

# Online delivery route recommendation in spatial crowdsourcing

Dezhi Sun<sup>1</sup> · Ke Xu<sup>1</sup> · Hao Cheng<sup>1</sup> · Yuanyuan Zhang<sup>2</sup> ·  
Tianshu Song<sup>1</sup> · Rui Liu<sup>1</sup> · Yi Xu<sup>1</sup>

Received: 25 February 2018 / Revised: 30 March 2018 / Accepted: 5 April 2018 /  
Published online: 28 May 2018  
© Springer Science+Business Media, LLC, part of Springer Nature 2018

**Abstract** With the emergence of many crowdsourcing platforms, crowdsourcing has gained much attention. Spatial crowdsourcing is a rapidly developing extension of the traditional crowdsourcing, and its goal is to organize workers to perform spatial tasks. Route recommendation is an important concern in spatial crowdsourcing. In this paper, we define a novel problem called the Online Delivery Route Recommendation (OnlineDRR) problem, in which the income of a single worker is maximized under online scenarios. It is proved that no deterministic online algorithm for this problem has a constant competitive ratio. We

---

This article belongs to the Topical Collection: *Special Issue on Big Data Management and Intelligent Analytics*  
Guest Editors: Junping Du, Panos Kalnis, Wenling Li, and Shuo Shang

---

✉ Rui Liu  
lr@buaa.edu.cn

Dezhi Sun  
buaasun@buaa.edu.cn

Ke Xu  
kexu@buaa.edu.cn

Hao Cheng  
chengh@buaa.edu.cn

Yuanyuan Zhang  
zhangyuanyuan@datang.com

Tianshu Song  
songts@buaa.edu.cn

Yi Xu  
xuy@buaa.edu.cn

<sup>1</sup> State Key Laboratory of Software Development Environment, School of Computer Science and Engineering, Beihang University, Beijing, China

<sup>2</sup> China Academy of Telecommunications Technology and Datang Telecom Technology, Industry Group, Beijing, China

propose an algorithm to balance three influence factors on a worker's choice in terms of which task to undertake next. In order to overcome its drawbacks resulting from the dynamic nature of tasks, we devise an extended version which attaches gradually increased importance to the destination of the worker over time. Extensive experiments are conducted on both synthetic and real-world datasets and the results prove the algorithms proposed in this paper are effective and efficient.

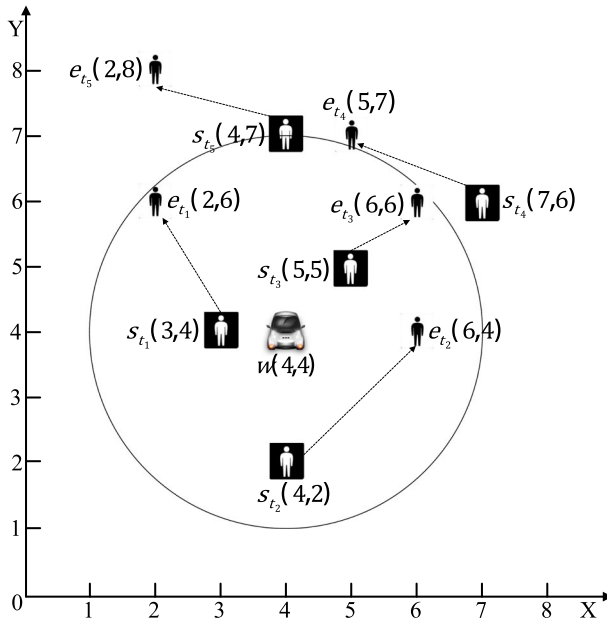
**Keywords** Route recommendation · Spatial crowdsourcing · Online algorithms

## 1 Introduction

In recent years, crowdsourcing has gained much attention, many crowdsourcing platforms have sprung up, such as Amazon Mechanical Turk (MTurk) [1]. As an extension of the traditional crowdsourcing, spatial crowdsourcing (a.k.a. mobile crowdsourcing) is developing rapidly, attributing to the widespread use of smartphones. Spatial crowdsourcing is characterized by organizing crowd workers (*workers* for short) on a mobile platform to perform spatial tasks by physically visiting the task-specific locations [44]. Some representative examples of spatial crowdsourcing include citizen sensing (i.e. organizing participants to collect sensor data from their surrounding environment using mobile devices), location based services. Many studies have investigated the problem of spatial data management [17, 18, 23, 38, 53, 56, 57].

On spatial crowdsourcing platforms, there is a phenomenon that some workers are efficient in terms of the money they make, while the others aren't. In other words, some workers can make more money during the same time period than others. Based on this observation, route recommendation or task recommendation in spatial crowdsourcing has been studied in [2, 5, 10, 16, 24, 28, 55, 60]. However, [2, 5] only study the offline scenario where all tasks are known in advance. Considering that in real-world applications, tasks on spatial crowdsourcing platforms are often released dynamically. Different from [12], tasks in spatial crowdsourcing are changed in both time and space. We study route recommendation under online scenarios, where future tasks are not known in advance. In addition, the task in [5, 16] is of a visit type, which means the task can be finished by a worker physically visiting the task-specific location. But in this paper, we tackle those delivery-type tasks that have a origin and a destination. An example of delivery tasks is ride requests, which ask a driver to pick up someone at some place and drop off the passengers at another place. If not explicitly specified, we use the word *task* to refer to a delivery-type task in this paper.

*Example 1* Figure 1 shows an example of a worker  $w$  and a set of tasks  $T = \{t_1, t_2, \dots, t_5\}$ . Each task is associated with two locations, its origin  $s_{t_i}$  and destination  $e_{t_i}$  and the time when each task is released is shown in Table 1. We assume that the worker  $w$  moves a length unit per hour, and the payoff of each task is directly proportional to the Euclidean distance between its origin and destination. Accordingly, the more time  $w$  spends on accomplishing tasks from their origins to destinations (valid work time), the more money  $w$  will make. In this example,  $w$  plans to start working at location (4,4) from 8:00 and arrive at (2,7) before 18:30, and he/she ignores those tasks whose origins are 3 length units away. One possible route is  $R_1^w = \langle t_1, t_5 \rangle$ . In other words,  $w$  first chooses the task  $t_1$ , so he/she heads for  $s_{t_1}$  and then  $e_{t_1}$ , and after  $w$  finishes  $t_1$  at about 11:14,  $w$  has to wait at location (2, 6) until  $t_5$  is released. Then  $w$  chooses to handle the task  $t_5$  and leaves for  $s_{t_5}$  and  $e_{t_5}$  successively.



**Figure 1** An example of  $T$  and  $w$

After that,  $w$  waits at  $(2,8)$  but there is no suitable task for  $w$ .  $w$  starts to head for his/her destination at 17:30 because if he/she keeps waiting at  $(2,8)$ , he/she won't be able to get to  $(2,7)$  on time. So the valid work time of  $w$  is the sum of the time of moving from  $s_{t_1}$  to  $e_{t_1}$  and from  $s_{t_5}$  to  $e_{t_5}$ , which amounts to  $2\sqrt{5}$  ( $\approx 4.47$ ) hours. Note that the deadline constraints of  $t_1$  and  $t_5$  are both satisfied in this case. However, the optimal route  $w$  can take is  $R_{opt}^w = (t_3, t_4, t_5)$ . In the optimal case,  $w$  successively visits  $s_{t_3}, e_{t_3}, s_{t_4}, e_{t_4}, s_{t_5}, e_{t_5}$  and still has enough time to travel to his/her destination. Along this route, the valid work time of  $w$  is  $\sqrt{2} + 2\sqrt{5}$  ( $\approx 5.89$ ) hours and all the tasks in  $R_{opt}^w$  can be finished before their respective deadlines.

As discussed above, different strategies may vary greatly in regard to efficiency. The question then arises: is it possible to recommend a worker some tasks in order to maximize his/her income? The challenges of route recommendation in spatial crowdsourcing are two-fold. First, some experienced workers can often make a better schedule so as to make better use of his/her working time to finish more tasks. However, such experience is gained from long-term observation and practice and can hardly be modeled. Secondly, tasks on most spatial crowdsourcing platforms can appear dynamically anytime and anywhere,

**Table 1** Release Times, Deadlines and Rewards of Tasks

Task	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$
Release time	7:30	7:30	7:30	10:30	12:30
Deadline	11:30	13:00	11:00	14:30	17:30
Reward	$\sqrt{5}$	$2\sqrt{2}$	$\sqrt{2}$	$\sqrt{5}$	$\sqrt{5}$

which would further make route recommendation harder. For example, if we recommend a task to a worker and the worker accepts it, and after he/she finishes this task, there are no available task in his/her vicinity, the worker would have to travel a long way to reach his/her next task. So in task recommendation, it's essential to take the future into consideration due to the uncertainty of the task's release time and location.

In this paper, we propose a method to recommend an appropriate route to a single worker in spatial crowdsourcing while taking into account: (a) the distance between the worker's location and the origin of each feasible task, (b) the distance between the origin and destination of each feasible task, and (c) the future demand originating from the destination of each task. These three aspects are fused using three parameters according to the importance of each concern, in order to recommend a best route to a single worker in the dynamics of tasks. In a basic version of this method, we tune the parameters based on the historical data. However, due to the lack of historical data and the dynamic nature of tasks, the basic version may not achieve good results under some circumstances. We further propose an algorithm which considers the destination based on the actual situation of worker. This algorithm proves to be effective to achieve better results than other heuristic algorithms as well as robust to cope with the dynamics of spatial crowdsourcing.

To the best of our knowledge, our work is the first to make delivery route recommendation for a single worker in spatial crowdsourcing under online scenarios. Extensive experiments are conducted on a synthesized dataset and a large scale New York City dataset. The experiments show that our algorithm is scalable and efficient, capable of processing thousands of tasks quickly. In summary, the contributions of this paper include:

- We define the Online Delivery Route Recommendation (OnlineDRR) problem to accommodate the real-world need to make a plan for a worker in spatial crowdsourcing in order to maximize the worker's income. We prove that the offline version of this problem is NP-hard and no deterministic online algorithm for the OnlineDRR problem can achieve a constant competitive ratio.
- We propose an effective algorithm to consider three influence factors on a worker's choice in terms of which task to undertake next. The parameters are tuned using historical data in the basic version of this algorithm. We also devise an advanced version to overcome the drawbacks of the basic version which may not achieve good results under some circumstances. The advanced version of this algorithm considers the destination of the worker in order to align the recommended route with his/her destination.
- We conduct extensive experiments on both synthetic and real datasets. The experimental results verify the effectiveness and efficiency of our proposed algorithms.

The rest of this paper is organized as follows. We define our problem in Section 2, and propose two heuristics and our algorithms in Section 3. Section 4 presents the experimental evaluation. We discuss the related work in Section 5 and finally conclude the paper in Section 6.

## 2 Preliminaries

In this section, we define some terminologies and formally give the definition of the OnlineDRR problem. Then, we prove the offline version of this problem is NP-hard and there is no deterministic online algorithm for the OnlineDRR problem that has a good competitive ratio.

## 2.1 Problem definitions

**Definition 1** (Task) A task  $t$ , denoted as  $\langle s_t, e_t, a_t, d_t, r_t \rangle$ , starting from the location  $s_t$ , and ending at the location  $e_t$ , with a reward of  $r_t$ , is released on the platform at time  $a_t$  and can only be finished by a worker before the deadline  $d_t$ , otherwise it will disappear from the platform thereafter.

**Definition 2** (Worker) A worker  $w$ , who arrives at the platform with initial location  $s_w$  at time  $a_w$  and plans to get to the location  $e_w$  before the deadline  $d_w$ , is denoted as  $\langle s_w, e_w, a_w, d_w, r_w \rangle$ .  $r_w$  is the radius of the worker's circular service area with his/her location as its center. Specifically, a worker  $w$  is only eligible to undertake those tasks originating from his/her service area.

**Definition 3** (Route) Given a worker  $w$  and a set of tasks  $T$ , a route  $R^w = \langle t_1, t_2, \dots, t_n \rangle$  is an ordered sequence of tasks if  $n < |T|$ ,  $t_i \in T$ ,  $t_i \neq t_j$  for  $i \neq j$ .

The time that  $w$  arrives at each  $t_i$  in  $R^w$  is hard to formulated due to the online nature of tasks. When  $w$  finishes a task  $t$  and there is no available tasks in  $w$ 's vicinity, he/she can stay at  $e_t$  until the next available task is released, or he/she can roam around. Even if there is some other available tasks around  $w$  after he/she finishes the last task,  $w$  can wait for a more suitable task to show up. Different algorithms may have different strategies, so we don't give the definition of the arrival time of a task.

A route  $R^w$  is *valid* for a worker  $w$ , if and only if  $w$  can finish all the tasks in  $R^w$  in order by every time reaching the starting point of a task and then the end point of the task before heading for the next task without overrunning his/her time budget  $d_w - a_w$ , i.e., for each task  $t_i$  in  $R^w$ ,  $f(t_i) \leq d_i$ , where  $f(t)$  represents the finish time of the task  $t$ . We use  $c(a, b)$  to represent the travel cost for a worker from the location  $a$  to  $b$ . For simplicity and without loss of generality, we assume that  $c(a, b)$  equals the Euclidean distance between  $a$  and  $b$ .

The total income of  $w$  along a valid route  $R^w$ , represented by  $I(R^w)$ , is defined as the sum of  $r_{t_i}$  for each  $t_i$  in  $R^w$ , i.e.,  $I(R^w) = \sum_{t_i \in R^w} r_{t_i}$ .

**Definition 4** (Online Delivery Route Recommendation Problem) Given a worker  $w$  and a set of tasks  $T$ , of which all the tasks are released in an online manner, the goal of the Online Delivery Route Recommendation (OnlineDRR) problem is to find a valid route  $R^w$  such that the total income of  $w$  is maximized.

## 2.2 Hardness of the problem

In this subsection, we prove the Offline Delivery Route Recommendation (OfflineDRR) problem is NP-hard. First, we give the definition of the OfflineDRR Problem.

**Definition 5** (Offline Delivery Route Recommendation Problem) Given a worker  $w$  and a set of tasks  $T$ , with the spatiotemporal information of all tasks in  $T$  is known a priori, the goal of the Offline Delivery Route Recommendation (OfflineDRR) problem is to find a valid route  $R^w$  such that the total income of  $w$  is maximized.

We prove that the OfflineDRR problem is NP-hard by reducing a variant of orienteering problem [8] to it. This variant is called Directed Orienteering Problem with Time Windows and Service Time (DOP-TW-ST-R) and is defined as follows.

**Definition 6** (Directed Orienteering Problem with Time Windows, Service Time and Range) Given a directed graph with  $n$  nodes, which are numbered from 1 to  $n$  and each of which has a score  $s_i \geq 0$  ( $s_1 = s_n = 0$ ), a time window  $[t_{1i}, t_{2i}]$ , denoting the earliest and latest time that the node  $i$  can be served, and a service time  $st_i$ , the DOP-TW-ST-R is to find a route, starting from the node 1 and ending at the node  $n$ , such that the total score is maximized and the total length is no greater than  $T_{MAX}$  and the distance between each adjacent node pair is no greater than the range  $r_{MAX}$ .

**Lemma 1** *The DOP-TW-ST-R is NP-hard.*

*Proof* We prove the NP-hardness of the DOP-TW-ST-R by reducing the orienteering problem. Given  $n$  nodes numbered from 1 to  $n$ , each of which has a score  $s_i \geq 0$  ( $s_1 = s_n = 0$ ), the orienteering problem is to find a route from the node 1 to the node  $n$  such that the total score is maximized and the total length is no greater than  $T_{MAX}$ . We consider a special case of the DOP-TW-ST-R, where for any two nodes  $i, j$ , the length of the edge  $(i, j)$  equals that of  $(j, i)$ , and the time window for each node is  $[0, T_{MAX}]$ , and the service time of each node is 0, and the range  $r_{MAX}$  is the maximum distance between any two nodes. Then the orienteering problem, which is NP-hard [8], is equivalent to the aforementioned special case of the DOP-TW-ST-R, so we can reduce the orienteering problem to the special case of the DOP-TW-ST-R. Therefore, the DOP-TW-ST-R is NP-hard.  $\square$

**Theorem 1** *The OfflineDRR problem is NP-hard.*

*Proof* For an instance  $\mathcal{I}$  of the DOP-TW-ST-R, we map the first node to the worker's start location, the last node to the worker's destination, and the other  $n - 2$  nodes to tasks. The service radius of the worker is  $r_{MAX}$  and the deadline of the worker is  $T_{MAX}$ . For a node  $i$  ( $1 \leq i \leq n$ ), we map it to a task  $t_i$ . Let the reward of  $t_i$  be the score of the node  $i$  in  $\mathcal{I}$ , the travel cost from the task's origin to its destination  $c(s_{t_i}, e_{t_i})$  be the service time of the node  $i$ ,  $c(s_w, s_{t_i})$  be the length of the edge  $(1, i)$ ,  $c(e_{t_i}, e_w)$  be the length of the edge  $(i, n)$ , the release time and deadline of  $t_i$  be  $t_{1i}, t_{2i} + st_i$  in  $\mathcal{I}$ , respectively. For any two node  $i, j$  ( $1 \leq i, j \leq n$ ), let  $c(e_{t_i}, s_{t_j})$  be the length of the edge  $(i, j)$ . Now we get an instance  $\mathcal{I}'$  of the OfflineDRR problem. As long as there exists a route in  $\mathcal{I}'$  with total income as  $T$ , there must be a route in  $\mathcal{I}$  collecting a total score of  $T$ , and vice versa. So the DOP-TW-ST-R can be reduced to the OfflineDRR problem. Since the DOP-TW-ST-R is NP-hard, the OfflineDRR problem is also NP-hard.  $\square$

In this paper, we study the OnlineDRR problem, in which the algorithms have no knowledge of the prospective tasks. The performance of online algorithms is usually measured by competitive ratios [15]. For our OnlineDRR problem, an algorithm ALG is called *c-competitive*, if  $ALG(\sigma) \leq c \cdot OPT(\sigma)$  holds for any input  $\sigma$ , where  $ALG(\sigma)$  and  $OPT(\sigma)$  denote the worker's total incomes gained by ALG and optimal offline algorithm OPT,

respectively. The competitive ratio of ALG is the infimum over all  $c$  such that ALG is  $c$ -competitive. We now show that no deterministic online algorithm can achieve a good competitive ratio for the OnlineDRR problem.

**Theorem 2** *No deterministic online algorithm for the OnlineDRR problem has a constant competitive ratio.*

*Proof* To prove there does not exist a deterministic online algorithm that is  $c$ -competitive, it suffices to find an input which can make the competitive ratio of the algorithm as low as possible. Without loss of generality, we only consider locations on the positive axis and assume that the worker moves a length unit per time unit. At time 0, a worker  $w = \langle 0, n, 0, n + 1, n \rangle$  appears at the location 0 and needs to reach the location  $n$  before time  $n + 1$ .

The adversary releases a task  $t_1 = \langle \delta, 1, 0, 1, 1 - \delta \rangle$  at time 0, and the task originates from position  $\delta$  and should be finished at position 1 before time 1. In terms of  $w$ 's position at time 1 (say  $x$ ), there are two cases: (1)  $x = 0$ , or (2)  $x > 0$ . Then the adversary can release another task accordingly to make the competitive ratio unboundedly small.

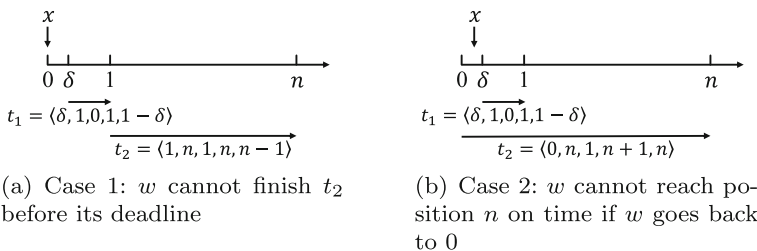
**Case 1 ( $x = 0$ )** In this case, the adversary will release another task  $t_2 = \langle 1, n, 1, n, n - 1 \rangle$  (see Figure 2a). Since  $w$  cannot arrive at position  $n$  before time  $n$ ,  $w$  is not eligible to undertake  $t_2$ . So  $w$  will end up finishing no tasks, i.e.,  $I(R_{\text{ALG}}^w) = 0$ , where  $R_{\text{ALG}}^w$  represents the  $w$ 's route given by ALG. However, the optimal offline algorithm OPT, which knows both tasks in advance, will give a better route  $R_{\text{OPT}}^w = \langle t_1, t_2 \rangle$ , and hence  $w$  can finish both tasks, and  $I(R_{\text{OPT}}^w) = n - \delta$ . In this case, the ratio between these two results is

$$\frac{I(R_{\text{ALG}}^w)}{I(R_{\text{OPT}}^w)} = \frac{0}{n - \delta} = 0.$$

**Case 2 ( $x > 0$ )** In this case, the adversary will release a task  $t_2 = \langle 0, n, 1, n + 1, n \rangle$  (see Figure 2b). If  $w$  accepts  $t_2$ , then  $w$  wouldn't be able to reach position  $n$  before his/her deadline. So  $w$  can only finish at most one task, i.e.  $t_1$ , and  $I(R_{\text{ALG}}^w) \leq 1 - \delta$ . However, with both tasks known in advance, OPT will recommend  $w$  to wait at position 0 until  $t_2$  are released and thus the total income achieved by OPT is  $I(R_{\text{OPT}}^w) = n$ . Since  $n$  can be arbitrarily large, the ratio between them can be arbitrarily small, i.e.,

$$\frac{I(R_{\text{ALG}}^w)}{I(R_{\text{OPT}}^w)} = \frac{1 - \delta}{n} \rightarrow 0.$$

□



**Figure 2** Different cases of releasing  $t_2$  in the adversarial analysis

### 3 Algorithms

In this section, we present a set of solutions to the online delivery route recommendation problem.

#### 3.1 The baseline approach

As defined in the OnlineDRR problem, the goal is to maximize the total income of a worker  $w$ . A natural way is to find the longest task every time the worker needs to make a choice. When there exist some tasks with the same length, the worker selects the nearest task. Based on this strategy, the workers will gain the maximum income at one time. However, this method does not consider the distance between the worker and the task, so it may take more time for a worker to go to the starting point of a task and he/she would have less time to finish the remaining tasks. Notice that for any task the worker chooses, he/she must have time to go to his destination after finishing this task, or he/she will not accept that task.

---

#### Algorithm 1: Baseline Approach

---

```

input :  $w, T, currentTime$ 
output:  $R^w$ 
 $idleTime \leftarrow 0$ ;
 $S_{candidate} \leftarrow \emptyset$ ;
while  $currentTime < d_w$  do
  if  $idleTime + c(w, e_w) \geq d_w$  then
     $\perp$  break;
  update  $T$ ;
  update  $currentTime$ ;
  if  $|T| > 0$  and  $currentTime \geq idleTime$  then
    foreach  $t \in T$  do
      if  $idleTime \geq a_t$  and  $idleTime + c(w, s_t) \leq d_t$  and
         $idleTime + c(w, s_t) + c(s_t, e_t) + c(e_t, e_w) \leq d_w$  then
           $\perp$   $S_{candidate} \leftarrow S_{candidate} \cup t$ ;
      if  $S_{candidate} = \emptyset$  then
         $\perp$   $idleTime \leftarrow currentTime$ ;
         $\perp$  continue;
      Find the task  $t^* \in S_{candidate}$  with maximum distance;
       $R^w \leftarrow R^w \cup t^*$ ;
       $S_{candidate} \leftarrow \emptyset$ ;
       $idleTime \leftarrow idleTime + c(w, s_{t^*}) + c(s_{t^*}, e_{t^*})$ ;
    if  $currentTime > idleTime$  then
       $\perp$   $idleTime \leftarrow currentTime$ ;
return  $R^w$ 

```

---



Algorithm 1 illustrates the procedure of the baseline approach for the OnlineDRR problem. We use *currentTime* to represent the current system time. In lines 1-2, we represent the time when the worker is ready to take a task as *idleTime*, and the valid candidate task sets for choosing as *S<sub>candidate</sub>*. In lines 3-17, we continue to find the valid tasks one by one before the deadline of the worker. Specifically, we check every task whether the worker could accept it which means he would have enough time to go back to his destination if he accepts this task. In lines 3-4, we check the deadline constraint of the worker. If there is no enough time for the work to go back to his destination, the worker would stop accepting the task and go to his destination. In lines 6-7, we keep on updating the task set *T* and the system time *currentTime*. In line 12, if the set *S<sub>candidate</sub>* is empty which means there is no suitable task at that time, the worker would keep on waiting until there is a new suitable task. In line 15, if the set *S<sub>candidate</sub>* is not empty, then the worker would select the one with maximum distance. Then we update set *R<sup>w</sup>* and the *idleTime* of the worker. In line 19, when *currentTime* is greater than *idleTime*, that means there is no suitable tasks at that time and the worker is waiting for tasks, we should set *idleTime* to the current system time. The algorithm will end when *currentTime* exceeds the deadline of the worker.

*Example 2* Suppose we have the scenario shown in Example 1, the worker *w* starts to work at location (4,4) and there is a set of tasks  $T = \{t_1, t_2, \dots, t_5\}$  with different release times. If the worker *w* adopts the baseline approach, he/she would choose the longest task one by one in a greedy way. First, the worker chooses task  $t_2$  which has the longest distance. Then when *w* arrives at the destination of task  $t_2$ , there is only one task remaining which is  $t_4$ , but the worker does not have enough time to finish it before its deadline. So he/she has to wait until there is a suitable task he could finish. Unfortunately, no task could be done before the deadline. So he/she has to head for his/her destination in order to get to the destination on time. Consequently, the worker *w* can only finish one task  $t_2$ . In the end, the recommended route for *w* is  $t_2$ .

### 3.2 Greedy algorithm

As illustrated in the baseline approach, it does not consider the distance between the worker *w* and remaining tasks. The greedy algorithm aims at reducing useless movements between the worker and tasks. It finds the nearest task in the service range of the worker which refers to the nearest neighbor query [11, 33].

Algorithm 2 illustrates the procedure of the greedy algorithm. The execution process is similar to the baseline approach. In lines 3-17, we try to find the valid tasks one by one before the deadline of the worker. In lines 3-4, we check the deadline constraint of the worker. In lines 6-7, we keep on updating the task set *T* and the system time *currentTime*. In lines 8-11, we try to set up the set *S<sub>candidate</sub>* which contains tasks satisfy the time constraints. In line 12, if the set *S<sub>candidate</sub>* is empty which means there is no suitable task at that time, he would keep on waiting until there is a new suitable task. In line 15, if the set *S<sub>candidate</sub>* is not empty, the worker will choose the task nearest to he/she in the set *S<sub>candidate</sub>*. The algorithm will end when *currentTime* exceeds the deadline of the worker.

**Algorithm 2:** Greedy Algorithm

---

```

input :  $w, T, currentTime$ 
output:  $R^w$ 
 $idleTime \leftarrow 0$ ;
 $S_{candidate} \leftarrow \emptyset$ ;
while  $currentTime < d_w$  do
  if  $idleTime + c(w, e_w) \geq d_w$  then
     $\perp$  break;
  update  $T$ ;
  update  $currentTime$ ;
  if  $|T| > 0$  and  $currentTime \geq idleTime$  then
    foreach  $t \in T$  do
      if  $idleTime \geq a_t$  and  $idleTime + c(w, s_t) \leq d_t$  and
         $idleTime + c(w, s_t) + c(s_t, e_t) + c(e_t, e_w) \leq d_w$  then
           $\perp$   $S_{candidate} \leftarrow S_{candidate} \cup t$ ;
      if  $S_{candidate} = \emptyset$  then
         $\perp$   $idleTime \leftarrow currentTime$ ;
         $\perp$  continue;
      Find the task  $t^* \in S_{candidate}$  nearest to worker  $w$ ;
       $R^w \leftarrow R^w \cup t^*$ ;
       $S_{candidate} \leftarrow \emptyset$ ;
       $idleTime \leftarrow idleTime + c(w, s_{t^*}) + c(s_{t^*}, e_{t^*})$ ;
    if  $currentTime > idleTime$  then
       $\perp$   $idleTime \leftarrow currentTime$ ;
return  $R^w$ 

```

---

*Example 3* We still consider the scenario in Example 1, the worker  $w$  starts to work at location (4,4) and there is a set of tasks  $T = \{t_1, t_2, \dots, t_5\}$  with different release times. If the worker  $w$  uses the greedy algorithm, he/she would choose the nearest order one by one. First,  $w$  would choose task  $t_1$  which is the nearest from he/she. Then when he/she arrives at the destination of task  $t_1$ , there is no available task in his/her service area. He/she needs to wait until  $t_5$  is released, and then finishes  $t_5$ . There is no more tasks after  $w$  finishes task  $t_5$ , so this process is finished. In the end, the recommended route for  $w$  is  $t_1 \rightarrow t_5$ .

### 3.3 Prediction based route recommendation algorithm

The baseline approach selects the longest task every time, but it does not consider the distance between the worker and tasks. The greedy algorithm chooses the nearest task instead, but it does not consider the reward of every task. In the same time, they do not consider the future demand around the destination of each task. In this paper, we propose a prediction

based route recommendation algorithm. We take into account these three factors: (a) the distance between the worker's location and the origin of each feasible task, (b) the distance between the origin and destination of each feasible task, and (c) the future demand originating from the destination of each task. Similar to [46], we define the unit original task demand around the destination of each task as  $UOTD(e_{t_i})$ . In the online scenario, the tasks can not be known in advance but we can use the state-of-the-art [46] to predict the unit original task demand. UOTD is useful for the worker to have more knowledge about the demand around the destination of each task. If a worker goes to somewhere with no task, he/she would have to wait or go to somewhere else. Obviously, this obstructs the worker from making more money. So if a worker goes to a place with many potential tasks in vicinity, he/she would have more chance to choose a suitable task in order to get more rewards. In this paper we take into account these three factors mentioned above so as to recommend better route for the worker. We define online route recommendation score function  $f(w, t_i)$  as follows.

$$f(w, t_i) = \alpha \left( 1 - \frac{c(w, s_{t_i})}{r_w} \right) + \beta \frac{c(s_{t_i}, e_{t_i})}{\widehat{MaxD}} + \gamma \frac{UOTD(e_{t_i})}{N} \quad (1)$$

where  $\alpha, \beta, \gamma \in (0,1)$  are parameters to balance these three factors: the distance between the worker's location and the origin of each feasible task, the distance between the origin and destination of each feasible task, and the future demand originating from the destination of each task. The parameters should satisfy  $\alpha + \beta + \gamma = 1$ ;  $c(w, s_{t_i})$  represents the spatial distance between worker and the start position of task and is normalized by  $r_w$  which represents the service range of the worker  $w$ ;  $c(s_{t_i}, e_{t_i})$  represents the spatial distance of task  $t_i$  and is normalized by  $\widehat{MaxD}$  which represents the maximum distance of all tasks. Similarly,  $UOTD(e_{t_i})$  represents the unit original task demand around the destination of task  $t_i$  and it is normalized by  $N$  which represents the sum of all tasks.

Algorithm 3 illustrates the procedure of the prediction based route recommendation algorithm for the OnlineDRR problem. In lines 1-2, we represent the time when the worker is ready to take a task as  $idleTime$ , and the valid candidate task sets for choosing as  $S_{candidate}$ . In lines 3-23, we continue to find the valid task one by one. In lines 3-4, we check the deadline constraint of the worker. In lines 10-11, We check every task whether the worker would have enough time to go back to his destination if he accepts this task, and we add valid tasks into set  $S_{candidate}$ . If the set  $S_{candidate}$  is empty, the worker would keep on waiting until there is new suitable task. In lines 15-17, for each task  $t$  in set  $S_{candidate}$ , we first compute the unit original task demand around the destination of each task  $UOTD(e_{t_i})$ , then we compute the online route recommendation score for each valid task. In lines 18-19, we put the task with highest score value into set  $R^w$ . Then we initialize  $S_{candidate}$  again, update  $idleTime$ . In lines 23-24, when  $currentTime$  is greater than  $idleTime$ , the worker is waiting for tasks and we set  $idleTime$  to the current system time. The algorithm will end when  $currentTime$  exceeds the deadline of the worker.

**Algorithm 3:** Prediction Based Route Recommendation Algorithm

---

```

input :  $w, T, currentTime$ 
output:  $R^w$ 
 $idleTime \leftarrow 0$ ;
 $S_{candidate} \leftarrow \emptyset$ ;
while  $currentTime < d_w$  do
  if  $idleTime + c(w, e_w) \geq d_w$  then
     $\perp$  break;
  update  $T$ ;
  update  $currentTime$ ;
  if  $|T| > 0$  and  $currentTime \geq idleTime$  then
    foreach  $t \in T$  do
      if  $idleTime \geq a_t$  and  $idleTime + c(w, s_t) \leq d_t$  and
         $idleTime + c(w, s_t) + c(s_t, e_t) + c(e_t, e_w) \leq d_w$  then
           $\perp$   $S_{candidate} \leftarrow S_{candidate} \cup t$ ;
    if  $S_{candidate} = \emptyset$  then
       $idleTime \leftarrow currentTime$ ;
       $\perp$  continue;
    foreach  $t_i \in S_{candidate}$  do
      compute  $UOTD(e_{t_i})$ ;
       $f(w, t_i) \leftarrow \alpha(1 - \frac{c(w, s_{t_i})}{R(w)}) + \beta \frac{c(s_{t_i}, e_{t_i})}{MaxD} + \gamma \frac{UOTD(e_{t_i})}{N}$ ;
     $t^* \leftarrow argmax_{t_i} f(w, t_i)$ ;
     $R^w \leftarrow R^w \cup t^*$ ;
     $S_{candidate} \leftarrow \emptyset$ ;
     $idleTime \leftarrow idleTime + c(w, s_{t^*}) + c(s_{t^*}, e_{t^*})$ ;
  if  $currentTime > idleTime$  then
     $\perp$   $idleTime \leftarrow currentTime$ ;
return  $R^w$ 

```

---

*Example 4* We still consider the scenario shown in Example 1, the worker  $w$  starts to work at location (4,4) and there is a set of tasks  $T = \{t_1, t_2, \dots, t_5\}$  with different release times. If the worker  $w$  follows the prediction based route recommendation algorithm, he/she would choose task  $t_3$ , which is neither the nearest nor the longest task. Then he would choose task  $t_4$  after he arrived at the destination of task  $t_3$ . There is still time for him to finish task  $t_5$  after he arrives at the destination of task  $t_4$ , so he finishes task  $t_5$  also. The prediction based route recommendation algorithm ends until the worker returns his/her destination. The final route is  $t_3 \rightarrow t_4 \rightarrow t_5$ , and is optimal.

### 3.4 Extended prediction based route recommendation algorithm

Although the prediction based route recommendation algorithm considers the three factors mentioned above, but it does not consider the end point of the worker  $w$ . It is possible that  $w$  goes far away from his/her destination when the time is close to the deadline. In this case the worker will have no other choice except going back to the destination. Then his/her total reward will be less because he/she spends a lot of time returning to his/her

destination without finishing tasks. To overcome the shortcomings of the above method, we propose an extended prediction based route recommendation algorithm. We define online route recommendation score function  $g(w, t_i)$  as follows.

$$g(w, t_i) = \mu f(w, t_i) + (1 - \mu) \frac{c(e_{t_i}, e_w)}{\exp(d_w - cT)} \quad (2)$$

The basic idea of the extended prediction based route recommendation algorithm is that we consider the impact of the distance between the destination of each task  $e_{t_i}$  and the destination of the worker  $e_w$ . When there are two tasks with the same three factors mentioned above, the worker will choose the closest one to his destination. The importance of the distance between the destination of each task  $e_{t_i}$  and the destination of worker  $e_w$  will grow over time via exponential function.  $cT$  is a parameter in exponential function which means current time. When the time is near deadline, a task which is nearer to the destination of worker  $w$  would get higher recommendation score, that will ensure the worker chooses tasks near his destination when the time is running out.

---

**Algorithm 4:** Extended Prediction Based Route Recommendation Algorithm

---

```

input :  $w, T, currentTime$ 
output:  $R^w$ 
 $idleTime \leftarrow 0$ ;
 $S_{candidate} \leftarrow \emptyset$ ;
while  $currentTime < d_w$  do
  if  $idleTime + c(w, e_w) \geq d_w$  then
     $\perp$  break;
  update  $T$ ;
  update  $currentTime$ ;
  if  $|T| > 0$  and  $currentTime \geq idleTime$  then
    foreach  $t \in T$  do
      if  $idleTime \geq a_t$  and  $idleTime + c(w, s_t) \leq d_t$  and
         $idleTime + c(w, s_t) + c(s_t, e_t) + c(e_t, e_w) \leq d_w$  then
         $\perp$   $S_{candidate} \leftarrow S_{candidate} \cup t$ ;
      if  $S_{candidate} = \emptyset$  then
         $\perp$   $idleTime \leftarrow currentTime$ ;
         $\perp$  continue;
      foreach  $t_i \in S_{candidate}$  do
        compute  $UOTD(e_{t_i})$ ;
         $f(w, t_i) \leftarrow \alpha(1 - \frac{c(w, s_{t_i})}{R(w)}) + \beta \frac{c(s_{t_i}, e_{t_i})}{MaxD} + \gamma \frac{UOTD(e_{t_i})}{N}$ ;
         $g(w, t_i) \leftarrow \mu f(w, t_i) + (1 - \mu) \frac{c(e_{t_i}, e_w)}{\exp(d_w - idleTime)}$ ;
         $t^* \leftarrow \operatorname{argmax}_{t_i} f(w, t_i)$ ;
         $R^w \leftarrow R^w \cup t^*$ ;
         $S_{candidate} \leftarrow \emptyset$ ;
         $idleTime \leftarrow idleTime + c(w, s_{t^*}) + c(s_{t^*}, e_{t^*})$ ;
      if  $currentTime > idleTime$  then
         $\perp$   $idleTime \leftarrow currentTime$ ;
return  $R^w$ 

```

---

Algorithm 4 illustrates the procedure of the extended prediction based route recommendation algorithm for the OnlineDRR problem. In lines 1-17, EPBR is similar to PBR. Specifically, in lines 1-2, we represent the time when the worker is ready to take a task as  $idleTime$ , and the valid candidate task sets for choosing as  $S_{candidate}$ . In lines 3-24, we continue to find the valid task one by one. In lines 3-4, we check the deadline constraint of the worker. In lines 10-11, We check every task whether the worker would have enough time to go back to his destination if he accepts this task, and we add valid tasks into set  $S_{candidate}$ . If the set  $S_{candidate}$  is empty, the worker would keep on waiting until there is new suitable task. In lines 15-17, for each task  $t$  in set  $S_{candidate}$ , we first compute the unit original task demand around the destination of each task  $UOTD(e_{t_i})$ , then we compute the online route recommendation score for each valid task. Next, we compute new recommendation score  $g(w, t_i)$  which considers the distance between the destination of each task  $e_{t_i}$  and the destination of worker  $e_w$ . After that, we select the task with the highest recommendation score value and add it into set  $S_{candidate}$ . In lines 23-24, when  $currentTime$  is greater than  $idleTime$ , the worker is waiting for tasks and we set  $idleTime$  to the current system time. The algorithm will end when  $currentTime$  exceeds the deadline of the worker.

*Example 5* We still consider the scenario shown in Example 1, the worker  $w$  starts to work at location (4,4) and there is a set of tasks  $T = \{t_1, t_2, \dots, t_5\}$  with different release times. If we set the value of  $\mu$  to 1, the result is the same with the prediction based route recommendation algorithm in this case. The final route is  $t_3 \rightarrow t_4 \rightarrow t_5$ , and is optimal. However, we should notice that the destination of the worker is arbitrary set up by the worker. The extended prediction based route recommendation algorithm considers the distance between the destination of each task  $e_{t_i}$  and the destination of the worker  $e_w$ . It would be much more helpful for a worker to select a task near his destination when approaching the deadline. The difference between the results of the two algorithms can be observed in the experimental section.

## 4 Experiments

### 4.1 Experimental settings

In this section, we describe our experimental settings for evaluating our proposed algorithms.

**Datasets** We use both synthetic and real datasets for evaluating our algorithms. We use New York City TCL Trip Record dataset [20] which contains nine-year taxi records in New York City. Each record contains pick-up and drop-off times, and coordinates, as well as the fare amount of each trip record.

For the synthetic dataset, we generate tasks following normal distribution. Both the worker and tasks are located in a 50\*50 grid. The configuration of the synthetic dataset is shown in Table 2. Default values are denoted in bold font. We test our proposed algorithms via varying 6 parameters: the service range of the worker  $r_w$ , the mean and standard deviation of coordinate values of tasks' start points,  $\bar{S}$ ,  $\delta_s$ , respectively, the mean and standard deviation of the length of all tasks,  $\bar{D}$ ,  $\delta_d$ , respectively, and the number of all tasks  $|T|$ .

In addition, four algorithms mentioned in this paper are all implemented in C++, and the experiments were performed on a machine with Intel i5 2.70GHZ 2-core CPU and 8GB memory.

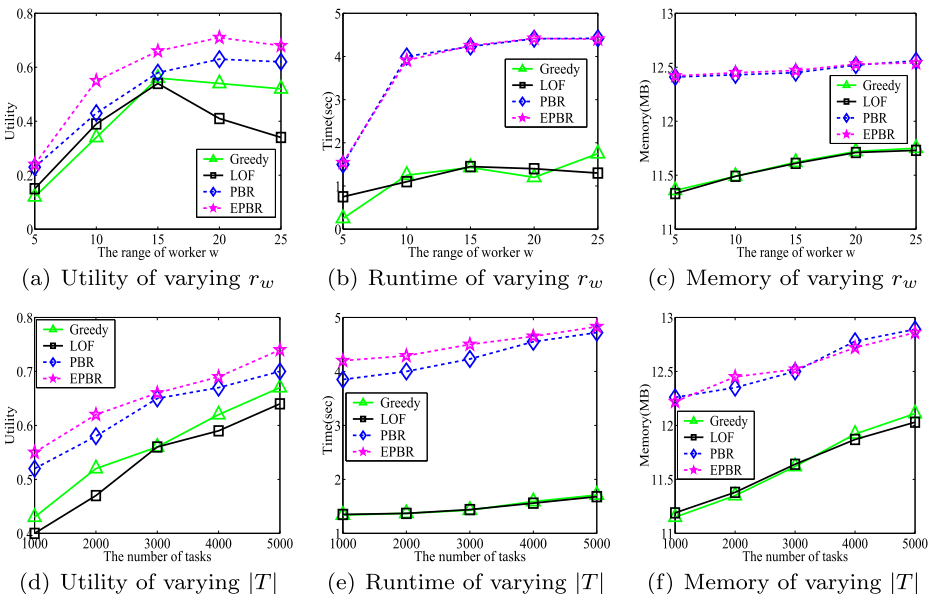
**Table 2** Synthetic Dataset

Factor	Settings
$r_w$	5, 10, <b>15</b> , 20, 25
$\bar{S}$	5, 15, <b>25</b> , 35, 45
$\delta_s$	<b>5</b> , 10, 15, 20, 25
$\bar{D}$	3, 6, <b>9</b> , 12, 15
$\delta_d$	1, 2, <b>3</b> , 4, 5
$ T $	1000, 2000, <b>3000</b> , 4000, 5000

### 4.2 Experiment results

In this section, we evaluate our proposed algorithms in terms of the income of the worker, running time, and memory cost. We define the reward of each task as utility, and its value is set as the non-idle time of the worker in the experimental settings. The income of the worker can be obtained by the total utility score he gets within his time budget. We show the results of utility normalized by the time budget of the worker. We test our algorithms via varying 6 parameters mentioned above. Longest Order First(LOF) is the baseline method, which chooses the task with maximum distance. The greedy algorithm selects the nearest task when it satisfies the constraints. We show the best performance results of PBR(Prediction Based Recommendation) and EPBR(Extended Prediction Based Recommendation). In order to acquire accuracy and stability, we carefully tuning the balance factor  $\alpha, \beta, \gamma, \mu$  based on grid search.

**Effect of  $r_w$**  We first study the effect of varying the parameter  $r_w$ , the results are shown in Figure 3a and c. From the utility result shown in Figure 3a, we can observe that EPBR



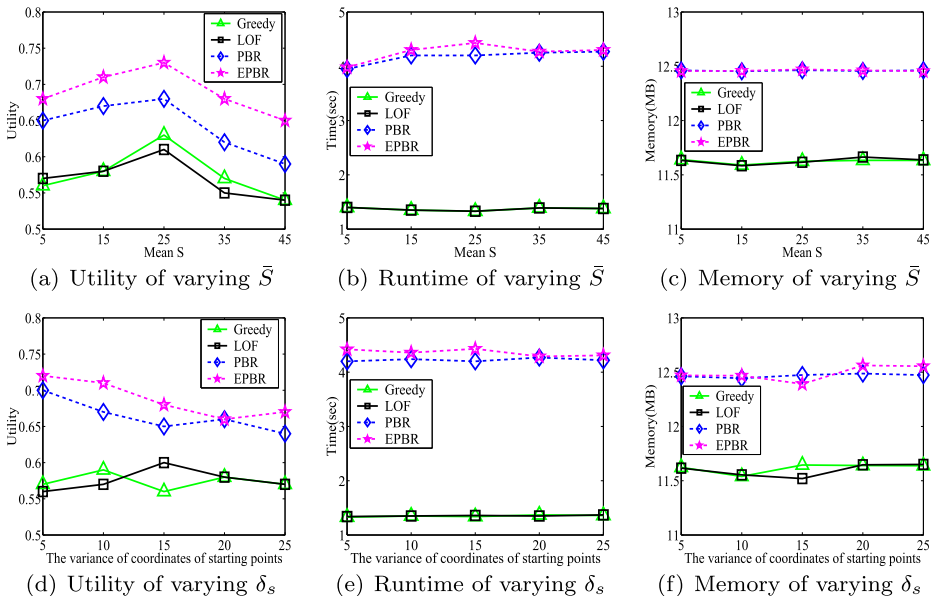
**Figure 3** Results on varying  $r_w$  and  $|T|$

performs better than the others and PBR is the second best. At first when the range is small, there is not enough tasks to choose, so the utility is low for all of them. The utility is higher with the increase of range for PBR and EPBR. When the range is larger than 15, there is more tasks to choose. The utility of LOF decreases because no-load distance will arise since the worker chooses to do the longest task. Greedy performs better than LOF when the range is larger than 15. The running time cost of Greedy and LOF are nearly the same. PBR and EPBR needs to compute the task demand around the destination of each chosen task and compute the highest score to recommend, so it takes more time than others.

**Effect of  $|T|$ .** We then study the effect of varying the parameter  $|T|$ , the results are shown in Figure 3d and f. From the utility result shown in Figure 3d, we can observe that the utilities of all algorithms get higher when the number of tasks increases. When  $|T|$  increases, there are more tasks in the grid so it is beneficial to the worker to choose better tasks. At the same time, the running time cost and memory cost of all algorithms grows as the number of tasks increases because more tasks need more time to process and more memory to store.

**Effect of  $\bar{S}$ .** We then study the effect of varying the parameter  $\bar{S}$ , the results are shown in Figure 4a and c. From the utility result shown in Figure 4a, we can observe that all four algorithms performs better when the mean coordinates of start points is 25. As it is the center of the grid, there is more tasks to choose so it is reasonable to get higher utility value. In contrast, the utility decreases when the coordinate becomes farther from center. At the same time, EPBR still performs the best. The running time cost and memory cost of all algorithms nearly remains unchanged.

**Effect of  $\delta_s$**  We then study the effect of varying the parameter  $\delta_s$ , the results are shown in Figure 4d and f. From the utility result shown in Figure 4d, we can observe that the utility



**Figure 4** Results on varying  $\bar{S}$  and  $\delta_s$



varies when  $\delta_s$  changes. The utilities of Greedy and LOF are higher alternately. For PBR and EPBR, the utility is higher when the variance is smaller. Similar to the results of effect of  $\bar{D}$ , the running time cost and memory cost of all algorithms nearly remains unchanged.

**Effect of  $\bar{D}$**  We then study the effect of varying the parameter  $\bar{D}$ , the results are shown in Figure 5a and c. From the utility result shown in Figure 5a, we can observe that the utility rises when the mean length of tasks increases. It is reasonable because the workers could get more reward each time when the average length of tasks increases, since the reward of a task is usually related to its length. When the average length of tasks is greater than 9, LOF performs better than Greedy. PBR and EPBR performs much better than the other two algorithms consistently. The runtime and memory cost of all the algorithms fluctuate as  $\bar{D}$  changes.

**Effect of  $\delta_d$ .** We then study the effect of varying the parameter  $\delta_d$ , the results are shown in Figure 5d and f. From the utility result shown in Figure 5d, we can observe that there is no much change for all four algorithms. The utility basically maintains stable for each algorithm. The utility of PBR and EPBR will decrease with the increase of the variance of tasks, but PBR and EPBR still performances much better than LOF and Greedy.

**Real dataset** Next we evaluate the proposed algorithms on the real dataset. There is a lot of work on mobile data processing [31–34, 54, 59]. After preprocessing, every order is extracted with pick-up/drop-off time, coordinate and taxi billing information. We select 2500 orders from New York City TCL Trip Record dataset as the task set every time and test 100 times. In the same time, we set the maximum wait time of each task as 5 minutes. When a task is not chosen since it has existed for 5 minutes, it will become invalid. The start point of the worker is initialized randomly. The values of parameters  $\alpha, \beta, \gamma, \mu$  are set to 0.5, 0.2, 0.3, 0.7, respectively. We test these four algorithms by varying the range of

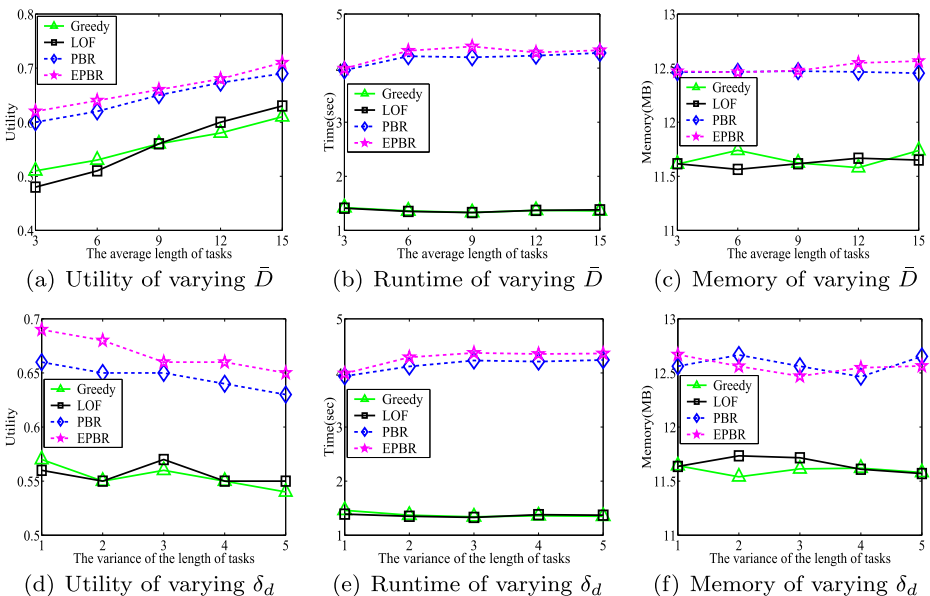
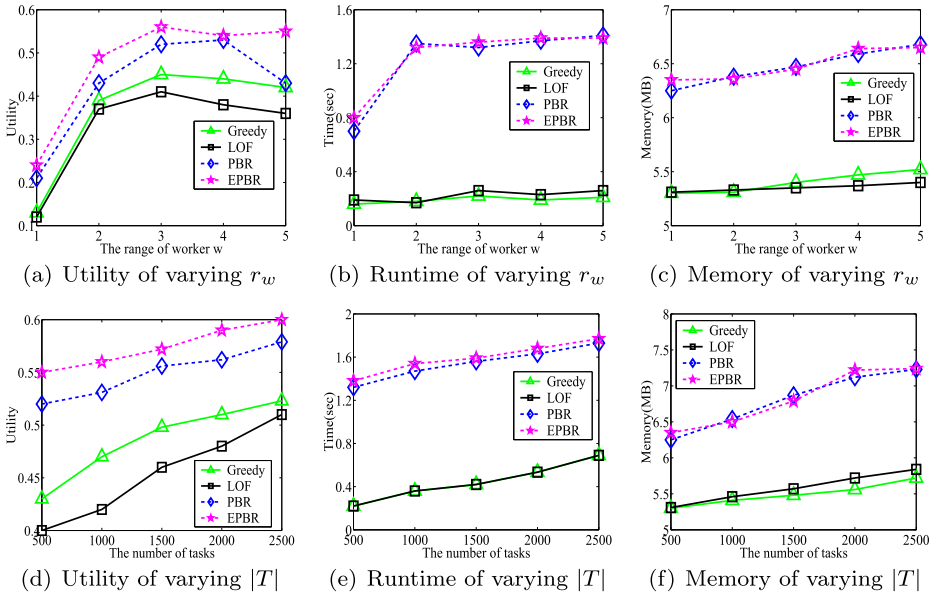


Figure 5 Results on varying  $\bar{D}$  and  $\delta_d$



**Figure 6** Results on the real dataset

worker and the number of orders. The experimental results based on real data are shown in Figure 6. As shown in Figure 6a, the utility on real data is not as high as synthetic data due to the density of tasks, and we can observe from Figure 6a, the utility of EPBR is still the highest which proves it is effective. The running time cost and memory cost of PBR is similar to EPBR. The running time cost and memory cost of Greedy and LOF are almost the same always. At the same time, the running time cost and memory cost of all algorithms grows as the number of tasks increases and all algorithms run fast.

## 5 Related work

### 5.1 Spatial crowdsourcing

Spatial data management and analysis is always one of fundamental topics in database and data mining communities [4, 22, 26, 27, 29, 35–37, 58]. In recent years, spatial crowdsourcing has been studied widely. One of the most important issues in crowdsourcing is task assignment [45, 47, 48]. For online scenarios, Tong [48] studies task assignment in crowdsourcing with workers arriving dynamically and [47] presents a comprehensive experimental comparison of online matching algorithms in spatial crowdsourcing.

Spatial crowdsourcing has attracted much interest from the industry as well as research communities with the development of smartphones. Task assignment [3, 7, 14, 39, 41, 49] is a primary focus in spatial crowdsourcing. Kazemi and Shahabi [14] define a maximum task assignment problem which aims to maximize the number of assigned tasks. Different from the offline scenarios adopted by the aforementioned work, task assignment under online scenarios has also been investigated and the problem is often formulated as a bipartite matching

problem [40, 42, 47, 48, 50]. Tong et al. [48] first identify the global online microtask allocation in spatial crowdsourcing and presented an algorithm with  $\frac{1}{4}$  competitive ratio. The difference of our work from task assignment lies in that most studies on task assignment optimizes the total number of assigned tasks or the total weighted value of assignment from the platform's perspective while in this paper, we try to maximize a single worker's income.

Different from task assignment problem, the issue of route recommendation in spatial crowdsourcing is less studied [2, 5, 16, 30]. [5] focus on offline scenarios, while tasks arrive dynamically in real applications. Considering the reality, tasks are specified with a deadline. In this paper, tasks will expire after their deadlines. Li et al. [16] is the closest one to our work, considering online scenarios, the biggest difference lies in that tasks in this paper are of delivery type while they are visit-type in [16]. Furthermore, how to make a plan for a worker in spatial crowdsourcing in order to maximize the worker's income is a challenging problem.

## 5.2 Orienteering problem

The orienteering problem is a routing problem with a time budget and its goal is to determine which nodes to visit and in which order so that the total score is maximized [8]. Vansteenwegen et al. [51] and Gunawan et al. [9] presented a comprehensive survey about the orienteering problem and many typical variants of it. There are some time-related variants such as the Orienteering Problem with Time Windows (OPTW) [13] where each node can only be visited during a particular time range, and the Time Dependent Orienteering Problem (TDOP) [6] where the cost between two nodes depends on time. Varakantham et al. [52] addressed the problem of crowd congestion and provided route guidance to multiple selfish users. The closest variant of the orienteering problem to our OnlineDRR problem is the DOP-TW-ST-R (see Definition 6) where each node is associated with a service time and a time window and is on a directed graph metric space. But to the best of our knowledge, the DOP-TW-ST-R has not been studied in existing literature. In addition, for most variants of the orienteering problem, online algorithms are seldom devised. In other words, most studies on time-related variants of the orienteering problem only focus on an offline scenario, where the spatiotemporal information of nodes is known in advance.

## 5.3 Route recommendation

With the application of online to offline mode, route recommendation [19, 21, 25, 28, 43, 55, 61] is becoming more and more important nowadays. Shang [25] proposed collective travel planning query in spatial networks so as to offer societal and environmental benefits. Liu et al. [19] presented a novel route recommendation system to provide self-drive tourists. This is helpful for reducing the traffic jams and queuing time. Su et al. [43] presented a crowd-based route recommendation system which evaluates the quality of recommended routes. The most satisfactory routes with high score will be recommended to users. Qu et al. [21] developed a cost-effective recommender system for taxi drivers so as to maximize their profits. They provided a net profit objective function for evaluating routes. In addition, [61] propose a personalized and time-sensitive route recommendation system which considers users' personal preferences and the temporal contexts. They propose a route generation algorithm to form routes under time-sensitive constraints. In this paper, we propose prediction based route recommendation algorithms and our work aims to maximize the benefits of the worker under online scenario. Our work considers three factors mentioned above: (a)

the distance between the worker's location and the origin of each feasible task, (b) the distance between the origin and destination of each feasible task, and (c) the future demand originating from the destination of each task.

## 6 Conclusion

In this paper, we propose a novel problem called the Online Delivery Route Recommendation (OnlineDRR) problem. We first give the definition of online delivery route recommendation problem, and prove the Offline Delivery Route Recommendation (OfflineDRR) problem is NP-hard. Then we prove there does not exist a deterministic online algorithm has a constant competitive ratio for the OnlineDRR problem. we presented a set of solutions to the online delivery route recommendation problem, including a baseline approach, Greedy, Prediction Based Route Recommendation Algorithm(PBR), Extended Prediction Based Route Recommendation Algorithm(EPBR). We verify the effectiveness and efficiency of the proposed algorithms through extensive experiments on real and synthetic datasets. The EPBR algorithm is the most effective and is also efficient enough.

## References

1. Amazon mechanical turk. <https://www.mturk.com/>
2. Chen, C., Cheng, S., Lau, H.C., Misra, A.: Towards city-scale mobile crowdsourcing: Task recommendations under trajectory uncertainties. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, pp. 1113–1119 (2015)
3. Cheng, Y., Yuan, Y., Chen, L., Giraud-Carrier, C.G., Wang, G.: Complex event-participant planning and its incremental variant. In: ICDE. IEEE, pp. 859–870 (2017)
4. Cheng, Y., Yuan, Y., Chen, L., Wang, G., Giraud-Carrier, C.G., Sun, Y.: Distr: A distributed method for the reachability query over large uncertain graphs. *IEEE Trans. Parallel Distrib. Syst.* **27**(11), 3172–3185 (2016)
5. Deng, D., Shahabi, C., Demiryurek, U.: Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In: Proceedings of the 21st acm sigspatial international conference on advances in geographic information systems. ACM, pp. 324–333 (2013)
6. Fomin, F.V., Lingas, A.: Approximation algorithms for time-dependent orienteering. *Inf. Process. Lett.* **83**(2), 57–62 (2002)
7. Gao, D., Tong, Y., She, J., Song, T., Chen, L., Xu, K.: Top-k team recommendation and its variants in spatial crowdsourcing. *Data Sci. Eng.* **2**(2), 136–150 (2017)
8. Golden, B.L., Levy, L., Vohra, R.: The orienteering problem. *Nav. Res. Logist.* **34**(3), 307–318 (1987)
9. Gunawan, A., Lau, H.C., Vansteenwegen, P.: Orienteering problem: A survey of recent variants, solution approaches and applications. *Eur. J. Oper. Res.* **255**(2), 315–332 (2016)
10. Guo, D., Zhu, Y., Xu, W., Shang, S., Ding, Z.: How to find appropriate automobile exhibition halls: Towards a personalized recommendation service for auto show. *Neurocomputing* **213**, 95–101 (2016)
11. Han, J., Zheng, K., Sun, A., Shang, S., Wen, J.: Discovering neighborhood pattern queries by sample answers in knowledge base. In: 32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016, pp. 1014–1025 (2016)
12. Hu, S., Wen, J., Dou, Z., Shang, S.: Following the dynamic block on the Web. *World Wide Web* **19**(6), 1077–1101 (2016)
13. Kantor, M.G., Rosenwein, M.B.: The orienteering problem with time windows. *J. Oper. Res. Soc.* **43**(6), 629–635 (1992)
14. Kazemi, L., Shahabi, C.: Geocrowd: enabling query answering with spatial crowdsourcing. In: Proceedings of the 20th international conference on advances in geographic information systems. ACM, pp. 189–198 (2012)
15. Krumke, S.O.: Online optimization: Competitive analysis and beyond. *ZIB* (2006)
16. Li, Y., Yiu, M.L., Xu, W.: Oriented online route recommendation for spatial crowdsourcing task workers. In: International Symposium on Spatial and Temporal Databases. Springer, pp. 137–156 (2015)

17. Liu, A., Wang, W., Shang, S., Li, Q., Zhang, X.: Efficient task assignment in spatial crowdsourcing with worker and task privacy protection. *GeoInformatica* **online first**, 1–28 (2017)
18. Liu, J., Zhao, K., Sommer, P., Shang, S., Kusy, B., Lee, J., Jurdak, R.: A novel framework for online amnesic trajectory compression in resource-constrained environments. *IEEE Trans. Knowl. Data Eng.* **28**(11), 2827–2841 (2016)
19. Liu, L., Xu, J., Liao, S.S., Chen, H.: A real-time personalized route recommendation system for self-drive tourists based on vehicle to vehicle communication. *Expert Syst. Appl.* **41**(7), 3409–3417 (2014)
20. nyc. <http://www.nyc.gov/html/tlc/html/home/home.shtml>
21. Qu, M., Zhu, H., Liu, J., Liu, G., Xiong, H.: A cost-effective recommender system for taxi drivers. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp. 45–54 (2014)
22. Shang, S., Chen, L., Jensen, C.S., Wen, J., Kalnis, P.: Searching trajectories by regions of interest. *IEEE Trans. Knowl. Data Eng.* **29**(7), 1549–1562 (2017)
23. Shang, S., Chen, L., Wei, Z., Guo, D., Wen, J.: Dynamic shortest path monitoring in spatial networks. *J. Comput. Sci. Technol.* **31**(4), 637–648 (2016)
24. Shang, S., Chen, L., Wei, Z., Jensen, C.S., Wen, J., Kalnis, P.: Collective travel planning in spatial networks. In: 33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19–22, 2017, pp. 59–60 (2017)
25. Shang, S., Chen, L., Wei, Z., Jensen, C.S., Wen, J., Kalnis, P.: Collective travel planning in spatial networks. *IEEE Trans. Knowl. Data Eng.* **28**(5), 1132–1146 (2016)
26. Shang, S., Chen, L., Wei, Z., Jensen, C.S., Zheng, K., Kalnis, P.: Trajectory similarity join in spatial networks. *PVLDB* **10**(11), 1178–1189 (2017)
27. Shang, S., Ding, R., Zheng, K., Jensen, C.S., Kalnis, P., Zhou, X.: Personalized trajectory matching in spatial networks. *VLDB J.* **23**(3), 449–468 (2014)
28. Shang, S., Guo, D., Liu, J., Wen, J.: Prediction-based unobstructed route planning. *Neurocomputing* **213**, 147–154 (2016)
29. Shang, S., Guo, D., Liu, J., Zheng, K., Wen, J.: Finding regions of interest using location based social media. *Neurocomputing* **173**, 118–123 (2016)
30. Shang, S., Liu, J., Zheng, K., Lu, H., Pedersen, T.B., Wen, J.: Planning unobstructed paths in traffic-aware spatial networks. *GeoInformatica* **19**(4), 723–746 (2015)
31. Shang, S., Lu, H., Pedersen, T.B., Xie, X.: Finding traffic-aware fastest paths in spatial networks, in *SSTD*, pp. 128–145 (2013)
32. Shang, S., Lu, H., Pedersen, T.B., Xie, X.: Modeling of traffic-aware travel time in spatial networks, in *MDM*, pp. 247–250 (2013)
33. Shang, S., Wei, Z., Wen, J., Zhu, S.: Probabilistic nearest neighbor query in traffic-aware spatial networks. In: *Web Technologies and Applications - 18th Asia-Pacific Web Conference, APWeb 2016, Suzhou, China, September 23–25, 2016. Proceedings, Part I*, pp. 3–14 (2016)
34. Shang, S., Xie, K., Zheng, K., Liu, J., Wen, J.: VID join: Mapping trajectories to points of interest to support location-based services. *J. Comput. Sci. Technol.* **30**(4), 725–744 (2015)
35. Shang, S., Yuan, B., Deng, K., Xie, K., Zheng, K., Zhou, X.: PNN query processing on compressed trajectories. *GeoInformatica* **16**(3), 467–496 (2012)
36. Shang, S., Yuan, B., Deng, K., Xie, K., Zhou, X.: Finding the most accessible locations: reverse path nearest neighbor query in road networks, in *ACM SIGSPATIAL*, pp. 181–190 (2011)
37. Shang, S., Zheng, K., Jensen, C.S., Yang, B., Kalnis, P., Li, G., Wen, J.: Discovery of path nearby clusters in spatial networks. *IEEE Trans. Knowl. Data Eng.* **27**(6), 1505–1518 (2015)
38. Shang, S., Zhu, S., Guo, D., Lu, M.: Discovery of probabilistic nearest neighbors in traffic-aware spatial networks. *World Wide Web* **20**(5), 1135–1151 (2017)
39. She, J., Tong, Y., Chen, L.: Utility-aware social event-participant planning. In: *SIGMOD. ACM*, pp. 1629–1643 (2015)
40. She, J., Tong, Y., Chen, L., Cao, C.C.: Conflict-aware event-participant arrangement and its variant for online setting. *IEEE Trans. Knowl. Data Eng.* **28**(9), 2281–2295 (2016)
41. She, J., Tong, Y., Chen, L., Song, T.: Feedback-aware social event-participant arrangement. In: *SIGMOD. ACM*, pp. 851–865 (2017)
42. Song, T., Tong, Y., Wang, L., She, J., Yao, B., Chen, L., Xu, K.: Trichromatic online matching in real-time spatial crowdsourcing. In: *ICDE. IEEE*, pp. 1009–1020 (2017)
43. Su, H., Zheng, K., Huang, J., Jeung, H., Chen, L., Zhou, X.: Crowdplanner: A crowd-based route recommendation system. In: 2014 IEEE 30th international conference on Data engineering (icde). IEEE, pp. 1144–1155 (2014)
44. Tong, Y., Chen, L., Shahabi, C.: Spatial crowdsourcing: Challenges, techniques, and applications. *Proceedings of the VLDB Endowment* **10**(12), 1988–1991 (2017)

45. Tong, Y., Chen, L., Zhou, Z., Jagadish, H.V., Shou, L., Lv, W.: Slade: A smart large-scale task decomposer in crowdsourcing, *IEEE Transactions on Knowledge and Data Engineering*. <https://doi.org/10.1109/TKDE.2018.2797962> (2018)
46. Tong, Y., Chen, Y., Zhou, Z., Chen, L., Wang, J., Yang, Q., Ye, J., Lv, W.: The simpler the better: a unified approach to predicting original taxi demands based on large-scale online platforms. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 1653–1662 (2017)
47. Tong, Y., She, J., Ding, B., Chen, L., Wo, T., Xu, K.: Online minimum matching in real-time spatial data: experiments and analysis. *Proceedings of the Vldb Endowment* **9**(12), 1053–1064 (2016)
48. Tong, Y., She, J., Ding, B., Wang, L., Chen, L.: Online mobile micro-task allocation in spatial crowdsourcing. In: *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, pp. 49–60 (2016)
49. Tong, Y., She, J., Meng, R.: Bottleneck-aware arrangement over event-based social networks: the max-min approach. *World Wide Web* **19**(6), 1151–1177 (2016)
50. Tong, Y., Wang, L., Zhou, Z., Ding, B., Chen, L., Ye, J., Xu, K.: Flexible online task assignment in real-time spatial data. *Proceedings of the VLDB Endowment* **10**(11), 1334–1345 (2017)
51. Vansteenwegen, P., Souffriau, W., Van Oudheusden, D.: The orienteering problem: A survey. *Eur. J. Oper. Res.* **209**(1), 1–10 (2011)
52. Varakantham, P., Mostafa, H., Fu, N., Lau, H.C.: Direct: A scalable approach for route guidance in selfish orienteering problems (2015)
53. Wang, Y., Li, J., Zhong, Y., Zhu, S., Guo, D., Shang, S.: Discovery of accessible locations using region-based geo-social data. *World Wide Web*, pp. 1–16 (2018)
54. Xie, K., Deng, K., Shang, S., Zhou, X., Zheng, K.: Finding alternative shortest paths in spatial networks. *ACM Trans. Database Syst.* **37**(4), 29:1–29:31 (2012)
55. Yang, B., Guo, C., Jensen, C.S., Kaul, M., Shang, S.: Stochastic skyline route planning under time-varying uncertainty. In: *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pp. 136–147 (2014)
56. Zheng, B., Wang, H., Zheng, K., Su, H., Liu, K., Shang, S.: Sharkdb: an in-memory column-oriented storage for trajectory analysis. *World Wide Web* **21**(2), 455–485 (2018)
57. Zheng, K., Su, H., Zheng, B., Shang, S., Xu, J., Liu, J., Zhou, X.: Interactive top-k spatial keyword queries. In: *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pp. 423–434 (2015)
58. Zheng, K., Zheng, Y., Yuan, N.J., Shang, S.: On discovery of gathering patterns from trajectories, in *ICDE*, pp. 242–253 (2013)
59. Zheng, K., Zheng, Y., Yuan, N.J., Shang, S., Zhou, X.: Online discovery of gathering patterns over trajectories. *IEEE Trans. Knowl. Data Eng.* **26**(8), 1974–1988 (2014)
60. Zhu, S., Wang, Y., Shang, S., Zhao, G., Wang, J.: Probabilistic routing using multimodal data. *Neurocomputing* **253**, 49–55 (2017)
61. Zhu, X., Hao, R., Chi, H., Du, X.: Fineroute: Personalized and time-aware route recommendation based on check-ins. *IEEE Trans. Veh. Technol.* **66**(11), 10461–10469 (2017)