an online adaptive scheduling algorithm AdaPlan to deliver packages by using occupied taxis without degrading the quality of passenger experience. However, one challenging aspect of crowdsourced delivery service is the incentive mechanism to motivate people to participate.

One appealing solution is to price the package delivery tasks more intelligently. The delivery tasks can be analogous to spatial crowdsourcing (SC) tasks because they are also associated with specific spatial and temporal information. To maximize the total revenue of the SC market, the platform typically decides the price per unit distance for the tasks and assigns tasks if only their requesters accept the unit prices [10–12]. Most recently, Xia et al. [13] established a task reward pricing model by a piecewise function based on tasks’ expected completion time and deadline, but they fail to consider the dynamic pricing for tasks. As a result of the spatiotemporal distributions of workers and tasks, each local market often varies in demand and supply, posing a need for dynamic pricing in each local market. Some crowdsourcing applications (e.g. Uber and DiDi) have developed grid-based dynamic pricing strategy, that is, partitioning the entire space into a fixed number of grids. Tong et al. [14] proposed a grid-based dynamic pricing model that diversified unit price for tasks in each grid by estimating the demand (tasks) and optimizing the supply (workers). However, delivery tasks distinguish themselves from typical SC tasks in that they require drivers not only to travel to an origin (e.g. retailer) but also to carry a
Problem statement

PROBLEM STATEMENT

p wants to get more reward. An improvement is returned to assign d five drivers ing problem in the context of CPD. Take a CPD problem with moting us to develop a new dynamic pricing method. Therefore, the typical dynamic pricing methods are limited to grid-based approaches to model the requests that are on their way of a pre-planned trip and bring type of matching is en-route; drivers are matched with delivery package from the origin to the destination (e.g. shopper). This type of matching is en-route; drivers are matched with delivery requests that are on their way of a pre-planned trip and bring a small possible deviation. In addition, present dynamic pricing methods are limited to grid-based approaches to model the demand and supply, which set the same price for packages in the same grid. Therefore, the typical dynamic pricing methods are incapable of achieving personalized pricing for packages, promoting us to develop a new dynamic pricing method.

The objective of this paper is to deal with the dynamic pricing problem in the context of CPD. Take a CPD problem with five drivers d1, ..., d5 and five packages p1, ..., p5 as an example, as shown in Figure 1. Typically the SC platform has a fixed unit price and assigns packages to the nearest drivers, so it returns an assignment < p4, d2 > because p4 is farther d2 from than p4. This means that d2 should stop his/her trip DriverTrip(d2) (denoted by the green dashed arrow) to take a detour Pickup(p4, d2) -> PackageTrip(p4) -> Return(p4, d2), that is, picking up p4, delivering it as addressed and returning to d2’s destination (denoted by the blue solid arrow). Note that current assignment is likely to leave one of p1, p2 and p3 unallocated because there are fewer drivers in their neighbour.

What if we have a dynamic pricing strategy? In reality, the price is influenced by the demand and supply, and the best known way is when demand is high the price of supply tends to go up. So we prefer a varying price dependent on fluctuation of the supply (drivers) and the demand (packages) in a domain-of-influence (DoI), denoted by a dashed circular region around each package. There are five DoIs in Figure 1, centred on p1, ..., p5 with radius r1, ..., r5, respectively. Note that the package-driver ratio is 3 in the DoI of p1, and 2/3 in that of p4. It is intuitive that the higher package-driver ratio leads to higher price. So some driver d2 has an incentive to take the yellow trip to deliver p1 instead of the blue one to deliver p4 because d2 wants to get more reward. An improvement is returned to assign p1 to d2 and p4 to d5/d4, and thus more packages can be allotted. Accordingly, the platform profit is increased because more drivers are motivated to participate, leading to the reduction in overall delivery cost.

Crowdsourced package delivery with pricing strategy (CPD-P) is still unexplored due to several challenges such as the requester’s budget, the unknown fluctuation of drivers and packages, and the uncertainty of the driver’s acceptance. The pricing model has price and spatial constraints on the drivers to ensure that total price is not over budget. To address these challenges, we propose a novel DoI-based pricing strategy (DoIPrice) by considering the varying package-driver ratio in a local region with the aim to maximize the stable matches and minimize the delivery cost. Based on the proposed pricing strategy, we select the candidate drivers for each package while obeying the time and price constraint. The final package-driver matches are obtained by combining any algorithm that computes the maximum flow and the linear programming algorithm that minimizes the cost of the flow, and further refined by the estimated driver’s perceived reward.

The CPD-P investigated in this study is significantly different from the CPD problem with fixed unit price [15–17]. In summary, we have made the following contributions.

1. This paper proposes a DoI-based dynamic pricing algorithm. To our best knowledge, we are the first to use the dynamic pricing approach in CPD to characterize the trade-off between the package demand and driver supply in the local market. Moreover, the hyperbolic temporal discounting function is used to figure out stable matches that further advance existing work.

2. We develop a three-stage framework consisting of the DoI-based pricing, matching pruning and package assignment algorithms to figure out the best-possible package-driver matching.

The rest of the paper is organized as follows. The problem statement and formulation is described in Section 2. Section 3 presents the details of the pricing and reward model. The three-stage framework is proposed in Section 4. Experimental results are shown in Section 5. Section 6 introduces related works. We conclude this work in Section 7.

2 | PROBLEM STATEMENT AND FORMULATION

This section describes our CPD-P problem and introduces the important notations to formulate the CPD-P problem.

2.1 | Problem statement

The CPD is the economic and eco-friendly solution of package delivery through an open call for any driver to participate in delivering a package. A driver that accepts the invitation should voluntarily take a detour to pick up that package, deliver it as addressed and then return to the original route, while it is up to the CPD platform, based on the status of road network and
volunteer drivers’ performance, to decide the right persons to perform the tasks.

According to principles of transport system modelling, the road network and its performances are represented by a graph $G$ that represents the transport network in terms of streets and intersections, and cost functions (i.e. travel times and costs).

**Definition 1. (Road Network)** A road network is a directed graph $G = (V, E)$, where $V$ is the node set and $E$ is the arc set. Each node $v \in V$ represents a unique spatial location on a Euclidean coordinate plane, denoted by its longitude and latitude GPS coordinates, and each arc $e \in E$ represents road segment denoted by a pair of a start node and an end node.

When calling for volunteer drivers, each package delivery request $p = < \text{ori}_p, \text{des}_p, \text{maxWT}_p, \text{maxR}_p >$ is published with the origin $\text{ori}_p$ and the destination $\text{des}_p$ of that package, the maximum waiting time $\text{maxWT}_p$ for a driver to pick up that package, and the maximum reward $\text{maxR}_p$ that a requestor likes to pay, whereas every volunteer driver $d$ is willing to announce his/her original route $< \text{ori}_d, \text{des}_d >$, referring to the origin $\text{ori}_d$ and the destination $\text{des}_d$ on the coordinate plane.

Provide with package delivery requests and driver announcements to create a bipartite graph as did in most related studies [14,17,19–21], the CPD platform is to find matches between packages and drivers that minimize the delivery cost while obeying constraints over the maximum waiting time and the maximum reward. Accordingly, the CPD platform can use a bipartite graph matching algorithm to obtain matches in polynomial time, and schedule the drivers to deliver packages with the fixed-unit-price contracts.

**Definition 2. (Matching Set)** Given a bipartite graph of a set $P$ of packages and a set $D$ of drivers, the matching set $M$ consists of matching pairs $< p, d > | \text{WT}(p, d) \leq \text{maxWT}_p$ and $\text{Price}(p, d) \leq \text{maxR}_p$, for any $p \in P$ and $d \in D$, where $\text{WT}(p, d)$ is the travel time from driver $d$ to package $p$, $\text{Price}(p, d)$ is the reward to driver $d$ and $\text{TC}(p, d)$ is the total time cost of package delivery by driver $d$.

The dynamic pricing strategy is introduced to extend the CPD problem with the aim to motivating drivers to prefer the packages apart from the others, as illustrated by the motivational example in Figure 1. We note that a crowdsourcing market is a CPD problem with the aim to motivating drivers to prefer the packages and drivers can be matched to each other only if requesters and drivers can be matched to each other if they both have strong incentives to be. So the CPD-P aims to generate stable matches through a dynamic pricing strategy.

**Definition 3. (Stable Matching)** Given a matching set $M$, a subset $A \subseteq M$ is stable if every pair $< p, d > \in A$ satisfies that the incentive payment $\text{Price}(p, d)$ meets the driver’s reward expectation on package $p$.

A driver’s preference of a package is profit-driven; based on the temporal discounting theory [22], the immediate reward decreases with the increasing amount of delay time. So the stable matching is to associate a package with a varying price determined by iterative adjustment of $\text{maxR}_p$ based on the driver supply and package demand in the local region (i.e. DoI) of that package. Since higher-priced packages are more appealing to drivers, they are motivated to move away from a local region, in the long run leading to the balance of driver supply and package demand.

2.2 | Problem formulation

The optimization objective of the CPD-P problem is to maximize the number of stable matches that not only minimizes the delivery cost but also increases the economic utility. Accordingly, the CPD-P problem consists of the maximum task assignment (MTA) and the minimization of delivery cost. Both can be formalized as the minimum-cost maximum flow (MCMF) problem on a flow network, in which the flow from a source node to a sink node is maximized in a directed capacitated network with arc costs subject to the constraint that the total cost of flow should be within a budget. To obtain good stable matches in our case, we start from the maximum package assignment that allows more stable matches, then, we optimize the delivery cost by minimizing the total cost of the flows.

For convenience, Table 1 summarizes the notations for the CPD-P problem that are used throughout the paper. In our problem definition, the packages and drivers are viewed as the tasks and workers respectively. Compared to the general MTA problem [23,24], packages and drivers are associated with location attributes (i.e. origins and destinations) rather than one location of typical tasks and workers. So the CPD-P problem can be formulated as follows:

1. Dynamic pricing: Given the sets of packages and drivers, the DoI-based dynamic pricing strategy is adopted to reallocate the total budget to price each package with considering the package demand and driver supply. Let $\text{maxR}'_p$ be the renewed value of $\text{maxR}_p$. Based on the allocation of the budget, the dynamic pricing procedure satisfy the budget constraint, that is, $\sum_{i=1}^{|P|} \text{maxR}'_p = \sum_{i=1}^{|P|} \text{maxR}_p$.

2. Matching pruning: According to the renewed $\text{maxR}'_p$ of each package, a new matching set $M$ of packages and drivers is obtained with considering the $\text{maxR}$ and $\text{maxWT}$ constraint, that is, $\text{WT}(p, d) \leq \text{maxWT}_p$ and $\text{Price}(p, d) \leq \text{maxR}_p$.

3. Package assigning: The package assignment problem can be formulated as the mathematical model of MCMF [17]. Let $G = (V, E)$ be the flow network graph constructed by the obtained matching set $M$, where $V$ refers to the set of vertices, and $E$ refers to the set of edges. We create a new source
vertex $s$ and a new destination vertex $t$. There are $|P|$ edges connecting the vertex $s$ to the vertices mapped from $P$ and $|D|$ edges connecting the vertex $t$ to the vertices mapped from $D$. Therefore, the set $V'$ contains $|P| + |D| + 2$ vertices. The set $E'$ contains $|P| + |D| + |M|$ edges. Each edge $(p, d) \in E'$ has capacity $\langle p, d \rangle = 1$ (i.e. single package assignment), flow $f(p, d) \geq 0$, and weight $w(p, d) \geq 0$, where the weight of sending the flow $f(p, d)$ is $f(p, d) \times w(p, d)$. Specifically, flow $f(p, d)$ and weight $w(p, d)$ are defined as below:

\begin{equation}
    f(p, d) = \begin{cases} 
1, & \text{if } <p, d> \in M \\
0, & \text{otherwise}
\end{cases},
\end{equation}

\begin{equation}
    w(p, d) = \begin{cases} 
\text{TC}(p, d), \text{Price}(p, d) \text{ or } \frac{\text{Price}(p, d)}{\text{TC}(p, d)}, & \text{if } <p, d> \in M \\
0, & \text{otherwise}
\end{cases}.
\end{equation}

Here, the weight $w(p, d)$ taking different values represent different assignment strategies.

The optimization objective is to find the maximum flow from $s$ to $t$ in $G'$ while ensuring its total weight is minimal with satisfying the capacity constraint and conservation constraint. Note that the net quantity flowing out of source $s$ is equal to the net quantity flowing into sink $t$. Moreover, since we focus on the one-to-one matching-based package assignment problem, the maximum flow also satisfies the two constraints: 1) A package $p_i$ is only assigned to no more than one driver, that is, $\sum_{d=1}^{\lvert D \rvert} f(p_i, d) \leq 1$, and 2) a driver $d_j$ can accept one package at most, that is, $\sum_{i=1}^{\lvert P \rvert} f(p_i, d_j) \leq 1$. Due to the decision variable $f(p_i, d_j)$ limiting to integer, we can solve this by mixed integer programming.

Note that the CPD-P problem can be in theory interpreted as a knapsack problem, which is a well-studied combinatorial optimization problem with numerous applications. Imagine creating a node, whenever the hiker selects an item. Her current selection may force her to select another item from a set of alternatives, which in turn may force her to select from another set of alternative items, and so on. This can be pictured as a tree rooted at each initial item node with all arcs directed away from the root. If we connect a source node to all initial item node and each leaf node to a sink node with a directed unit capacity arc of cost zero, the resulting problem is again a MCMF problem.

## REWARD MODEL AND PRICING METHOD

The CPD platform computes the price of the package delivery service based on the distance of the package trip and the detour that the driver will make to offer the service and returns a set of drivers that can make it to the requesters within its price and temporal constraints. Since the variety of package demand and driver supply across the DoIs of different packages, to further optimize the package assignment results, the platform internally prices packages dynamically, that is, the packages with high ratio of demand and supply in their DoIs are high-priced, contrarily are low-priced.

### 3.1 Driver’s reward model

The driver’s reward model is introduced in ref. [25]. As shown in Figure 1, the origins and destinations of drivers’ delivery requests and packages’ announcements are plotted by black triangles and black circles respectively. The dotted line represents the original driver trip from her/his origin to her/his destination. The solid line represents the delivery route that the driver $d$ provides a delivery service to the package $p$’s delivery request. After $p$ being picked up, $d$ goes through the package trip till $d_{p,d}$, then returns to his own destination. As shown in Figure 1, the price cost of the delivery service including two components: 1) The price cost of the package trip $\text{PackageTrip}(p)$. 2) The price cost of the detour $\text{Detour}(p, d)$ (i.e. driver $d$ changes his initial route to pick up and drop off $p$, then returns to his own destination). The detour’s price cost plays a major role in matching drivers to packages, as the detour varies with the driver trip. Then we define the price cost in the following equation:

\begin{equation}
    \text{Price}(p, d) = \text{PackageTrip}(p) + \text{Detour}(p, d).
\end{equation}
Further formulation of the detour price cost is stated as follows:

\[
\text{Detour}(p_i,d) = \text{Pickup}(p_i,d) + \text{PackageTrip}(p_i) + \text{Return}(p_i,d) - \text{DriverTrip}(d).
\]  

According to Equations (3) and (4), we deduce the final equation to calculate the price cost for any package-driver matching.

\[
\text{Price}(p_i,d) = \text{Pickup}(p_i,d) + 2 \times \text{PackageTrip}(p_i) + \text{Return}(p_i,d) - \text{DriverTrip}(d).
\]  

It is important to note that the price cost of any trip between origins and destinations is proportional to the shortest path distance in road network \(G\). For example, PackageTrip(\(p_i\)) is proportional to the shortest path distance from \(\text{ori}_p\) to \(\text{des}_p\). Furthermore, assume the cars travel at constant speed in the road network, we can calculate the time cost of entire delivery procedure for each package by using the following formulation:

\[
\text{TC}(p_i,d) = \text{WT}(p_i,d) + \text{TripTime}(p_i,d).
\]

where \(\text{WT}(p_i,d)\) refers to the waiting time for being picked up and \(\text{TripTime}(p_i,d)\) refers to the travel time from \(\text{ori}_p\) to \(\text{des}_p\).

### 3.2 Domain-of-influence-based pricing method

The DoI is a circular region centred on the origin of each package’s delivery request, as shown in Figure 1. Given \(|P|\) packages’ delivery requests in the entire region, each package is associated with a diversified price \(p_c\) and a corresponding DoI radius \(r_i\) (\(i = 1, 2, \ldots, |P|\)).

Higher-priced packages will appeal to distant drivers, and it means that their DoIs are associated with larger radius. Similarly, the DoIs of lower-priced packages have smaller radius. For a given package \(p_i\), its DoI radius \(r_i\) and price \(p_c\) satisfy the Equation (7).

\[
r_i = \overline{s} + 7 \times \frac{p_i}{p_c} = \overline{s} \left( 1 + \frac{p_i}{p_c} \right).
\]  

Equation (7) indicates that the DoI radius \(r_i\) is positively correlated to price \(p_c\), since the average distance \(\overline{s}\) and average price \(\overline{p_c}\) are invariable with the given \(|P|\) packages’ delivery requests. Let \(S_{p_i} = \{S_{p_i,d}\}\) be a set of distances from the origin of a package \(p_i\)’s delivery request to the origins of other packages’ delivery requests (drivers’ announcements). Then average distance \(\overline{s}\) and average price \(\overline{p_c}\) are derived from Equations (8)–(10).

\[
s_i = \max(\min(S_{p_i,d}), \min(S_{p_i,d})).
\]

\[
\overline{s} = \frac{1}{|P|} \sum_{i=1}^{|P|} s_i.
\]

The distance \(s_i\) refers to the minimum DoI radius. Two possible cases are shown in Figure 2 where each case demonstrates that an optional driver exists within \(s_i\). The average distance \(\overline{s}\) denotes the average distance of all the minimum DoI radius. According to the initial price of all packages, we determine the average price \(\overline{p_c}\).

\[
\overline{p_c} = \frac{1}{|P|} \sum_{i=1}^{|P|} p_c.
\]  

The distance \(s_i\) refers to the minimum DoI radius. Two possible cases are shown in Figure 2 where each case demonstrates that an optional driver exists within \(s_i\). The average distance \(\overline{s}\) denotes the average distance of all the minimum DoI radius. According to the initial price of all packages, we determine the average price \(\overline{p_c}\).

\[
\overline{p_c} = \frac{1}{|P|} \sum_{i=1}^{|P|} p_c.
\]

The high priced of packages motivate more distant drivers to take part in competition of completing the package deliveries, which leads to a relative balance between the package demand and driver supply in the DoIs. And in contrary, lower-priced packages make more drivers quit from the competition, so that the package demand and driver supply comes to the balance. Then an important pricing parameter \(\beta\) is introduced to represent the ratio of demand and supply, that is, the number of packages accomplished by the unit driver in the DoI of each package.

\[
\beta = \frac{\rho_p}{\rho_d},
\]

where \(\rho_p\) (\(\rho_d\)) denotes the number of packages (drivers) located in the DoI of package \(p\). The ratio of demand and supply increases with the increase of \(\rho_p\) and decreases with the increase of \(\rho_d\). Each package \(p_i\) corresponds to a different \(\beta_i\). Given the set of packages in current time, we can calculate the total reward \(p_{c_{\text{sum}}}\) of all packages. Based on the allocation idea, each package \(p_i\) is priced by the proportion \(\beta_i\), which is determined by the corresponding \(\beta_i\):

\[
\beta_i = \beta \left/ \sum_{i=1}^{|P|} \beta_i \right.,
\]

\[
p_{c_i} = \beta' \cdot p_{c_{\text{sum}}} = \beta' \sum_{i=1}^{|P|} p_c.
\]

The allocation mechanism can guarantee that the higher ratios of demand and supply of the DoIs make packages high-priced. However, the allocation mechanism cannot guarantee
the basic reward for drivers to complete package delivery. For each package $p_i$, the basic reward is denoted as $\text{PackageTrip}(p_i)$. Therefore, we revise the Equation (13) as follows:

$$pc_i = \beta_i \left( pc_{sum} - \sum_{j=1}^{|P|} \text{PackageTrip}(p_j) \right) + \text{PackageTrip}(p_i)$$

$$= \beta_i \left( \sum_{j=1}^{|P|} \max R_{p_i} - \sum_{j=1}^{|P|} \text{PackageTrip}(p_j) \right) + \text{PackageTrip}(p_i)$$

$$= \beta_i \left( \sum_{j=1}^{|P|} \max R_{p_i} - \sum_{j=1}^{|P|} \text{PackageTrip}(p_j) \right) + \text{PackageTrip}(p_i).$$

The DoIs increase as packages’ prices rise. High-priced packages motivate more distant drivers to join their candidate driver sets, which achieve the balance of supply and demand in the DoIs. And in contrary, when the packages are low-priced, their DoIs decrease, and some drivers quit from the competition, which also make the demand and supply in the DoIs reach balance. The DoI-based pricing strategy needs to obtain stable price through iterative calculation. If there are prior practical data, the original price of each package can be used as the initial iteration value. Otherwise, the double price of each package trip can be used as the initial iteration value. The Example 1 shows the procedure of the iterative calculation

**Example 1.** We use the running instance in Figure 1 to calculate corresponding parameters. We use the Equations (8)–(10) to obtain the $s_1, \ldots, s_5 = \{s(p_1, d_1), s(p_2, d_1), s(p_3, p_4), s(p_4, d_1), s(p_5, p_4)\} = \{2, 3, 2, 1, 1\}$, $\bar{r} = 1.8$ and $\bar{p} = \sum_{i=1}^5 \text{max} R_{p_i} / 5 = 8$ respectively. Then the initial values of $\{r_1, \ldots, r_5\}$ are set to $\{4.05, 3.38, 3.83, 3.16, 3.60\}$ by Equation (7). Then we calculate two iterations to obtain the stable price, which is shown in Figure 3. To simplify, we represent $\text{PackageTrip}(p)$ as $PT$.

The waiting time of $p_1$ and $p_4$ for being picked up is 5.5 and 2, respectively. Therefore, $p_1$ has a higher reward 1.14 than $p_4$ with 0.91.

### 4 DESCRIPTION OF THE FRAMEWORK DOIPRICE

Our proposed framework DoIPrice consists of pricing, pruning and assignment stages. Its inputs include offline road network data and online requests (i.e. submitted package delivery request set $P$ and driver announcement set $D$). The outputs are the matches between $P$ and $D$, as shown in Figure 4. The details about our proposed three-stage framework are presented as follows.

#### 4.1 Stage one: MaxPrice pricing

The initial maximum reward of each package is usually set according to its shortest path distance, which ignores the influence of supply and demand on price. To model the relation between the driver supply and package demand, we propose the DoI-based strategy and then use it to reprice the maximum reward for each package.

The initial pricing is used to determine the radius of the DoI. When the DoI of each package is obtained, the pricing parameters are calculated by the package density and driver density, and then dynamic pricing allocation is adopted to price all packages. Repeat above steps until the price of each package converges to
4.2 Stage two: Matching pruning

Online package delivery problem consists of a number of rounds and each round has a fixed time interval. We focus on solving the fast matching between the large scale packages and drivers in a round, which takes account of the price cost and time cost simultaneously. In order to reduce the time-consuming, a pruning algorithm proposed in the work [26] is adopted to calculate matching results with two procedures. The maximum reward of each package is updated from MaxPrice pricing stage before matching stage begins.

4.2.1 Pre-pruning procedure

Obviously, shortest paths on the road network are the ideal way to determine whether a driver can satisfy the constraints of packages’ delivery requests, but it is not suitable because the computation cost is so expensive. So we have a pre-pruning on the drivers to avoid a prohibitive cost.

Algorithm 2 gives the pseudo code of the pre-pruning process. First, we remove the drivers in the no-parking roads (e.g. viaducts) from the matching table (Lines 1–5). Lines 7–18 show the process of pruning drivers who cannot pick up the packages.

4.2.2 Skyline pruning procedure

The input of this procedure is a set of candidate drivers produced from the pre-pruning procedure, we need to adjust the nodes of drivers on the one-way road and calculate the actual road network distances of Pickup and Return trips, then return the final skyline drivers satisfying the time and price constraints. Before the shortest path calculation, we have to make adjustments to the drivers on one-way roads. In general, the shortest path is started from the closest node to the driver, and it is exactly when the driver on two-way road, because the driver can move to any node in this edge. But the case is different when age within its maximum waiting time and maximum reward, based on Euclidean distance computations. The Euclidean distance between any two nodes is usually less than the actual road network distance of the same two nodes. Meanwhile, the cost of calculating the Euclidean distance is much less than the actual shortest path calculation, so we can early prune a lot of drivers without any actual road network calculation. As line 14 shows, we compute Price(p, d) by using Euclidean distance, which is named EuclideanPrice(p, d). Obviously, we can conclude that: EuclideanPickup(p, d) < Pickup(p, d), EuclideanReturn(p, d) < Return(p, d), so we further make the conclusion that EuclideanPrice(p, d) < Price(p, d). If EuclideanPrice(p, d) > Price(p, d), we remove the driver d from the matching table without any further computation as well.
the driver travels on one-way roads, because the driver cannot turn around and he must move forward until the next intersection, so the shortest path calculation must be started from the next node rather than the closest node in the edge. Similarly, if the destination locates on the one-way road, we have to choose that node. So we should make adjustments in order to achieve a correct result in shortest path when considering one-way roads.

Algorithm 3 Skyline pruning

Input: A set of drivers generated from pre-pruning procedure; a package \( p \), the road network data;

Output: A skyline driver set \( D' \) matching with \( p \);

1: compute \( \text{Pickup}(p, d) \) for each driver \( d \) in matching table;
2: sort drivers in matching table by values of (EuclideanReturn \((p, d)\)−DriverTrip\((d)\)) in ascending order;
3: Initialize variable \( \text{MAX} \) with \( \min(R) \); \( \text{MAX} \) is used to improve the efficiency of the pruning procedure, which is updated by the following iterative loops.
4: While matching table \( \not= \emptyset \) do
5: chose the nearest driver to pick up the package;
6: if \( \text{Pickup}(p, d) > \max\text{WT}_{p} \)
7: break;
8: end if
9: if \( \text{Pickup}(p, d)+2 \times \text{PackageTrip}(p) + \text{EuclideanReturn}(p, d)\)−DriverTrip\((d)\) > \( \text{MAX} \)
10: Remove \( d \) and other drivers that below \( d \) from matching table;
11: else
12: compute the shortest path Return\((p, d)\);
13: if \( \text{Pickup}(p, d)+2 \times \text{PackageTrip}(p)+ \text{Return}(p, d)\)−DriverTrip\((d)\) > \( \text{MAX} \)
14: remove \( d \) from matching table;
15: else
16: \( \text{MAX} \leftarrow \text{Pickup}(p, d)+2 \times \text{PackageTrip}(p)+ \text{Return}(p, d)\)−DriverTrip\((d)\);
17: move \( d \) from matching table to result set \( D' \);
18: end if
19: end if
20: return the skyline driver set \( D' \);

the driver travels on one-way roads, because the driver cannot turn around and he must move forward until the next intersection, so the shortest path calculation must be started from the next node rather than the closest node in the edge. Similarly, if the destination locates on the one-way road, we have to choose that node. So we should make adjustments in order to achieve a correct result in shortest path when considering one-way roads.

Algorithm 3 shows the skyline pruning procedure. On line 1, we calculate the shortest path distance of Pickup procedure instead of the Euclidean distance for each driver in matching table. Although the cost of the road network calculation is expensive, we think this expensive cost is necessary. Suppose there is a package and two drivers, driver A and driver B. Driver A and the package are in the same two-way road, and driver B is in another one-way road. The Euclidean distance between driver B and the package is less than that between driver A and the package. Since driver B travels on the one-way road and he must reach the next intersection, so the road network distance may be too long to satisfy the constraints of package, then the matching algorithm would be stopped, but in fact, driver A is closer to the package while satisfying maximum waiting time constraint. In this way, we can guarantee suitable match-

Algorithm 4 Package assigning

Input: The matching set \( M \) obtained from stage-two; the package set \( P \), the skyline driver set \( D' \);

Output: A stable matching set;

1: Number all the packages and drivers in sequence \( \{1, 2, \ldots, |P| + |D|\} \);
2: Create capacity and cost array;
3: For each \( p, d > 1 \) in \( M \);
4: capacity\((d)\) \( \leftarrow \) \( 1 \);
5: assign cost\((d)\) with weight; //According to the optimization objectives, the weight can be assigned as TC\((\bullet)\), Price\((\bullet)\) and unit price (i.e. Price\((\bullet)/TC\((\bullet))\), respectively.
6: end for
7: obtain \( A \) by using the algorithm for solving MCMF problem; //Maximize the number of assignments (MA) and minimize the total time cost (MT), price cost (MP) or unit price (MR);
8: for each \( a \) in \( A \);
9: Make a judgment that \( a \) is stable or not (16);
10: put \( a \) in the stable matching set if \( a \) is stable;
11: end for
12: return the stable matching set;

Stage three: Task assignment

So far, we obtain skyline drivers for each package’s delivery request. All the package-driver matches are used to construct a weighted bipartite graph. The solution of the CPD-P problem is a stable matching set.

4.3.1 Assignment scheme

The package assigning procedure is shown as the pseudo code in Algorithm 4. From line 1 to line 5, we initialize the capacity and cost array on basis of the matching results in M. Then we adopt the algorithm for solving the MCMF problem to obtain maximum assignments (MA) with minimum cost (MC). With considering the driver’s acceptance, we use the maximum assignments to obtain the stable matching set. Meanwhile, the cost array is assigned with time cost, price cost or unit price respectively, such that we further obtain the minimum time cost (MT), minimum price cost (MP) or minimum unit price (MR) from
the corresponding maximum stable matching set (Lines 7–12). To figure out the stable matches, we employ the hyperbolic temporal discounting function to estimate each driver’s perceived reward for the payment of a package, which characterizes the willingness to accept the task.

In implementation, we build two matrixes, that is, capacity matrix and weight matrix. The capacity matrix is a \((0, 1)\)-matrix, where 1 means matched and 0 means none. Through the matching pruning process, the matching between a package and a driver associates with two weights, that is, time cost and price cost. Thus we can define three weight matrixes by using time cost (MAMT), price cost (MAMP) or unit price (MAMR), respectively. Each weight matrix corresponds to an assignment strategy and the solution is obtained in polynomial time.

4.3.2 Assignment with optimal pricing

The DoI-based pricing strategy is essentially a heuristic strategy. To obtain the true system optimal pricing strategy, we determine the optimal price for each package by the way of searching optimal package assignments, which consists of following steps:

1. Find the reachable drivers for each package. The reachable driver subset for a package should satisfy the following maximum waiting time constraint.

2. Find package-driver matchings without considering the maximum price constraint. We assume that there is no limit on the maximum price of each package. According to the driver’s reward model, all the possible package-driver matchings can be obtained after matching pruning stage, which are associated with costs of TC\((p, d)\) and Price\((p, d)\).

3. Apply the package assigning algorithm to find the optimal package assignments with maximum-flow and minimum cost. In particular, we adopt three assignment strategies (i.e. MAMP, MAMT and MAMR) to obtain three sets of package assignments. For each package-driver pair \(\langle p, d \rangle\) in the optimal package assignment set, the optimal price of package \(p\) is obtained as Price\((p, d)\). Other packages that are not belonging to the optimal package assignment set, are average pricing with the rest of the budget.

4.3.3 Acceptance behaviour model

The driver’s choice to accept or reject the assigned package depends on whether the package’s price is higher than his perceived reward or not. The perceived reward follows the temporal discounting theory, that is, the greater is the delay to a future reward, the less is its present, subjective value. For the present purpose we define perceived reward as the package’s maximum reward discounted by delay time,

\[
V'(p_{\bar{c}}) = p_{\bar{c}}/(1 + k_{p} \times TC(p, d)).
\]  

where \(k_{p} \in [0, 1]\) represents the discounting parameter. Note that the perceived reward is inversely proportional to the delay time to receive the reward. It means that the longer delay TC\((p, d)\) make smaller perceived reward \(V'(p_{\bar{c}})\). We assume \(k_{p}\) meets the Gaussian distribution. Thus we further define the driver’s acceptance behaviour as follows:

\[
y = \begin{cases} 
1, & V'(p_{\bar{c}}) \leq \text{Price}(p, d) \\
0, & \text{otherwise}
\end{cases}
\]  

5 EVALUATION

5.1 Experiment setup

1. Road network data

The road network data for experiments in this section is produced by the MNTG [18]. We generated the road network of Ji Nan, Shan Dong province of China, containing 9131 edges and 8331 nodes.

2. Request simulation

The drivers are generated by Brinkoff road network generator from the MNTG and the number of drivers ranges from 2000 to 5000. And we generate about 2000 package delivery requests through the random function on the roads. And we set the maxWT\(_{p}\) constraint as 5 min and the initial maxR\(_{p}\) constraint as double price cost of the package trip. Here, we make two assumptions: 1) the average travel speed of each driver is fixed at 40 km h\(^{-1}\), regardless of the traffic conditions. 2) The ridesharing cost is 1 RMB per km.

3. Evaluation environment

All the algorithms are implemented in Java and the experiments are performed on a server with Intel(R) Xeon®E5-2620 2.40 GHz CPU and 8 GB RAM.

4. Compared algorithms

In order to verify the validity of our proposed approach, it is necessary to compare it with the previously proposed algorithms.

(i) FixedPrice. The price of each package remains unchanged without considering the demand-supply ration.

(ii) GridPrice. It is a dynamic pricing algorithm used in some spatial crowdsourcing companies such as Uber [37] that adopts the demand-supply ration in a grid to obtain a pricing parameter. We empirically optimize the pricing parameter on our datasets. As stated in the work [14], workers in spatial crowdsourcing can only serve a portion of spatial tasks, since some tasks require a traveling distance beyond the capability of workers. Consequently, the unified market in traditional crowdsourcing tends to fragment into multiple local markets (denoted by the grids) in spatial crowdsourcing. In detail, the entire road network space is partitioned into a number of grids with the same size. Then the imbalanced demand and supply across local markets can be modelled by the packages and drivers located in the grids.
5.2 | Experiment results

Comparison results of the evaluation metrics are derived from the number of packages, the delivery time and the prices of different packages, which are shown in Tables 2–5. In order to simplify the representation, NA and NS denote the number of maximum assignments and the number of stable matches respectively. For the stable matches, SoT and SoP refer to the sum of their delivery time and price cost.

1. Results of assignment success rate: In Table 2, we compare two pricing approaches against our DoIPrice framework, with the same 2000 package delivery requests, and the number of drivers is from 500 to 5000. We can see that the DoIPrice achieves best result of SAR under all assignment strategies. GridPrice performs worst when \(|D| > 3000\) and it indicates that more packages are rejected by the drivers because all packages are evenly priced after Grid-based pricing. For the FixedPrice and DoIPrice, MAMT performs better with the number of drivers increasing. Though both of FixedPrice and DoIPrice have the similar performance on SAR with the number of drivers reaching 5000, the DoIPrice significantly has increased the number of stable matches, which improves the SR accordingly. The reason for this result is that large number of drivers provide more opportunities for package delivery to choose a driver with satisfying the package’s maximum reward constraint. Moreover, DoIPrice can use the budget to re-price the packages based on package demand and driver supply, that is, the package is high priced on the basis that package demand exceeds driver supply. Compared to GridPrice, DoIPrice has improved the SAR by 11–16% and SR by 3% with \(|D| = 500\), 7–16% and 8–22% with \(|D| \geq 4000\), which proves that our approach is more practical to meet the diverse demand-supply conditions. Since the task assignment in local market is usually affected by the real traffic condition, for example, traffic jam, DoIPrice can improve it by scheduling drivers in multiple markets.

2. Results of economic utility: Figures 5 and 6 show the comparison results of economic utilities of compared algorithms. In general, when the number of drivers grows, the economic utility gets higher. As expected, for the same 2000 package delivery requests, the economic utility of DoIPrice is higher than FixedPrice. In particular, the economic utility rises 5–10% in DoIPrice-MAMP, 9–12% in DoIPrice-MAMT and 6–10% in DoIPrice-MAMR. When \(|D| \leq 1000\), compared to FixedPrice, the growth is improved by 30–70% as shown in Figure 5a. Figure 5b shows that DoIPrice has improved the economic utility of GridPrice by over 30% when \(|D| = 500\) and \(|D| \geq 4000\) and it proves that DoIPrice is more practical for dealing with the dynamic pricing when the packages and drivers are imbalance. Note that GridPrice performs worst with \(|D| \geq 4000\), while DoIPrice has stable performance. The reason is that DoIPrice avoids centralized distribution of drivers in multiple regions by increasing the package’s price cost. Moreover, according to temporal

GridPrice sets the price for a given grid \(g\) as follows:

\[
\beta_{pg} = \frac{\rho_g}{\rho_d}. \tag{17}
\]

\[
\beta'_{pg} = \frac{\sum_{i=1}^{N} \beta_{pg}}{N}. \tag{18}
\]

\[
p_{pcpg} = \frac{\rho_g \left( \sum_{j=1}^{\max p_j} R_{pi} - \sum_{j=1}^{p_{x,dg}} p_{c,dg} \right)}{\rho_d \left( \sum_{j=1}^{N} \frac{\rho_g}{\rho_{dg}} \right)} + p_{c,dg}. \tag{19}
\]

where \(\rho_g\) (\(\rho_d\)) denotes the number of packages (drivers) located in the grid \(g\) and \(N\) refers to the total number of grids. In this paper, we partition the entire region into 50 \(\times\) 50 grids. This is more practical for dealing with the dynamic pricing when the packages and drivers are imbalance. Note that GridPrice performs worst when \(|D| > 3000\) and it indicates that more packages are rejected by the drivers because all packages are evenly priced after Grid-based pricing. For the FixedPrice and DoIPrice, MAMT performs better with the number of drivers increasing. Though both of FixedPrice and DoIPrice have the similar performance on SAR with the number of drivers reaching 5000, the DoIPrice significantly has increased the number of stable matches, which improves the SR accordingly. The reason for this result is that large number of drivers provide more opportunities for package delivery to choose a driver with satisfying the package’s maximum reward constraint. Moreover, DoIPrice can use the budget to re-price the packages based on package demand and driver supply, that is, the package is high priced on the basis that package demand exceeds driver supply. Compared to GridPrice, DoIPrice has improved the SAR by 11–16% and SR by 3% with \(|D| = 500\), 7–16% and 8–22% with \(|D| \geq 4000\), which proves that our approach is more practical to meet the diverse demand-supply conditions. Since the task assignment in local market is usually affected by the real traffic condition, for example, traffic jam, DoIPrice can improve it by scheduling drivers in multiple markets.

- Stable assignment rate (SAR) is defined as the ratio of the stable matches to the maximum assignments. Meanwhile, the stability rate (SR) derives from the ratio of the stable matches to the total packages’ delivery requests submitted to the server.

- Economic utility (UC) is defined as the ratio of the SAR against the sum of the price cost for all packages. When enterprises invest less in packages and get higher reward, the higher the economic utility is, and the better the pricing strategy is. The economic utility is expressed as follows:

\[
U = \frac{\text{SAR}}{\sum_{i=1}^{\max p_i} p_{yij}}. \tag{20}
\]

- Average delivery time (ADT) is defined as the ratio of the sum of the delivery time cost for all packages against the number of stable matches. The formulation is expressed as follows:

\[
\text{ADT} = \frac{\sum_{i=1}^{\max p_i} \sum_{j=1}^{\max d_j} \text{TC}(p_i, d_j) \times y_{ij}}{\sum_{i=1}^{\max p_i} \sum_{j=1}^{\max d_j} y_{ij}}. \tag{21}
\]

- Average delivery price (ADP) is defined as the ratio of the sum of the price cost for all packages against the number of stable matches. The formulation is expressed as follows:

\[
\text{ADP} = \frac{\sum_{i=1}^{\max p_i} \sum_{j=1}^{\max d_j} \text{Price}(p_i, d_j) \times y_{ij}}{\sum_{i=1}^{\max p_i} \sum_{j=1}^{\max d_j} y_{ij}}. \tag{22}
\]

- CPU time indicates the CPU execution time of an algorithm. Its unit is second.
TABLE 2  The comparison results on SAR and SR of different pricing algorithms based on different assignment strategies

| | FixedPrice | GridPrice | DoIPrice |
|---|---|---|---|
| | MAMP | MAMT | MAMR | MAMP | MAMT | MAMR | MAMP | MAMT | MAMR |
| $|P| = 2000$  
$|D| = 500$  
NA | 314 | 314 | 314 | 334 | 334 | 334 | 356 | 356 | 356 |
NS | 188 | 232 | 173 | 253 | 252 | 240 | 320 | 324 | 296 |
SAR | 0.60 | 0.74 | 0.55 | 0.76 | 0.75 | 0.72 | 0.90 | 0.91 | 0.83 |
SR | 0.09 | 0.12 | 0.09 | 0.13 | 0.13 | 0.12 | 0.16 | 0.16 | 0.15 |
| $|D| = 1000$  
NA | 585 | 585 | 585 | 610 | 610 | 610 | 542 | 542 | 542 |
NS | 382 | 427 | 346 | 474 | 471 | 455 | 485 | 423 | 457 |
SAR | 0.65 | 0.73 | 0.59 | 0.78 | 0.77 | 0.75 | 0.89 | 0.78 | 0.84 |
SR | 0.19 | 0.21 | 0.17 | 0.24 | 0.24 | 0.23 | 0.24 | 0.21 | 0.23 |
| $|D| = 2000$  
NA | 967 | 967 | 967 | 1036 | 1036 | 1036 | 982 | 982 | 982 |
NS | 572 | 597 | 558 | 627 | 617 | 594 | 601 | 669 | 594 |
SAR | 0.59 | 0.62 | 0.58 | 0.61 | 0.60 | 0.57 | 0.61 | 0.68 | 0.60 |
SR | 0.29 | 0.30 | 0.28 | 0.31 | 0.31 | 0.30 | 0.30 | 0.33 | 0.30 |
| $|D| = 3000$  
NA | 1116 | 1116 | 1116 | 1283 | 1283 | 1283 | 1136 | 1136 | 1136 |
NS | 685 | 734 | 667 | 762 | 750 | 728 | 733 | 827 | 717 |
SAR | 0.61 | 0.66 | 0.60 | 0.59 | 0.58 | 0.57 | 0.65 | 0.73 | 0.63 |
SR | 0.34 | 0.37 | 0.33 | 0.38 | 0.38 | 0.36 | 0.37 | 0.41 | 0.36 |
| $|D| = 4000$  
NA | 1341 | 1341 | 1341 | 1287 | 1287 | 1287 | 1388 | 1388 | 1388 |
NS | 811 | 862 | 795 | 708 | 694 | 675 | 862 | 975 | 855 |
SAR | 0.60 | 0.64 | 0.59 | 0.55 | 0.54 | 0.52 | 0.62 | 0.70 | 0.62 |
SR | 0.41 | 0.43 | 0.40 | 0.35 | 0.35 | 0.34 | 0.43 | 0.49 | 0.43 |
| $|D| = 5000$  
NA | 1617 | 1617 | 1617 | 1481 | 1481 | 1481 | 1747 | 1747 | 1747 |
NS | 1273 | 1314 | 1261 | 1084 | 987 | 955 | 1403 | 1441 | 1393 |
SAR | 0.79 | 0.81 | 0.78 | 0.73 | 0.67 | 0.64 | 0.80 | 0.82 | 0.80 |
SR | 0.64 | 0.66 | 0.63 | 0.54 | 0.49 | 0.48 | 0.70 | 0.72 | 0.70 |

discounting theory, less delivery time produces greater immediate value, which attracts drivers to join the delivery process. This is the reason why DoIPrice-MAMT has best performance on improving economic utility.

3. Results of delivery time: Table 3 shows the comparison results of ADT of different pricing algorithms. DoIPrice achieves best results of ADT with varying $|D|$. MAMR performs worst with ADT reaching about 5.38–11.64, followed by MAMP and MAMT with 3.8–10.69 and 3.85–10.65, respectively. It is clear that MAMT assignment strategy can reduce the ADT significantly. For a package, the transportation time is fixed, so the delivery time depends on the waiting time for being picked up. Our DoIPrice makes more nearby drivers satisfying the package’s maximum reward constraint. This indicates that DoIPrice can be well applicable to task assignment in a real life if the time-dependent of travel time is modelled.

When $|D| > 2000$, that is, the driver supply is sufficient, GridPrice performs worst. This is because the unlimited supply leads to the similar package-driver ratio in each grid so that packages’ prices are similar, which reduces the number of stable matches.

4. Results of delivery price: As the results shown in Table 4, we can see that the ADP decreases significantly in DoIPrice. Normally, the MAMP has better performance on ADP than the MAMT under the condition of maximum assignments, which is inconsistent with our experiment result. This can be explained by two reasons: 1) The driver will make a choice to accept the package or not, so the number of stable matches is less than the number of maximum assignments. The ADP focus on the stable matching. 2) DoIPrice improves the number of stable matches rather than the number of maximum assignments.

5. Results of CPU time: In order to evaluate the efficiency of all algorithms, we compared the CPU time spent in pricing strategies, and the results are shown in Figure 7. We can see that compared algorithms have the similar performance on the CPU time because they construct similar size of flow network graphs to achieve the final solutions. Because of the iterative calculation in DoI-based pricing, DoIPrice takes a solution time, but it is not much different from other algorithms, and its advantage in solving effect can make up for this disadvantage.
TABLE 3  The comparison results on ADT of different pricing algorithms based on different assignment strategies

| | FixedPrice | GridPrice | DoIPrice |
|---|---|---|---|
| | MAMP | MAMT | MAMR | MAMP | MAMT | MAMR | MAMP | MAMT | MAMR |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
TABLE 5  The comparison results of OptPrice and DoIPrice based on different assignment strategies

| $|P| = 100$ | OptPrice | DoIPrice |
|-----------|-----------|-----------|
|           | MAMP      | MAMT      | MAMR      | MAMP      | MAMT      | MAMR      |
| $|D| = 50$   | UC        | 0.03      | 0.03      | 0.02      | 0.02      | 0.03      |
|           | SAR       | 0.68      | 0.68      | 0.55      | 0.79      | 0.83      | 0.71      |
|           | SR        | 0.21      | 0.21      | 0.17      | 0.19      | 0.20      | 0.17      |
|           | ADP       | 10.24     | 10.46     | 13.85     | 5.83      | 5.79      | 6.62      |
|           | ADT       | 12.29     | 12.22     | 17.04     | 8.89      | 8.30      | 10.19     |
| $|D| = 100$  | UC        | 0.05      | 0.05      | 0.05      | 0.04      | 0.04      | 0.04      |
|           | SAR       | 0.67      | 0.67      | 0.65      | 0.79      | 0.81      | 0.77      |
|           | SR        | 0.40      | 0.40      | 0.39      | 0.34      | 0.35      | 0.33      |
|           | ADP       | 11.56     | 11.66     | 12.58     | 6.22      | 6.20      | 6.58      |
|           | ADT       | 13.06     | 13.03     | 14.38     | 8.69      | 8.21      | 9.29      |
| $|D| = 150$  | UC        | 0.06      | 0.07      | 0.06      | 0.05      | 0.05      | 0.05      |
|           | SAR       | 0.65      | 0.69      | 0.63      | 0.75      | 0.77      | 0.75      |
|           | SR        | 0.51      | 0.54      | 0.49      | 0.39      | 0.40      | 0.39      |
|           | ADP       | 13.05     | 12.55     | 14.63     | 6.50      | 6.58      | 6.57      |
|           | ADT       | 13.97     | 13.03     | 15.67     | 8.79      | 8.21      | 8.90      |
| $|D| = 200$  | UC        | 0.06      | 0.06      | 0.06      | 0.04      | 0.05      | 0.05      |
|           | SAR       | 0.63      | 0.62      | 0.65      | 0.67      | 0.69      | 0.69      |
|           | SR        | 0.50      | 0.49      | 0.51      | 0.37      | 0.38      | 0.38      |
|           | ADP       | 12.70     | 13.33     | 13.51     | 7.26      | 7.41      | 7.25      |
|           | ADT       | 13.77     | 13.66     | 15.14     | 10.22     | 9.28      | 10.42     |

5.3  Case study on exact solution

Since the designed optimal pricing strategy cannot obtain the exact optimal solution in polynomial time, we attempt to use the real data to further find the true exact optimal solution with the aim of comparing with the DoIPrice. In this study, we work on a set of real CPD instances (i.e. $|P| = 100$ and $50 \leq |D| \leq 200$) are provided by a delivery company running its business with an intra-city O2O (online to offline) delivery platform. The dataset collects all the demand of requesters within a week. We use these instances to perform offline experiments, in which we assume the average speed of for each driver is 36 km per hour, and the travel cost is set as 1 dollar per KM.
For each dynamic requester demand, we place package delivery task to the third-party crowdsourcing delivery platform by using both pricing strategies. For simplicity, we represent the optimal pricing approach as OptPrice. The corresponding experiment results on different metrics are shown as Table 5 and Figure 8. As can be seen from the comparison results, OptPrice performs better than DoIPrice on SR and UC, while DoIPrice has better performance on SAR, ADT, ADP and CPU time. The AoptPrice has the similar performance as DoIPrice when $|D| = 50$ (i.e. the supply is limited) and $|D| = 200$ (i.e. the supply is sufficient), which indicates that DoIPrice is practical to optimize the delivery cost with considering the imbalanced demand and supply.

6 | RELATED WORK

Crowdsourced package delivery: The CPD problem aims to transport packages from origins to destinations at minimum costs. There exist two proposed solutions that recruit the workers overlapped in the space and time to deliver the packages [27,28]. However, the number of suitable workers is limited, resulting in longer and uncontrollable package delivery delay. To improve the efficiency of package delivery, some papers combine the passenger flow and package flow to share transport, which is formulated as the share-a-ride problem [4,7,29,30]. Specifically, they insert the package requests into the rider trips and discover the optimal package delivery paths to minimize the transportation cost. Due to the insertions, taxis may stop multiple times and make longer detour, which degrades the service quality to passengers. Moreover, the long-distance package may fail in delivering, since the rider trips are mainly short-distance.

In recent years, some studies have developed relay network traffic flow models to minimize the impact on passenger service and maximize the success rate of package delivery [8,9,31–33]. However, they still suffer from some limitations: 1) They assume drivers will accept the assigned delivery tasks. 2) Package delivery tasks are often associated with the fixed unit price. 3) Most focus on finding the optimal path that minimizes the package’s delivery time and maximizes the total number of assigned packages, but neglect the uncertainty of drivers.

To the best of our knowledge, no previous work utilizes the pricing strategy to improve both of stable matching and delivery cost in CPD, which motivates this study. Compared to the proposed solutions with no pricing or fixed price, in this paper, we introduce dynamic pricing to the package assigning process.

Dynamic pricing in SC: Some pioneer works have considered the effect of prices in spatial crowdsourcing [34,35]. In ref. [34], prices are obtained based on the profiles of workers and requesters, which are used as inputs to find matches between workers and requesters to maximize the revenue. In ref. [35], an incentive mechanism in spatial crowdsourcing is proposed.
Since it is designed for crowdsourcing, a special case of crowdsourcing, where a task is to collect data at a location, so it is not applicable for CPD problem. Dynamic pricing has been recently introduced in some spatial crowdsourcing applications [36–39] and they only assume a single market with number of supply fixed. In ref. [14], the proposed solution sets prices by jointly optimizing the drivers in multiple dependent grids to contribute more revenue. Our work is also related to dynamic pricing based on the supply and the demand. However, it differs from our work in three-fold: 1) [14] is designed for dynamic pricing in fixed grids and tasks have the same price in the same grid. DoIs vary dynamically with the imbalanced supply and demand. Thus DoI-based pricing is capable of implementing personalized pricing. 2) The objective of [14] is to maximize the expected total revenue, while we aim to maximize the number of stable matches and minimize the delivery cost. 3) The work [14] simply assume the worker accept the assigned task with certainty, while we model the driver's perceived reward to estimate whether to accept the assigned package or not. Therefore, it is a crucial step to assign the perceived package to the driver, since the number of stable matches is affected by the drivers’ choices to accept or not.

7 CONCLUSION AND FURTHER WORK

In this paper, we propose the DoI based dynamic pricing algorithm (DoIPrice) for CPD problem to optimize the stable matching and delivery cost. In order to achieve high effectiveness and efficiency, we addressed a few challenges by designing a three-stage framework to obtain the stable matches between packages and drivers with taking account the driver's reward model. We adopt the temporal discounting function to model the relationship between the driver's reward and his/her completion time to estimate the unknown acceptance probabilities. To the best of our knowledge, it is the first work in CPD that utilizes the dynamic pricing strategy to maximize the stable matching and minimize the delivery cost. The performance of the developed algorithm is extensively tested on Jinan data set and real delivery data.

In the future, we plan to broaden and deepen this work in two directions. First, we attempt at developing more advanced pricing strategy to model the package demand and driver supply. Second, the outcome of online package assignment is stochastic, and we explore novel approaches to estimate the price cost dynamically.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (NO. 61672122, NO. 61902050, NO. 61906027), the Fundamental Research Funds for the Central Universities (NO. 3132019355), the Next-Generation Internet Innovation Project of CERNET (No. NGII20181205), and the China Postdoctoral Science Foundation Funded Project (No: 2019M661080).

REFERENCES

1. Carbone, V., Rouquet, A., Roussat, C.: The rise of crowd logistics: A new way to co-create logistics value. Journal of Business Logistics 38(4), 238–252 (2017)
2. Andweilzahmar, D., Alexander, G.N., Qing, H.: Crowdsourcing the last mile delivery of online orders by exploiting the social networks of retail store customers. Transportation Research Part E Logistics and Transportation Review 105, 105–122 (2017)
3. Rai, H., Verlinde, S., Merckx, J.: Crowd logistics: An opportunity for more sustainable urban freight transport. European Transport Research Review 9(3), 1–39 (2017)
4. Chen, W., Mes, M., Schutten, M.: Multi-hop driver-parcel matching problem with time windows. Flexible Services and Manufacturing Journal 30(3), 517–553 (2018)
5. Ghiyas, V., Demir, E., Van Woensel, T.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. Computers and Operations Research 72, 12–30 (2016)
6. Rais, A., Abeloos, F., Carvalho, M.S.: New mixed integer-programming model for the pickup-and-delivery problem with transshipment. Eur. J. Oper. Res. 235(5), 530–539 (2014)
7. Arslan, A., et al.: Crowdsourced delivery: A dynamic pickup and delivery problem with ad hoc drivers. Transp. Sci. 53(1), 222–235 (2019)
8. Chen, C., et al.: TaxiExp: A novel framework for city-wide package express shipping via taxi crowd sourcing. 2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Intl Conf on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops, Bali, Indonesia, pp. 244–251 (2014)
9. Chen, C., et al.: TripImputor: Real-time imputing taxi trip purpose leveraging multi-sourced urban data. IEEE Trans. Intell. Transp. Syst. 19(10), 3292–3304 (2018)
10. Hamed, S., Vincent, W.S.: Profit maximization in mobile crowdsourcing: A truthful auction mechanism. 2015 IEEE International Conference on Communications (ICC), London, UK, pp. 3216–3221 (2015)
11. Yang, D., et al.: Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing. Proceedings of the 18th annual international conference on Mobile computing and networking, Istanbul, Turkey, pp.173–184 (2012)
12. Tong, Y., et al.: SLADE: A smart large-scale task decomposer in crowdsourcing. IEEE Trans. Knowl. Data Eng. 30(8), 1588–1601 (2018)
13. Xia, J., et al.: Profit-driven task assignment in spatial crowdsourcing. Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, pp.1914–1920 (2019)
14. Tong, Y., et al.: Dynamic pricing in spatial crowdsourcing: A matching-based approach. Proceedings of the 37th ACM SIGMOD International Conference on Management of Data (SIGMOD 2018), Houston, TX, USA, pp. 773–788 (2018)
15. Liu, Y., et al.: FoodDNet: Toward an optimized food delivery network based on spatial crowdsourcing. IEEE Trans. Mob. Comput. 18(6), 1288–1301 (2019)
16. Voccia, S.A., et al.: The same-day delivery problem for online purchases. Transp. Sci. 53(1), 38–77 (2017)
17. Kazemi, I., Shahabi, C.: GeoCrowd: Enabling query answering with spatial crowdsourcing. Proceedings of the 20th International Conference on Advances in Geographic Information Systems. ACM, Redondo Beach, California, USA, pp. 189–198 (2012)
18. Mokbel, M., et al.: A demonstration of MNTG - A web-based road network traffic generator. Proceedings of 2014 IEEE 30th International Conference on Data Engineering (ICDE), Chicago, II, USA, pp. 1246–1249 (2014)
19. To, H., Shahabi, C., Kazemi, L.: A server-assigned spatial crowdsourcing framework. ACM Transactions on Spatial Algorithms and Systems 1(1), 1–28 (2015)
20. Huang, Z., et al.: Online vertex-weighted bipartite matching: Beating 1-1/e with random arrivals. ICALP 79, 1–14 (2018)
21. Wang, Y., et al.: Adaptive dynamic bipartite graph matching: A reinforcement learning approach. Proceedings of 35th IEEE International Conference on Data Engineering (ICDE), Macao, China, pp. 1478–1489 (2018)

22. Murphy, J.G., Vuchinich, R.E., Simpson, C.A.: Delayed reward and cost discounting. The Psychological Record 51(4), 571–577 (2001)

23. Chen, R., et al.: Fusion of multi-RSMOTE with fuzzy integral to classify bug reports with an imbalanced severity distribution. IEEE Trans. Fuzzy Syst. 27(12), 2406–2420 (2019)

24. Cheng, P., Jian, X., Chen, L.: An experimental evaluation of task assignment in spatial crowdsourcing. Proceedings of the VLDB Endowment 11(11), 1428–1440 (2018)

25. Cao, B., et al.: SHAREK: A scalable dynamic ride sharing system. Proceedings of 16th IEEE International Conference on Mobile Data Management, Pittsburgh, PA, USA, pp. 4–13 (2015)

26. Ning, J., Chen, R., Zhou, Z.: Improving large scale real-time ridesharing with heuristics for road networks. Proceedings of 2018 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Shenzhen, China, pp. 513–521 (2018)

27. McInerney, J., Rogers, A., Jennings, N.: Crowd-sourcing physical package delivery using the existing routine mobility of a local population. Proceedings of 3rd International Conference on the Analysis of Mobile Phone Datasets With a special day on the Data for Development (D4D) challenge, Boston, USA, pp. 447–456 (2014)

28. Sadilek, A., Krumm, J., Horvitz, E.: Crowdphysics: Planned and opportunistic crowdsourcing for physical tasks. Proceedings of 7th International AAAI Conference on Weblogs and Social Media (ICWSM), Boston, USA, pp. 536–545 (2013)

29. Xu, Y., et al.: An efficient insertion operator in dynamic ridesharing services. Proceedings of the 35th International Conference on Data Engineering (ICDE), Macau, China, pp. 1022–1033 (2019)

30. Wang, J., et al.: Multiobjective vehicle routing problems with simultaneous delivery and pickup and time windows: Formulation, instances, and algorithms. IEEE Transactions on Cybernetics 46(3), 582–594 (2015)

31. Kafle, N., Zou, B., Lin, J.: Design and modeling of a crowdsourced enabled system for urban parcel relay and delivery. Transp. Res. B, Methodol. 99, 62–82 (2017)

32. Chen, C., et al.: Using taxis to collect citywide e-commerce reverse flows: A crowdsourcing solution. Int. J. Prod. Res. 55(7), 1833–1844 (2017)

33. Chen, C., et al.: CrowdDeliver: Planning city-wide package delivery paths leveraging the crowds of taxis. IEEE Trans. Intell. Transp. Syst. 18(6), 1478–1496 (2017)

34. Mohammad, A., et al.: Price-aware real time ridesharing at scale: An auction-based approach. Proceedings of 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2016), California, USA, pp. 1–10 (2016)

35. Liu, J.X., et al.: Budget-aware dynamic incentive mechanism in spatial crowdsourcing. J. Comput. Sci. Tech. 32(5), 890–904 (2017)

36. OpenStreetMap is a free, editable map of the whole world that is being built by volunteers largely from scratch and released with an open-content license. https://www.openstreetmap.org (2020)

37. Uber is an on-demand and cashless car service that allows you to use your smartphone or tablet to request rides from private drivers. https://www.uber.com/ (2020)

38. Moshe, B., et al.: Dynamic pricing with limited supply. ACM Transactions on Economics and Computation 3(1), 1–26 (2001)

39. Adish, S., Andreas, K.: Truthful incentives in crowd-sourcing tasks using regret minimization mechanisms. Proceedings of the 22nd international conference on World Wide Web, pp. 1167–1178 (2013)

How to cite this article: Zhou Z, Chen R, Guo S: A domain-of-influence based pricing strategy for task assignment in crowdsourcing package delivery. IET Intell Transp Syst. 2021;15:808–823. https://doi.org/10.1049/itr2.12062.