Real-Time Cross Online Matching in Spatial Crowdsourcing

Yurong Cheng [*1], Boyang Li [#2], Xiangmin Zhou [†3], Ye Yuan [*1], Guoren Wang [*1], Lei Chen [§4]

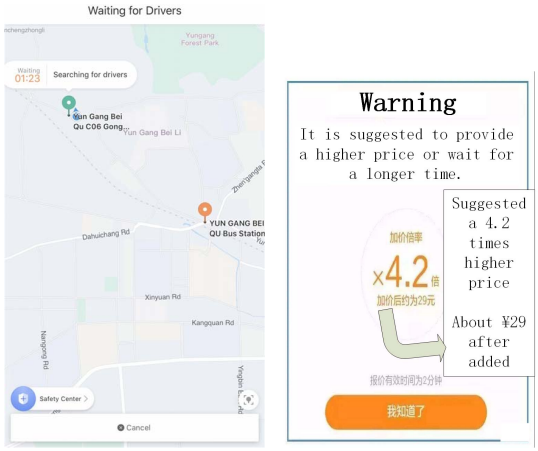
[*] *Beijing Institute of Technology, China* [#] *Northeastern University, China*
[†] *RMIT University, Melbourne, Australia* [§] *Hong Kong University of Science and Technology*
[1] {yrcheng,yuan-ye,wanggr}@bit.edu.cn [2] liboyang@stumail.neu.edu.cn [3] Xiangmin.zhou@rmit.edu.au
[4] leichen@cse.ust.hk

***Abstract*—With the development of mobile communication techniques, spatial crowdsourcing has become popular recently. A typical topic of spatial crowdsourcing is task assignment, which assigns crowd workers to users' requests in real time and maximizes the total revenue. However, it is common that the available crowd workers over a platform are too far away to serve the requests, so some user requests may be rejected or responded at high money cost after long waiting. Fortunately, the neighbors of a platform usually have available resources for the same services. Collaboratively conducting the task allocation among different platforms can greatly improve the quality of services, but have not been investigated yet. In this paper, we propose a Cross Online Matching (COM), which enables a platform to "borrow" unoccupied crowd workers from other platforms for completing the user requests. We propose two algorithms, deterministic cross online matching (DemCOM) and randomized cross online matching (RamCom) for COM. DemCOM focuses on the largest obtained revenue in a greedy manner, while RamCom considers the trade-off between the obtained revenue and the probability of request being accepted by the borrowed workers. Extensive experimental results verify the effectiveness and efficiency of our algorithms.**

I. Introduction

With the development of intelligent phone and mobile communication techniques (e.g. 5G), spatial crowdsourcing platforms are showing their irreplaceablity in people's daily life. Most users have several spatial crowdsourcing service APPs on their mobile phones, such as intelligent transportation (e.g. Uber [1] and DiDi [2]), food delivery (e.g. Meituan [3] and Ele.me [4]), and express (e.g. FedEx [5] and SF Express [6]). These platforms provide the convenient services to users, as well as the economic gain to crowd workers. As statistics on Uber in Aug. 2018, 436,000 Uber rides take place per day, compared to 275,000 taxi rides, and 122,000 Lyft rides [7]. Designing advanced task allocation is promising for improving the service quality of these platforms, enhancing their revenue and improving their user experience.

A good spatial crowdsourcing platform should be able to assign crowd workers to serve real-time requests considering the users' satisfaction and its own benefit [8]. This requires that the total travel distance of crowd workers is minimized or the total revenue of the platform is maximized. Existing studies [9] [10] focus on the situation where users' requests are responded by allocating the workers over a single platform.



(a) No Available Crowd Workers (b) Suggested a Higher Price

Fig. 1. A use case on DiDi Platform

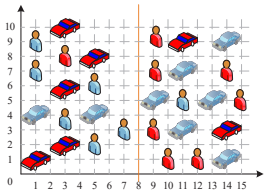


Fig. 2. Non-uniform Distribution of Requests and Crowd Workers

However, in real world, service on a single platform may lead to rejection of requests due to a high cost of both time and money. For example, Fig. 1(a) shows a request over DiDi, a taxi and ride-sharing service platform. The green location is the origin, while the orange one is the destination. This request is not responded since no available crowd workers nearby can serve it. Presently, DiDi handles this problem by conditionally providing services based on users' preference with respect to the service cost and waiting time. The user starting this request will be asked if s/he would like to wait for a longer time until any available worker happens to appear and can serve it. For example, in Fig. 1(a), the user is expected to wait about 1 minute and 23 seconds before the request is responded. The "1'23" is the response time of the request, not the waiting time for user. It means that, after "1'23", the platform starts to search drivers to serve this user. After such a driver is assigned, the user waits for the driver to travel to

2375-026X/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDE48307.2020.00008

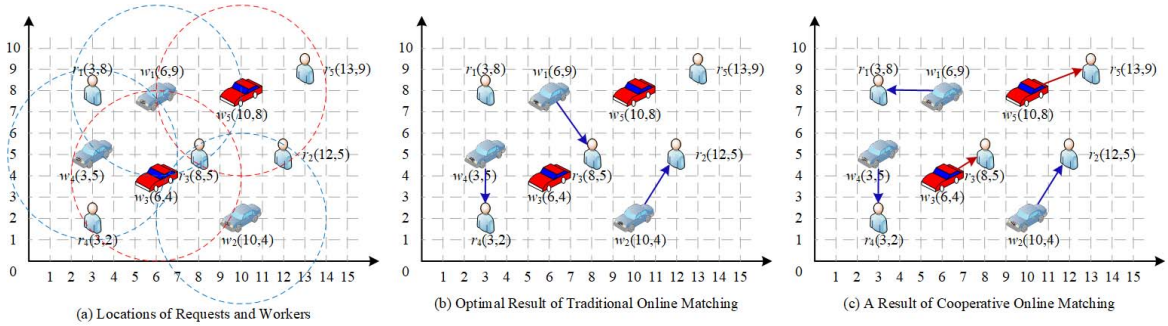


Fig. 3. Traditional Online Matching vs Cooperate Online Matching

TABLE I
VALUE OF REQUESTS

Requests	r_1	r_2	r_3	r_4	r_5
Value	4	9	6	3	4

TABLE II
ARRIVAL ORDER

Time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
Arrival Order	w_1	w_2	r_1	w_3	r_2	r_3	w_4	r_4	w_5	r_5

her/his current location. Or the platform intermediately calls a driver from a further place for service if the user would like to pay a higher price. For instance, the request shown in Fig. 1(a) is suggested to pay a 4.2 times higher price, which is about ¥29 ($\approx$ USD 4.3) in total, as shown in Fig. 1(b). Otherwise, the request would be rejected by the platform. Obviously, all solutions would unavoidably reduce the satisfaction of users, leading to the lose of the users over the platform.

Why does the case in Fig. 1(a) happen in the real world? The main reason behind this is the non-uniform distribution of requests and crowd workers. Due to the location-aware dynamic scheduling of workers, it is common that a large number of workers may gather in a region with few requests, while quite a lot requests may come from another place with few workers. Take the situation shown in Fig. 2 as an example. The cars and users in red belong to one spatial crowdsourcing platform, while those in blue belong to another platform providing the same services. In the left region, the users in blue are much more than the cars in blue, while the users in red are much less than the cars in red. However, in the right region, the reverse happens with respect to the user and car distributions. If more than two users in blue in the left region submit a request, the corresponding platform would have to ask a blue car in the right region to serve the third user after long traveling. As illustrated by Fig. 1, the waiting situation is caused by this biased distribution of workers and users, which introduces much higher time and money costs, leading to the downgrading of user satisfaction.

Fortunately, in practice, we find that different platforms may provide the same service, which provides a new clue of reducing the problems of non-uniform distribution. For example, DiDi and Shenzhou are two popular ride-sharing platforms, while Meituan, Ele.me and Baidu are three food delivery service platforms in China. When one platform has no available workers nearby to serve an emerged request, another platform may happen to have unoccupied available workers. This commonly happens among different apps providing the same services to the same requests. Smart users usually find that they cannot be served by one app (eg. DiDi) currently (except that they would like to wait or spend more money), but can be responded intermediately if they submit this request to another app (e.g. Shenzhou). Also, this is the reason that we often see some smart crowd workers (both drivers or food deliveryman) always open several apps at the same time to expect more assigned requests. For the case shown in Fig. 2, when the blue cars are not enough in the left region, some red cars belonging to other platforms can be available there. If we design an advanced solution that allows some unoccupied crowd workers from other platforms to serve some requests of the user target platform, this platform would not have to move a worker far away. Thus, the users would not incur the long waiting time and high service cost, or the risks of receiving the rejection of their requests. As a result, the user target platform can serve more requests for higher revenue and user can have better online experience. Nevertheless, just like the above statement, while opening several apps at the same time can achieve the completeness of more requests, it has two main shortcomings in practice. Firstly, for the users, it is too fussy to try different apps one by one, typing the same requirement over and over. Secondly, for the workers, if these platforms assign requests to them at the same time, they may reject some assigned requests without a plausible reason, leading to a punishment from the platforms in terms of money and credibility. Thus, it is highly demanded that a more intelligent strategy be designed to systematically conduct the proper cooperation among different platforms. Fig. 3 shows a scenario of this online matching across platforms.

Example 1: Fig. 3(a) shows the locations of 5 persons (i.e., requests) and 5 cars (i.e. workers). Each circle centered by a worker shows his/er service range. All the requests and workers in blue belong to the same platform, while those in red belong to other ones. Table I lists the value of each request, which indicates how much each user should pay when his request is completed. Table II shows the arrival order of each worker and request. With the existing online matching [9] [10], the optimal solution is as in Fig. 3(b). In this case, the platform can serve 3 requests, and the optimal revenue is $9 + 6 + 3 = 18$. If we borrow w_3 and w_5 from other platforms, the new matching result is as in Fig. 3(c). In this case, the platform can serve 5 requests. Suppose that a

worker from other platforms will receive 50%[1] of the overall payment for his completed request from the target platform. The new revenue of the target platform as in Fig. 3(c) is $4 + 9 + 6 \times 50\% + 3 + 4 \times 50\% = 21$. Thus, a cooperation crossing platforms helps to achieve a win-win situation.

In this paper, we propose a Cross Online Matching (COM) to deal with the cooperations across spatial crowdsourcing platforms. We should address two challenges. First, we need to construct a good model that takes into account both the task assignment and the incentive mechanism for the matching between workers and tasks. Lack of a proper incentive mechanism may lead to the high request rejection rate of workers from lender platforms, or the great revenue loss of the target platform. Second, we need to design efficient algorithms which guarantee the fast services over multiple platforms. Long-time waiting undoubtedly causes the lose of user requests and economic losses of the platforms. Specifically, we first propose a cross online matching model (COM), which allows the workers and requests to interact across platforms in an optimized way. Then we propose two alternative algorithms, deterministic cooperative online matching and randomized cooperative online matching, for real-time request processing. Our contributions are summarized as follows:

- We propose Cross Online Matching (COM) for the cooperations across different platforms. COM integrates the task assignment and incentive mechanism in spatial crowdsourcing, which strives for the optimized task allocation over multiple platforms.
- We propose two algorithms, DemCOM and RamCom, for COM. While DemCOM focuses on the largest revenue from cooperation greedily, RamCom considers the trade-off between the obtained revenue and the probability of request being accepted by cooperative platforms.
- We analyze the competitive ratios (CRs) of two algorithms to evaluate their performance. DemCom obtains the same CR with the greedy algorithm for online matching, while RamCom can achieve a CR up to $\frac{1}{8e}$.
- We conduct extensive experiments on both real and synthetic datasets to verify the effectiveness and efficiency of our proposed approach.

II. Problem Statement

In this section, we formally define the cross online matching problem, and present two online models for this matching.

A. Problem Definition

Before proceeding to the problem formulation, we first introduce five important concepts, Request, Inner Crowd Worker, Outer Crowd Worker, Outer Payment and Revenue.

Definition 2.1 (Request): A request in a spatial crowdsourcing platform is a triplet $r =< t, \mathbf{l}_r, v_r >$, where t is the arrival time of r, $\mathbf{l}_r$ denotes the location in a 2D space of r at time t, and v_r is the value that the requester can pay for the platform after being served.

[1]Later in Sections III and IV, we will illustrate how this percentage is calculated. Here, we just make an assumption that this ratio is known.

In online matching [9] [10], it is assumed that the platform must determine immediately whether to serve or reject an incoming request. Unlike online matching, the cross online matching algorithms decide whether an incoming request is served by the workers on the target platform or by those on other platforms, or rejected.

Definition 2.2 (Inner Crowd Worker): An inner crowd worker, denoted as $w_{in} =< t, \mathbf{l}_{w_{in}}, rad_{w_{in}} >$, is a worker serving a request target platform. Here, t is the arrival time of w_{in}, $\mathbf{l}_{w_{in}}$ the location in a 2D space of w_{in} at time t, and $rad_{w_{in}}$ his/er service radius in the 2D space.

When a worker arrives at the platform, s/he will wait in a waiting list until a request is assigned. Thus, a worker can only serve a request arriving after him/er. For instance, in Example 1, r_1 can only be served by w_1, as w_1 arrives before r_1, and its service radius covers r_1's location. Each platform maintains a *waiting list* of workers, ordered by their arrival time as shown in Table II. A worker being assigned to a request would be deleted from the waiting list.

Definition 2.3 (Outer Crowd Worker): An outer crowd worker denoted as $w_{out} =< t, \mathbf{l}_{w_{out}}, rad_{w_{out}} >$ is a crowd worker belonging to non-target spatial crowdsourcing platforms. Here, t is the arrival time of w_{out}, $\mathbf{l}_{w_{out}}$ the location in a 2D space of w_{out} at time t, $rad_{w_{out}}$ his services radius in the 2D space. The platforms served by outer crowd workers are ***cooperative platforms*** or *lender platforms*, and the requests assigned to outer crowd workers are *cooperative requests*.

The outer crowd workers may belong to several cooperative platforms that only share the information of their unoccupied workers. Since the platforms providing the same services in the real world are actually competitors, the accurate distribution of workers and requests of each platform is not visible to its competitors. The waiting list of each platform also maintains the information of outer crowd workers. An outer crowd worker being assigned to any request would be deleted from all its waiting lists over all platforms.

Definition 2.4 (Outer Payment): When a platform asks the outer workers to serve a request r, if an outer worker w_{out} would like to serve r and obtain a payment $v'_r \in (0, v_r]$, we call v'_r as the outer payment of request r.

Given that rejecting a request r may cause the user's dissatisfaction, v'_r may equal v_r if the platform providing r cares about its users' satisfaction more than revenue.

Definition 2.5 (Revenue): If an inner worker w_{in} is assigned to a request r, the platform would gain a value of v_r. If an outer worker w_{out} is assigned to serve a request r with outer payment $v'_r \in (0, v_r]$, then the platform would gain a value of $v_r - v'_r$. Let M be a feasible matching, M_{in} be the matching that requests are served by inner crowd workers, and M_{out} be the matching that requests are served by outer crowd workers, then the total revenue of the platform is calculated by:

$$Rev = Rev_{in} + Rev_{out} = \sum_{i=1}^{|M_{in}|} v_{r_i} + \sum_{i=1}^{|M_{out}|} (v_{r_i} - v'_{r_i}) \quad (1)$$

We address the problem of cross online matching by considering the collaboration across spatial crowdsourcing platforms. The *Cross Online Matching (COM)* problem is formally defined as follows:

Definition 2.6 (COM Problem): Given a set of request R, a set of inner crowd workers W_{in}, and a set of outer crowd workers W_{out}, each worker or request arrives sequentially. COM calculates a feasible matching result M whose revenue Rev is maximized, satisfying the following constraints:

- Time constraint: workers can only serve the requests arriving at the platform after them.
- 1-By-1 constraint: one worker can only serve one request at a time point, and one request can only be served by one worker at a time point.
- Invariable constraint: once a worker w is assigned to a request r, their matching cannot be changed until the service is completed.
- Range constraint: a worker can only serve such requests whose locations are within the radius of this worker.

In COM, the 1-By-1 constraint can be equivalently generalized to the applications that one worker serve several requests at one time point, by treating this worker as multiple workers located in the same place. After a worker w finishes the service of r, s/he can come back to the platform again at a new time point. Although COM uses the Euclidean distance, without loss of generality, it can be equivalently changed into the shortest path distance in road networks by just changing the service range from circulars to irregular shapes.

Traditional online task assignment (TOTA) [9] is a special case of COM where $W_{out} = \phi$. Same as TOTA [9], COM is NP-hard. To evaluate the performance, online algorithms focus more on *competitive ratio (CR)* than approximate ratio [9]. TOTA defined two types of CRs in two online models. Similarly, we define two types of CRs in two online models for COM, which will be illustrated in the next subsection.

B. Online Models of COM

In TOTA, the *competitive ratio (CR)* is defined to compare the performance of the algorithms with respect to the optimal solution of its corresponding offline version [9]. For the offline version, the platform knows the spatiotemporal information and arrival order of all requests and workers before assigning the tasks. Different from TOTA, our COM brings in the outer crowd workers to complete the requests. Thus, the offline version of our COM should also know the spatiotemporal information, arrival order, and even the outer payment of the outer crowd workers in advance. We treat all requests as a set of vertices, and all workers as the other set of vertices. For each worker, we construct multiple edges, each of which is connected to a request that s/he can serve satisfying all constraints in Definition 2.6, and weighted by the value of serving the corresponding request. Then the offline version can be treated as a maximal weighted bipartite graph matching [11]. For example, Fig. 4(a) shows the equivalent weighted bipartite graph of TOTA in Example 1 (i.e., Fig. 3(b)). Fig. 4(b) shows the equivalent weighted bipartite graph of COM in Example 1 (i.e., Fig. 3(c)). Thus, the optimal solution of the offline COM can be calculated by the maximal weighted bipartite graph matching as in [11].

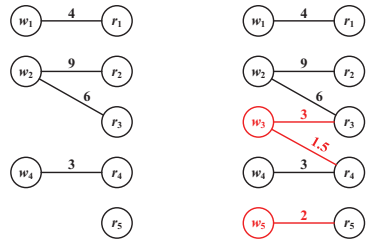


Fig. 4. Offline Versions of Example 1

Based on the offline COM, we introduce two types of CRs under two online models as follows.

Definition 2.7 (CR in the Adversarial Model [9]): The competitive ratio in the adversarial model of a specific online algorithm for COM is the minimum ratio between the result of the online algorithm and the optimal result over all possible arrival orders of the requests and the workers, i.e.,

$$CR_A = \min_{\forall G(T,W_{in},W_{out}) \text{and} \forall ord \in Ord} \frac{MaxSum(M)}{MaxSum(OPT)} \quad (2)$$

where $G(T, W_{in}, W_{out})$ is an arbitrary input of requests and workers, Ord is the set of all possible input orders, ord is one order in Ord, $MaxSum(M)$ is the total revenue obtained by the online algorithm, and $MaxSum(OPT)$ is the optimal total utility of the offline scenario.

Definition 2.8 (CR in the Random Order Model [9]): The competitive ratio in the random order model of a specific online algorithm for COM is defined as follows.

$$CR_{RO} = \min_{\forall G(T,W_{in},W_{out})} \frac{\mathbb{E}[MaxSum(M)]}{MaxSum(OPT)} \quad (3)$$

where $G(T, W_{in}, W_{out})$ is an arbitrary input of requests and workers, $\mathbb{E}[MaxSum(M)]$ the total revenue expectation of the online algorithm over all possible arrival orders of tasks and workers in the specific $G(T, W_{in}, W_{out})$, and $MaxSum(OPT)$ the optimal total utility of offline scenario.

The adversarial model bounds the worst-case ratio over all possible inputs and arrival orders, while the random order model bounds the worst average performance of an online algorithm. According to [12], for online algorithms in spatial crowdsourcing, usually the appearance probability of the worst case is almost $\frac{1}{k!}$, where $k = \min\{|T|, |W|\}$, and $|T|$ (resp. $|W|$) is the number of requests (resp. workers) in the platform. Thus, researchers focus on CR_{RO} more than CR_A.

III. Deterministic Method

This section proposes a *Deterministic Cross Online Matching (DemCOM)* to solve the COM in a greedy manner.

A. Basic Idea

Whenever a request r comes to the platform, the platform can obtain a revenue v_r if r is completed by an inner crowd worker. The obtained revenue must be smaller than v_r if r is

completed by an outer crowd worker. To obtain the largest revenue, DemCOM gives an inner crowd worker a higher priority to serve an incoming request r. If no inner crowd worker can serve r, DemCOM calculates the minimum outer payment v'_r which may attract any outer crowd workers. If the outer payment is smaller than v'_r, no outer crowd worker would like to serve r. This can ensure that the platform can obtain a largest revenue $v_r - v'_r$. Then DemCOM determines whether and which outer crowd worker can serve r based on the acceptance probability of a worker to the request r, which is estimated based on the history requests of each outer crowd worker and formally defined as follows.

Definition 3.1 (Acceptance Probability of a Worker): Given a cooperative request r with outer payment v'_r, a crowd worker w with N history requests, the acceptance probability of w reflects the probability that w would like to serve r with value v'_r, which is denoted as $pr(v_l, x)$ and calculated by

$$pr(v'_r, w) = \frac{N(v \leq v_l)}{N} \tag{4}$$

where N is the number of the worker's completed history requests; $N(v \leq v'_r)$ is the number of such history requests whose value is not larger than v'_r.

We use Equation 4 to estimate that a worker is more likely to serve r if v'_r is a relatively higher price in a worker's request completion history, and reject it otherwise. To determine whether each outer crowd worker w would like to serve r with price v'_r, we generate a random number x from 0 to 1. We assert that w would like to serve r and put w into a set W^r_{out} if $x \leq pr(v'_r, w)$. We check all the outer crowd workers satisfying all constraints in Definition 2.6. The request r is rejected if $W^r_{out} = \phi$, while a worker is selected from W^r_{out} to serve it otherwise.

B. Algorithm Description

We first describe the procedure of DemCOM. Then we illustrate the minimum outer payment v'_r with which some outer crowd workers would like to provide service.

1) Procedure of DemCOM: Algorithm 1 shows the procedure of DemCOM. The input of DemCOM is the request set R, the inner crowd worker set W_{in}, and the outer crowd worker set W_{out}. Each $r \in R$, $w_{in} \in W_{in}$, and $w_{out} \in W_{out}$ arrives at the platform sequentially. The output is a feasible matching result M with its corresponding total revenue Rev. Initially, let $M = \phi$ and $Rev = 0$ (Line 1). For each new arrival request r, we check all the unoccupied inner crowd workers to find the ones who can serve r satisfying all constraints in Definition 2.6. If multiple inner crowd workers can serve r, we assign r to the nearest worker, and the total revenue is updated by $Rev = Rev + v_r$ (Lines 2-6). If no inner crowd worker can serve r, DemCOM identifies all the outer crowd workers who have the ability to serve r satisfying all constraints in Definition 2.6 and put them into a set W^r_{out} (Line 8). If $W^r_{out} = \phi$, DemCOM rejects r (Lines 9-10). Otherwise DemCOM calls Algorithm 2 to estimate the minimum outer payment v'_r (Line 12). The details of Algorithm 2 will be illustrated in Section III-B2. If the estimated outer payment v'_r is even larger than v_r, DemCOM would directly reject r, since the platform would lose money to complete r (Lines 13-14). Otherwise, DemCOM determines whether and whom to serve r with value v'_r (Lines 15-26). For each worker $w_{out} \in W^r_{out}$, we calculate his/er acceptance probability $pr(v'_r, w_{out})$, and generate a random number $x \in [0, 1]$. If $x > pr(v'_r, w_{out})$, it means that w_{out} would not like to serve r, and is removed from W^r_{out} (Lines 17-20). After checking all workers in W^r_{out}, if $W^r_{out} \neq \phi$, we assign r to its nearest outer crowd worker, and the total revenue is updated by $Rev = Rev + v_r - v'_r$ (Lines 21-24). Otherwise, DemCOM rejects r (Lines 26).

Algorithm 1: Deterministic Algorithm for COM (DemCOM)

```
Input: R, W_in, W_out
Output: A Feasible Matching M, total revenue Rev
1  M = φ, Rev = 0 ;
2  for each new arrival request r ∈ R do
3      Greedily check whether w_in ∈ W_in can serve r ;
4      if r can be served by w_in ∈ W_in then
5          Put < r, w_in > into M ;
6          Rev+ = v_r ;
7      else
8          Calculate a set of crowd workers W^r_out who can serve r ;
9          if W^r_out = φ then
10             Reject r ;
11         else
12             Estimate the minimum outer payment v'_r using Algorithm 2 ;
13             if v'_r > v_r then
14                 Reject r ;
15             else
16                 for each w_out ∈ W^r_out do
17                     Calculate pr(v'_r, w_out) ;
18                     Generate a random number x ∈ [0, 1] ;
19                     if x > pr(v'_r, w_out) then
20                         Remove w_out from W^r_out ;
21                 if W^r_out ≠ φ then
22                     Greedily assign r to a w_out ∈ W^r_out ;
23                     Put < r, w_out > into M ;
24                     Rev+ = v_r − v'_r ;
25                 else
26                     Reject r ;
27 Return (M, Rev) ;
```

Example 2 (DemCOM): Back to our running example in Example 1. When r_1 arrives, the inner worker w_1 can serve it. w_1 is assigned to r_1. When r_2 arrives, the inner worker w_2 can serve it. w_2 is assigned to r_2. When r_3 arrives, no inner crowd worker can serve it. Suppose that the outer payment of r_3 is 2. If w_3 would like to serve r_3, we assign r_3 to w_3. When r_4 arrives, w_4 is assigned to it. When r_5 arrives, no inner crowd worker can serve it. Suppose that the outer payment of r_5 is 3, and the outer crowd worker w_5 serves it. Thus, the obtained total revenue is $4 + 9 + (6 - 2) + 3 + (4 - 3) = 21$.

2) Estimating the minimum Outer Payment: We calculate the minimum outer payment v'_r of a request r, with which r can be possibly served by an outer crowd worker. We estimate such v'_r using a Monte Carlo sampling method. We choose Monte Carlo sampling for estimating the minimum outer payment, because each crowd worker chooses a request independently, which satisfies the condition of Monte Carlo

sampling. Moreover, the precision of Monte Carlo sampling is independent of the size and distribution of datasets. Since the appearance time and location of users and those of requests are totally random over spatial crowdsourcing platforms, there exists no priori knowledge on the distribution of data, which fits well the Monte Carlo sampling.

In each sampling instance, we treat each outer crowd worker whose service range can cover the location of r as a random variable x, and sample each worker x_i who has an acceptance probability $pr(v_l, x_i)$ (see Definition 3.1) with price v_l for request r. If this worker would like to serve r, then $x_i = 1$; otherwise $x_i = 0$. Notice that we just use sampling to simulate the acceptance of the workers with respect to r. For all x_i, if any $x_i = 1$, it means that the estimated v_l is larger than the minimum outer payment v'_r. The sampled value should be smaller than v_l for next sampling process. Otherwise, the estimated v_l is smaller than v'_r, the estimated value should be larger than v_l in the next sample. Using this sampling method, the minimum outer payment v'_r is approximated by these v_l. The details are shown in Algorithm 2.

The input of Algorithm 2 is the cooperative request r, and the set of outer crowd workers that satisfy all constraints in Definition 2.6. The output is the estimated minimum outer payment v'_{rm}. Let $n_s = \lceil 4 \ln \frac{2}{\xi} / \eta^2 \rceil$ denote the number of sampling instances (Line 1). Here, ξ and η control the accuracy of sampling, which will be discussed in detail in Section III-C. Initially, let $v'_r = 0$, $v_l = 0$, $v_h = v_r$, and $v_m = \frac{1}{2} v_h$. These parameters are used to record the intermediate results of dichotomy when calculating each sampling instance. Firstly, we sample each instance (Lines 3-15), in which the acceptance probability of each worker w_{out} to request r with value v'_r is calculated by Equation 4. We sample each outer crowd worker with probability $pr(v_r, x_j)$, and check whether someone would like to serve it (Line 4). If no outer crowd worker would like to serve r with value v_r in this instance (i.e. all $x_j = 0$), the algorithm directly sets this instance as $v_r + \epsilon$, and goes to sample another instances (Lines 5-6). Otherwise, we apply dichotomy to estimate the instance (Lines 7-15). We sample each outer crowd worker with probability $pr(v_m, x_j)$ to serve r (Line 9). If any outer crowd worker would like to serve r, the sampled value v_m is larger than the estimation value. Then we set $v_h = v_m$ (Lines 10-11). Otherwise, we set $v_l = v_m$ (Lines 12-13), and update v_m using $v_m = \frac{1}{2}(v_h - v_l) + v_l$ (Line 14). we recursively conduct this process (Lines 9-14) until $v_m = v_l > \epsilon$, where v_m is the value of this sampled instance (Line15). After sampling n_s instances, we calculate the average value and output it (Line 16).

Algorithm 2: Estimate the minimum Outer Payment

Input: r, W^r_{out}
Output: The minimum outer payment v'_r

1 $n_s = \lceil 4 \ln \frac{2}{\xi} / \eta^2 \rceil$;
2 $v'_r = 0$, $v_l = 0$, $v_h = v_r$, $v_m = \frac{1}{2} v_h$;
3 **for** *int* $i = 0$; $i < n_s$; $i++$ **do**
4 Sample each $w_{out} \in W^r_{out}$ with probability $pr(v_r, x_j)$ to serve r;
5 **if** *there exists* $x_j = 1$ **then**
6 $v'_r += (1 + \xi) v_r$;
7 **else**
8 **while** $v_m - v_l > \xi v_r$ **do**
9 Sample each $w_{out} \in W^r_{out}$ with probability $pr(v_m, x_j)$ to serve r ;
10 **if** *there exists* $x_j = 1$ **then**
11 $v_h = v_m$;
12 **else**
13 $v_l = v_m$;
14 $v_m = \frac{1}{2}(v_h - v_l) + v_l$;
15 $v'_{rm} += v_l$;
16 $v'_{rm} = v'_{rm} / n_s$;
17 Return (v'_{rm}) ;

C. Algorithm Analysis

In Algorithm 2, each instance is sampled independently. According to Monte Carlo sampling theory, the following lemma holds.

Lemma 1: According to Monte Carlo sampling theory, if $n_s \geq 4 \ln \frac{2}{\xi} / \eta^2$, we have

$$pr((\hat{v'_{rm}} - v'_{rm}) > \xi v'_{rm}) < \eta$$

where $\xi \in (0, 1)$, $\eta \in (0, 1)$.

This lemma ensures the accuracy of the minimum outer payment. Given the minimum outer payment v'_{rm}, we deduct the competitive ratio of DemCOM.

Theorem 1: The competitive ratio in adversarial model CR_A of the DemCOM algorithm has no bounds. The competitive ratio in random order model CR_{RO} of the DemCOM is same as that of the greedy algorithm for the tradition online task assignment (TOTA) problem.

We posted the proofs in [13].

Complexity Analysis. The calculational complexity of Algorithm 2 is $O(n_s \times \log(\max(vr)) \times |W_{out}|$, where n_s is the number of the sampling instances; $\max vr$ is the largest value of all the requests; $|W_{out}|$ is the number of the outer crowd workers. In Algorithm 1, given an incoming request, the complexity of checking whether an inner crowd worker can serve it is $O(|W_{in}|)$, where $|W_{in}|$ is the number of inner crowd workers. The complexity of finding outer crowd worker to serve it is $O(n_s \times \log(\max(vr)) \times |W_{out}| + |W_{out}|)$. Thus, the complexity of DemCOM algorithm is $O(|R|(|W_{in}| + n_s \times \log(\max(vr)) \times |W_{out}|))$. The space complexity is $O(|W| + |R|)$, where $|W|$ (resp. $|R|$) is the size of workers (resp.requests).

D. Shortcomings of DemCOM

In DemCOM, a request r is assigned to an outer crowd worker only when it cannot be served by any inner crowd worker. In a bad case, the inner crowd workers are assigned to the requests with small values. When a request with a large value arrives, the inner crowd workers are all occupied. Thus, the requests with large values have to be served by the outer crowd workers. According to our experiments in Section V, we can see that the minimum outer payment should be about 70% of each cooperative request value. Thus, it is not economical to handle a request with a large value as a cooperative request. The other shortcoming is that Algorithm 2 only calculates the minimum outer payment. According to our experiments in

Section V, the ratio that the outer crowd workers would like serve cooperative requests with the minimum outer payment is only about 17%. This means that a large number of cooperative requests are still rejected. If small number of requests are accepted by the outer crowd workers, the COM would degrade into the TOTA (according to the proof in [13]). Thus, we consider a randomized algorithm to overcome the above two disadvantages of DemCOM algorithm.

IV. Randomized Algorithm

In this section, we propose a *randomized algorithm for cross online matching* problem (called RamCOM).

A. Basic Idea

Recall that DemCOM only assigns the requests to the outer crowd workers while the inner crowd workers can never serve them. Thus the inner crowd workers may miss the future requests with larger values. Also, the outer payment provided by DemCOM may be too small to stimulate the outer crowd workers. In our RamCOM, we technically pick up the requests with larger values to the inner crowd workers, and leave those with smaller values to the outer crowd worker. Moreover, unlike DemCOM that only considers the minimum outer payment to ensure the largest revenue obtained from the cooperative requests, RamCOM also consider the probability that an outer worker would like to serve a cooperative request. Specifically, to pick up the requests with larger values, we calculate a random threshold of values. If the value of a request is larger than the threshold, we assign it to an inner crowd worker. For the second purpose, we consider a trade-off result between the obtained revenue of the platform and the probability that the outer crowd workers would like to serve a cooperative request. Inspired by the existing study [14], we involve a definition of *maximum expect revenue* as follows.

Definition 4.1 (Maximum Expect Revenue): Given a cooperative request r with value v_r and an outer payment v'_r, a set of workers W who have the ability to serve r satisfying all constraints in Definition 2.6, the *acceptance probability of an outer payment*, denoted as $pr(v'_r, W)$, is the probability that any worker in W who would like to serve r with value v'_r. Further given $pr(v'_r, W)$, the *expect revenue* (denoted as $\mathbb{E}(v'_r, W)$) is calculated as

$$\mathbb{E}(v'_r, W) = (v_r - v'_r) \times pr(v'_r, W) \tag{5}$$

and the *maximum expect revenue* $\mathbb{E}(v_r, W)_{\max}$ is

$$\mathbb{E}(v_r, W)_{\max} = \max_{0<v'_r \le v'_r} (\mathbb{E}(v'_r, W)) \tag{6}$$

The maximum expect revenue is a trade-off between the obtained revenue of a platform and the probability that outer crowd workers would like to serve a cooperative request. A larger outer payment indicates a higher possibility that an outer crowd worker would like to serve the request. However, when the outer payment is larger, the revenue obtained by the platform becomes smaller. Given a set of requests and workers in a spatial crowdsourcing platform, [17] studied the pricing strategy for each request such that the expected revenue (Definition 4.1) of the platform is maximized. For each request, the users prefer a price as small as possible, while the worker/platform pursues a high price. Such a trade-off is handled by constructing a bipartite graph model that considers the demand and supply relationship in different locations and time, and approximately estimating the price based on this graph. The method in [17] has guaranteed the approximate ratio of the estimated price with a small computational cost, which is appropriate to computing the maximized expected revenue in our COM.

B. Algorithm Description

The details of the RamCOM is shown in Algorithm 3. The input is the set of requests R, the inner crowd workers W_{in}, and the outer crowd workers W_{out}. The output is a feasible matching M and its corresponding total revenue Rev. Let $\theta = \lceil \ln(\max(v_r)+1) \rceil$, where $\max(v_r)$ is the largest value of all requests in R. Let k control the threshold of the request values which are assigned to the inner crowd workers. An integer from $\{1, \cdots, \theta\}$ with probability $\frac{1}{\theta}$ is selected and given to k. Initially, let $M = \phi$, and $Rev = 0$ (Lines 1-2). For each new arrival request r, if its value $v_r > e^k$, we find all the inner crowd workers satisfying all the constraints in Definition 2.6. We randomly select one from these inner crowd workers to serve r (Lines 4-8). Otherwise, we call the algorithms in [14] to calculate the maximum expect outer payment v_{re} (Line 10). With the maximum expect outer payment v_{re}, we assign r to the outer crowd workers. We recursively conduct the above process, until all requests are assigned.

Algorithm 3: Randomized Algorithm for COM (RamCOM)

```
   Input: R, W_in, W_out
   Output: A Feasible Matching M, total revenue Rev
1  θ = ⌈ln(max(v_r) + 1)⌉, M = φ, Rev = 0 ;
2  k ← randomly choosing an integer from {1, ··· , θ} with probability 1/θ ;
3  for each new arrival request r ∈ R do
4      if v_r > e^k then
5          Find all w_in that can serve r satisfying all constraints ;
6          if there exists w_in that can serve r then
7              Randomly assign a w_in to serve r ;
8              Rev+ = v_r ;
9      else
10         Calculate ERev(r) and v_re using the algorithm in [14];
11         Call Lines 13-26 of Algorithm 1 to assign outer crowd workers to
           serve r ;
12 Return (M, Rev) ;
```

Example 3: Considering Example 1, where $\max(v_r) = 9$, $\theta = 3$. Suppose $k = 1$, the threshold is e. When r_1 arrives, since $v_{r_1} > e$, the inner worker w_1 is assigned to r_1. When r_2 arrives, since $v_{r_2} > e$, the inner worker w_2 is assigned to r_2. When r_3 arrives, although $v_{r_3} > e$, no unoccupied inner crowd worker can serve r_3. Thus, we ask the outer crowd worker to serve r_3. Suppose the algorithms in [14] is used to calculate the distribution of $(v_{r_3} - v'_{r_3})$ as $\{1, 2, 3, 4, 5\}$ with corresponding acceptance probabilities $\{0.9, 0.8, 0.4, 0.3, 0.2\}$. Thus, we obtain the maximized $\mathbb{E}(v_{r_3}, W)_{\max} = 2 * 0.8 = 1.6$, the acceptance probability 0.8, the maximum expected outer pay-

TABLE III
REAL DATASETS

	DiDi			Yueche		
	RDC10	**RDC11**	**RDX11**	**RYC10**	**RYC11**	**RYX11**
$\|R\|$	91,321	100,973	57,611	90,589	100,448	57,638
$\|W\|$	9,145	11,199	2,441	7,038	9,333	2,686
rad (km)	1.0	1.0	1.0	1.0	1.0	1.0

TABLE IV
SYNTHETIC DATASETS

Factors	**Setting**
$\|R\|$	500, 1000, **2500**, 5k, 10k, 20k, 50k, 100k
$\|W\|$	100, 200, **500**, 1000, 2500, 5k, 10k, 20k
rad	0.5, **1**, 1.5, 2, 2.5
value distribution	**real**, normal

ment 4, and the obtained revenue 2. Using the same method, we assign r_4 to w_4 and r_5 to w_5.

C. Algorithm Analysis

Now, we analyze the competitive ratio of the RamCOM algorithm. The approximate ratio of the maximized expect revenue $\mathbb{E}(v_r, W)_{\max}$ of each request is $1/e$ [14]. Thus, we deduce the competitive ratio of the RamCOM algorithm.

Theorem 2: The competitive ratio of the RamCOM algorithm can reach $\frac{1}{8e}$.

We post the proofs in [13].

According to [14], the complexity of estimating v_{re} is $O(\max(v_r))$. The complexity of assigning each to inner crowd workers is $O(|W_{in}|)$, while that of each request to outer crowd workers is $O(|W_{out}|)$. Thus, the total complexity of RamCOM is $O(|R|(|W_{in}| + |W_{out}| + \max(v_r))$. The space complexity is $O(|W| + |R|)$, where $|W|$ (resp. $|R|$) is the size of workers (resp.requests).

V. EXPERIMENTAL EVALUATION

In this section, we evaluate the effectiveness, efficiency and scalability of our proposed COM algorithms by conducting a series of experiments on both real and synthetic datasets and report the experimental results.

A. Experimental Setup

We conduct experiments over real datasets from DiDi [15] and Yueche (a taxi calling service) [16]. We use 3 real datasets recording the requests and workers in Chengdu (a city of China) in Nov. 2016 (denoted as *RDC11* for DiDi company and *RYC11* for Yueche company respectively), those in Chengdu in Oct. 2016 (denoted as *RDC10* and *RYC10* respectively) and those in Xi'an (another city of China) in Nov. 2016 (denoted as *RDX11* and *RYX11* respectively). The details are shown in Table III. Here, $|R|$ is the average number of requests, and $|W|$ is the average number of workers on the platform for each day.

We also generate some synthetic datasets to test the scalability of our algorithms. For different cooperative platforms, we generate equal number of requests as well as equal number of workers. The requests and workers are picked up from the real datasets, *RDC11* and *RYC11*. We randomly select 250-50k requests and 50-10k users from each of these two datasets, which forms a set consisting of 500-100k requests and 100-20k users in total. The location and arriving time of each request and those of each worker keep same as those in *RDC11* and *RYC11*. The settings of these synthetic datasets are shown in Table IV. The ratio of request number to worker number is varying over different cities. For example, the ratio of $|R|$ to $|W|$ is about 10 in Chengdu, while about 25 in Xi'an. In addition, the default value has little influence to the experimental results on scalability. Thus, following the setting of existing work [9], we set the default values $|R| = 2500$ and $|W| = 500$.

We compare our DemCOM and RamCOM with the state-of-art online matching algorithm [9] solving the traditional online task assignment problem, which is denoted as *TOTA*. Also, to test the competitive ratio of our algorithm, we compare our algorithms with the offline version of COM introduced in Section II-B, which is denoted as *OFF*.

We evaluate the effectiveness of our algorithms in terms of five metrics: (1) the total revenue obtained by DiDi (resp. Yueche), denoted as Rev_D (resp. Rev_Y); (2) the number of requests completed by each algorithm on DiDi (resp. Yueche), denoted as |CpR(D)| (resp. |CpR(Y)|); (3) the total number of cooperative requests accepted by the two platforms, denoted as |CoR|; (4) the accepted ratio of cooperative requests, denoted as |AcpRt|; (5) the average rate of each outer payment v_r' to the request value v_r, denoted as $\overline{v_r'/v_r}$. We evaluate the efficiency of our algorithms by reporting the average response time of each request and the memory cost of the two platforms. We test the scalability of our algorithms in terms of the total revenue, response time, memory cost, and acceptance ratio w.r.t $|R|$, $|W|$, and rad, respectively. Our experiments are conducted on a server with 64GB Memory, 8TB hard disk, and 4 6-core Intel(R) Xeon(R) CPU X5675 @3.07GHz, with Linux system, using C++ and its standard template library (STL).

B. Effectiveness

We compare our approach with existing methods in terms of effectiveness, and report the matching results using four methods, including DemCOM, RamCOM, OFF and TOTA. The experiments are conducted over the real datasets, and the evaluation is evaluated in terms of five metrics: Total Revenue, Completed Requests, Cooperative Requests, Acceptance Ratio and Outer Payment Rate. The test results are reported in Tables V-VII.

1) Effectiveness w.r.t Total Revenue: As shown in the results, both our DemCOM and RamCOM algorithms achieve higher effectiveness than TOTA algorithm in terms of the total revenue. OFF performs best in terms of the total revenue among all methods, but can never be achieved in the real world since the arrival order of requests and workers are ignored. Comparing with TOTA, DemCOM and RamCOM can improve the total revenue of each platform for about ¥20,000 (about USD 2,143) and ¥70,000 (about USD 9,986) per day respectively. For one year, the increase can reach USD 3.6 million, which is very considerable in the real world. Comparing with TOTA, our DemCOM and RamCOM are 5% and 17.5% closer to the revenue upper bound calculated by OFF respectively. Note that COM increases the revenue without the increase of the arrived requests and workers over

TABLE V
RESULTS ON RDC10 AND RYC10

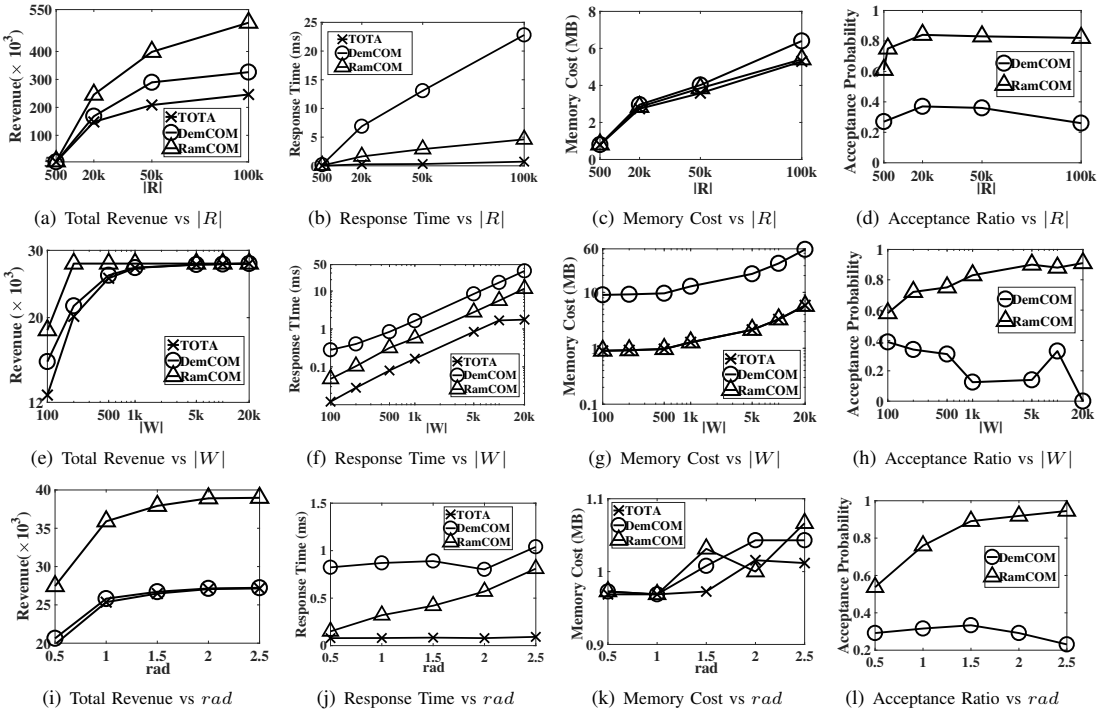
Methods	$Rev_D(\times 10^6)$	$Rev_Y(\times 10^6)$	Response Time (ms)	Memory (MB)	|CpR(D)|	|CpR(Y)|	|CoR|	|AcpRt|	$\overline{v'_r/v_r}$
OFF	1.752	1.743	0.34	13.58	91,321	90,589	-	-	-
TOTA	1.343	1.348	0.43	13.58	68,689	68,453	-	-	
DemCOM	1.369	1.372	0.43	13.58	71,931	71,721	7,077	0.16	0.72
RamCOM	1.436	1.437	0.56	13.58	69,186	68,560	72,417	0.66	0.81

TABLE VI
RESULTS ON RDC11 AND RYC11

Methods	$Rev_D(\times 10^6)$	$Rev_Y(\times 10^6)$	Response Time (ms)	|CpR(D)|	|CpR(Y)|	|CoR|	|AcpRt|	$\overline{v'_r/v_r}$
OFF	1.914	1.924	0.32	100,973	100,448	-	-	-
TOTA	1.612	1.594	0.52	81,912	81,706	-	-	-
DemCOM	1.621	1.614	0.52	85,737	85,460	6,220	0.17	0.70
RamCOM	1.645	1.646	0.75	82,385	82,760	91,699	0.75	0.82

TABLE VII
RESULTS ON RDX11 AND RYX11

Methods	$Rev_D(\times 10^6)$	$Rev_Y(\times 10^6)$	Response Time (ms)	|CpR(D)|	|CpR(Y)|	|CoR|	|AcpRt|	$\overline{v'_r/v_r}$
OFF	1.103	1.102	0.52	57,611	57,638	-	-	-
TOTA	0.512	0.509	0.50	24,695	24,907	-	-	-
DemCOM	0.525	0.523	0.53	26,818	26,736	6,531	0.09	0.77
RamCOM	0.555	0.549	0.55	26,730	26.666	16,487	0.25	0.82



(a) Total Revenue vs $|R|$ (b) Response Time vs $|R|$ (c) Memory Cost vs $|R|$ (d) Acceptance Ratio vs $|R|$

(e) Total Revenue vs $|W|$ (f) Response Time vs $|W|$ (g) Memory Cost vs $|W|$ (h) Acceptance Ratio vs $|W|$

(i) Total Revenue vs rad (j) Response Time vs rad (k) Memory Cost vs rad (l) Acceptance Ratio vs rad

Fig. 5. Scalability of COM Algorithms

a platform. This verifies the superiority of our COM over the TOTA in terms of the total revenue for each platform.

2) Effectiveness w.r.t Completed Requests: The numbers completed requests of DemCOM and RamCOM are both larger than that of TOTA, but smaller than that of OFF. OFF can never be achieved in the real world as discussed in Section V-B1. Comparing with TOTA, DemCOM and RamCOM can enable each platform complete more requests, and satisfy more users. Comparing with DemCOM, RamCOM completes fewer requests, but obtains more revenue. This means that the RamCOM completes more requests with larger values.

3) Effectiveness w.r.t Cooperative Requests: Since OFF and TOTA do not cooperate with other platforms, they do not have cooperative requests. Comparing with DemCOM, RamCOM complete more cooperative requests. This means that RamCOM remains more requests with smaller revenue to cooperative platforms, and the incentive mechanism of RamCOM makes more users complete cooperative platforms, comparing with DemCOM.

4) Effectiveness w.r.t Acceptance Ratio: Since OFF and TOTA do not cooperate with other platforms, they also do not have acceptance ratio. The accpetance ratio of RamCOM is larger than that of DemCOM. It means that the cooperative workers are more likely to accept the cooperative requests using RamCOM than DemCOM. That is, the incentive mechanism of RamCOM is better than that of DemCOM.

5) Effectiveness w.r.t Outer Payment Rate: Since OFF and TOTA do not cooperate with other platforms, they do not

have outer payment rate as well. The outer payment is about 0.70 using DemCOM, while it is about 0.8 using RamCOM. Although RamCOM pay more to cooperative workers, it brings more completed cooperative requests to platforms. RamCOM only increases 10% of the outer payment rate, but it increases its acceptance ratio about 4 times larger than that of DemCOM. Thus, the incentive mechanism of RamCOM is better than that of DemCOM.

C. Efficiency

We evaluate the efficiency of our COM algorithms on the real datasets by comparing with OFF and TOTA. We report the experimental results in terms of response time and memory cost as shown in Tables V-VII.

1) Efficiency w.r.t Response Time: The average response time of our two COM algorithms is a little larger than that of TOTA and OFF. The largest increase is only 0.43ms. In the real world, users nearly cannot detect such increase of response time, which indicates that our COM algorithms are highly efficient.

2) Efficiency w.r.t Memory Cost: The memory costs of OFF, TOTA, DemCOM and RamCOM are almost the same. Since the memory cost has little change among different algorithms, and the largest memory cost is less than 100 MB, we only report it in the results tested on RD10 and RY10. Such small memory cost indicates that COM algorithms are cost efficient.

D. Scalability

We test the scalability by comparing our COM algorithms with TOTA on the synthetic datasets. Since OFF can never be achieved in the real world, we do not compare with it. The results are shown in Fig. 5.

1) Total revenue w.r.t $|R|$: Fig. 5(a) reports the results of total revenue obtained by the two platforms w.r.t the total number of requests $|R|$. We can see that the total revenue obtained from all the three algorithms increases with the increase of $|R|$. This is because more requests are completed when $|R|$ increases. Moreover, the total revenue increase of RamCOM is the largest, while the increase of TOTA is the smallest. With the increase of $|R|$, the number of inner crowd workers is not enough to serve the requests, thus a large number of requests are rejected by TOTA, when $|R|$ becomes much larger than $|W|$. However, DemCOM and RamCOM can ask the outer crowd workers to serve more requests, so their increased total revenue is larger. The acceptance ratio of RamCOM is larger than that of DemCOM, so the increase of RamCOM is more obvious.

2) Total revenue w.r.t $|W|$: Fig. 5(e) shows the results of total revenue obtained by the two platforms w.r.t $|W|$. When $|W| < 1,000$, the total revenues calculated by the three algorithms increase with the increase of $|W|$. When $|W| > 1,000$, the total revenue almost keeps same. This is because when $|W| < 1,000$, with the increases of W, more requests can be served. However, when $|W| > 1,000$, the number of requests is not enough. All the requests can be served by the inner crowd workers.

3) Total revenue w.r.t rad: Fig. 5(i) shows the results of total revenues obtained by the two platforms w.r.t rad. The total revenue calculated by RamCOM is the largest, while that calculated by DemCOM is only a little larger than the result obtained by TOTA. This is because the outer payment calculated by DemCOM is so small that only a small number of outer crowd workers would like to serve the cooperative requests. Moreover, the total revenue of the three algorithms increase slightly with the increase of rad. This is because with the increase of rad, each worker can serve more requests.

4) Average Response time w.r.t $|R|$: Fig. 5(b) shows the average response time of each request in the three algorithms w.r.t $|R|$. The response time increases with the increase of $|R|$ for each of these algorithms, since more requests should be checked and assigned by algorithms. Moreover, the response time of TOTA is the smallest, while those of our COM algorithms are larger. This is because, TOTA does not need to estimate the outer payment. However, the largest increase is smaller than 25ms, and the response time increase of the algorithms is almost linear with the exponential increase of $|R|$, which ensures the scalability of the algorithms.

5) Average Response time w.r.t $|W|$: Fig. 5(f) shows the average response time of each request in the three algorithms w.r.t $|W|$. The response time of each algorithm increases with the increase of $|W|$, because more workers should be checked when assigning each requests. TOTA achieves the fastest response time, while other algorithms incurs up to 50ms higher time cost than TOTA.

6) Average Response time w.r.t rad: Fig. 5(j) shows the average response time of each request in the three algorithms w.r.t rad. The response time of each algorithm keeps steady with the increase of rad. This indicates that rad has little influence on the calculation time of the algorithms. Moreover, with the increase of rad, the response time of RamCOM increases slightly. This is because when rad is larger, more outer crowd workers can serve each cooperative request, and the time cost of estimating the outer payment becomes higher.

7) Memory Cost w.r.t $|R|$, $|W|$ and rad: Figs. 5(c), 5(g) and 5(k) show the memory costs of the algorithms w.r.t $|R|$, $|W|$ and rad respectively. We can see that the memory cost of each algorithm increases with the increase of $|R|$ and $|W|$. This is because more memory is used to store the increased number of requests and workers. The memory costs of the algorithms keep steady with the increase of rad. Since the numbers of $|R|$ and $|W|$ are the same, no more memory is needed for storage and calculation.

8) Acceptance ratio w.r.t $|R|$: Fig. 5(d) shows the acceptance ratio of cooperative requests w.r.t $|R|$. TOTA has no cooperative requests and thus has no acceptance ratio. We can see that when $|R| \leq$ 20k, with the increase of $|R|$, the acceptance ratio of DemCOM and that of RamCOM increase. This is because with the increase of $|R|$, the inner crowd workers are not enough, and more requests are assigned to outer crowd workers. However, when $|R| >$ 20k, the acceptance ratio of DemCOM decreases. This is because too many requests are left to outer crowd workers, while the outer

payment of Algorithm 2 is too small. The number of requests successfully accepted by the outer crowd workers almost keeps same, leading to the acceptance ratio decrease of DemCOM.

9) Acceptance Probability w.r.t $|W|$: Fig. 5(h) shows the acceptance ratio change of cooperative requests with the increase of $|W|$. We can see that when $|W|$ is smaller than 1,000, the acceptance ratio increase with the increase of $|W|$, since the cooperative requests are left to more outer crowd workers, and are more possible to be accepted. When $|W|$ becomes larger than 1,000, the curves of acceptance ratio shake terribly since almost all requests are served by the inner crowd workers. Since few requests are left to the outer crowd workers, the acceptance ratio is hard to be stable.

10) Acceptance Probability w.r.t rad: Fig. 5(l) shows the acceptance ratio of cooperative requests w.r.t. different rad values. Clearly, when rad increases, the acceptance ratio of RamCOM increases, since requests are left to more outer crowd workers. Likewise, for DemCOM, when $rad \leq 1.5$, the acceptance ratio of cooperative requests increases as well. However, when $rad > 1.5$, the acceptance ratio of cooperative requests decreases. This is because fewer requests become cooperative requests when the inner crowd workers can serve more requests.

VI. Related Work

This section first surveys two main related topics in spatial crowdsourcing, task assignment and incentive mechanisms. Then we analyze their difference from our COM.

A. Two Main issues in Spatial Crowdsourcing

The task assignment studies how to allocate requests to suitable workers that can either maximize the total revenue or minimize the total cost, satisfying some spatial-temporal constraints. Task assignment approaches can be categorized into two types: offline and online matching. The offline matching supposes that the spatial-temporal information such as locations and arrival order of all requests and workers are pre-known. Ahuja et al. [11] propose Hungarian algorithm to calculate the exact solution. Alternatively, Kazemi et al. [8] reduce the offline matching to the maximum flow problem [11], and use Ford-Fulkerson algorithm to calculate the exact solution. However, the calculation complexity of the above exact solutions is extremely high due to its cubic relationship with the number of tasks and that of workers. To overcome this problem, greedy-based approximate solutions have been designed for efficient processing in offline matching [8]. The greedy-based approximation sacrifices the computation accuracy for fast processing.

In online matching, the information on requests or workers is not pre-known. There are two types of online matching, one-side matching and two-side matching. While one-side matching enables one-side vertices of the bipartite graph known to the other side, two-side matching keeps two parties blind to each other. Several one-side matching approaches have been proposed to maximize the total number of assignments [17] [18] [19]. Karp et al. [17] propose three algorithms, Greedy, RANDOM and RANKING. Among them, RANKING can reach a competitive ratio $1 - 1/e$ under adversarial model. Feldman et al. [18] proposes an algorithm with a ratio 0.67 under random model. Manshadi et al. [19] use a Monte Carlo sampling method to improve the ratio to 0.706. Approaches have been proposed to maximize the total revenue [20]. Aggarwal et al. [20] proposes a Perturbed-Greedy algorithm with a competitive ratio $1 - 1/e$ under adversarial model. In real applications, most task assignment problem requires two-side matching. Approaches [21] [10] have been proposed to maximize the total number of assignments using two-side matching. Tong et al. [10] applied an offline-guide-online method to obtain a ratio of 0.47. Huang et al. [21] extended the RANKING algorithm to obtain a competitive ratio of 0.554 under adversarial model. Research has been done to maximize the total revenue. In [9] Tong et al. proposed a Greedy-RT method that has a ratio of $\frac{1}{2e\ln\lceil U_{max}+1\rceil}$ under adversarial model. Approaches have been proposed to minimize the total cost [22] [23] [24] [12]. Kalyanasundaram et al. [22] proposed a Greedy-based algorithm, which has no bounded competitive ratio under adversarial model. Meyerson et al. [23] considered a randomized greedy algorithm using an HST structure to capture the location information of workers and requests, and reach a ratio of $O(\log^3 |W|)$, where $|W|$ is the number of workers. Based on [23], Bansal et al. [24] improved the ratio to $O(\log |W|)$ on 2-HST metric. Tong et al. [12] comprehensively compared the above algorithms, and concluded that the Greedy algorithm has only a $\frac{1}{k!}$ chance to reach the worst case, where $k = \min\{|T|, |W|\}$, and $|T|$ (resp. $|W|$) is the number of requests (resp. workers). Generally, Greedy obtains a better result than the other three algorithms, though its ratio is the worst under adversarial model.

The incentive mechanism problem determines the reward motivating workers to serve the requests. The reward has been determined according to the dynamic supply and demand in space and time [25] [26] [14]. While the platform determines the reward, the workers decide whether to accept it. In [25] and [26], authors use a Markov method to decide how much to pay for the workers. Tong et al. [14] use bipartite graph model to calculate the expect maximum pricing strategy. In [27] [28] the incentive mechanism has been studied to allow workers to choose their expected reward, and the platform to decide the rewards afterwards. Asghari et al. [27] studied the incentive mechanism for ride-sharing, while Zhang et al. [28] focused on the citizen sensing problem. There exist extensive studies on auctioning and incentives in competitive market. Hammond et.al. provided a good survey [29] on this topic. These studies research the allocation and incentive rules in an economic environment that is a collection of economic agents. The problem of auctioning and incentives is solved by using the game theory methods that consider the games between producers and agents from an economic view [30] [31]. However, these methods cannot be directly applied to our COM, since they do not consider the spatial and temporal constraints of users and those of workers. Moreover, the calculated results have not been guaranteed.

B. Relationship with COM

From the above illustration, all the existing task assignment studies focus on maximizing the total revenue or minimizing the cost of one single platform. The cooperation among different spatial crowdsourcing platforms has not investigated, which is the key of our COM problem. In COM, different platforms have competition and cooperation as well among each other. The task assignment among outer crowd workers cannot simply keep same as that for inner crowd workers. COM needs to involve an incentive mechanism for attracting outer crowd worker, which calculates the outer payment during the process of matching. Assigning a request to different workers would lead to different outer payment. Moreover, existing studies on incentive mechanism only consider the requests and workers on a single platform. They only focus on how to motive more workers to complete the requests and enable the platform to obtain more revenue. However, in the COM problem, the cooperative platforms wish the outer payment as much as possible, while the platform offering the request wishes the outer payment as little as possible. This requires new incentive mechanism. Reversely, the outer payment would directly influence the task assignment strategy of the COM problem. When a platform receives a request, it would decide whether to serve it using the inner crowd workers, or using the outer crowd workers, or just to reject it, according to the outer payment. Therefore, the COM problem needs an integration of novel task assignment and advanced incentive mechanism.

VII. Conclusion

In this paper, we propose a Cross Online Matching (COM), which enables a platform to "borrow" some unoccupied crowd workers from other platforms providing the same type of services to complete these requests. The COM integrates the task assignment and incentive mechanism of spatial crowdsourcing. We propose two algorithms, DemCOM and RamCom for COM. The competitive ratio of DemCOM is same as that of the greedy algorithm used for traditional online matching, while the competitive ratio of RamCOM can reach $\frac{1}{8e}$. Extensive experimental results over real and synthetic datasets verify the effectiveness and efficiency of our COM approaches.

In the future work, researchers can further consider the route of workers during the cross online matching process. Besides obtaining the high total revenue, the cooperation can be improved if the crowd workers can provide the service after short travel distances.

Acknowledgment

Yurong Cheng is supposed by the NSFC (Grant No. 61902023, U1811262) and the China Postdoctoral Science General Program Foundation (No. 2018M631358). Ye Yuan is supported by the NSFC (Grant No. 61932004, 61572119 and 61622202) and the Fundamental Research Funds for the Central Universities (Grant No. N181605012). Guoren Wang is supported by the NSFC (Grant No. 61672145 and 61732003). Lei Chen is supported by the Hong Kong RGC CRF Project C6030-18GF , the National Science Foundation of China (NSFC) under Grant No. 61729201, Science and Technology Planning Project of Guangdong Province, China, No. 2015B010110006, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Didi-HKUST joint research lab grant, Microsoft Research Asia Collaborative Research Grant and Wechat Research Grant.

References

[1] "Uber," https://www.uber.com/.
[2] "Didi," https://www.didiglobal.com/.
[3] "Meituan," http://waimai.meituan.com/.
[4] "Ele.me," https://www.ele.me/home/.
[5] "Fedex," https://www.fedex.com/en-us/home.html.
[6] "Sf express," http://www.sf-express.com/sg/en/.
[7] "Uber revenue and usage statistics (2018)," http://www.businessofapps.com/data/uber-statistics/#4.
[8] L. Kazemi and C. Shahabi, "Geocrowd: Enabling query answering with spatial crowdsourcing," in *SIGSPATIAL*, 2012.
[9] Y. Tong, J. She, B. Ding, L. Wang, and C. Lei, "Online mobile micro-task allocation in spatial crowdsourcing," in *ICDE*, 2016.
[10] Y. Tong, L. Wang, Z. Zhou, B. Ding, C. Lei, J. Ye, X. Ke, Y. Tong, L. Wang, and Z. Zhou, "Flexible online task assignment in real-time spatial data," *PVLDB*, vol. 10, no. 11, pp. 1334–1345, 2017.
[11] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network flows: theory, algorithms, and applications," *JORS*, vol. 45, no. 11, pp. 791–796, 1993.
[12] Y. Tong, J. She, B. Ding, C. Lei, T. Wo, and X. Ke, "Online minimum matching in real-time spatial data: experiments and analysis," *PVLDB*, vol. 9, no. 12, pp. 1053–1064, 2016.
[13] "Proofs," https://github.com/cyrbit/cyrbit.github.io/blob/master/apendix.pdf.
[14] Y. Tong, L. Wang, Z. Zhou, L. Chen, B. Du, and J. Ye, "Dynamic pricing in spatial crowdsourcing: A matching-based approach," in *SIGMOD*, 2018, pp. 773–788.
[15] "Gaia of didi," https://outreach.didichuxing.com/research/opendata/en/.
[16] "Dataset," https://pan.baidu.com/s/1q7ThIY2jIRw94azITUBaAw.
[17] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, "An optimal algorithm for on-line bipartite matching," in *STOC*, 1990.
[18] J. Feldman, A. Mehta, V. Mirrokni, and S. Muthukrishnan, "Online stochastic matching: Beating 1-1/e," *FOCS*, vol. 157, no. 1, 2009.
[19] V. H. Manshadi and A. Saberi, "Online stochastic matching: Online actions based on offline statistics," *Mathematics of Operations Research*, vol. 37, no. 4, pp. 1285–1294, 2011.
[20] G. Aggarwal, G. Goel, C. Karande, and A. Mehta, "Online vertex-weighted bipartite matching and single-bid budgeted allocations," 2011.
[21] Z. Huang, K. Ning, Z. G. Tang, X. Wu, and Z. Xue, "How to match when all vertices arrive online," 2018.
[22] B. Kalyanasundaram and K. Pruhs, "On-line weighted matching," 1991.
[23] A. Meyerson, A. Nanavati, and L. J. Poplawski, "Randomized online algorithms for minimum metric bipartite matching," in *SODA*, 2006.
[24] N. Bansal, N. Buchbinder, A. Gupta, and J. Naor, "A randomized o(log2k)-competitive algorithm for metric bipartite matching," *Algorithmica*, vol. 68, no. 2, pp. 390–403, 2014.
[25] S. Banerjee, D. Freund, and T. Lykouris, "Pricing and optimization in shared vehicle systems: An approximation framework," in *EC*, 2017.
[26] M. Chen, W. Shen, P. Tang, and S. Zuo, "Optimal vehicle dispatching for ride-sharing platforms via dynamic pricing," in *WWW*, 2018.
[27] M. Asghari, D. Deng, C. Shahabi, U. Demiryurek, and Y. Li, "Price-aware real-time ride-sharing at scale: an auction-based approach," in *SIGSPATIAL*, 2016.
[28] X. Zhang, Z. Yang, Z. Zhou, H. Cai, L. Chen, and X. Li, "Free market of crowdsourcing: Incentive mechanism design for mobile sensing," *TPDS*, vol. 25, no. 12, pp. 3190–3200, 2014.
[29] P. Hammond, *Allocation Mechanisms, Incentives, and Endemic Institutional Externalities*. Springer, 2019.
[30] U. Blum and F. Kalus, "Auctioning public financial support incentives," *International Journal of Technology Management*, vol. 26, no. 2-4, pp. 270–276, 2003.
[31] L. Hurwicz, "The design of mechanisms for resource allocation," *American Economic Review*, vol. 63, no. 2, pp. 1–30, 1973.