Flexible Online Task Assignment in Real-Time Spatial Data

Yongxin Tong[†] Libin Wang[†] Zimu Zhou[*] Bolin Ding[§] Lei Chen[‡] Jieping Ye[¶] Ke Xu[†]
[†] SKLSDE Lab and IRI, Beihang University, Beijing, China, [*] ETH Zurich, Zurich, Switzerland
[‡] The Hong Kong University of Science and Technology, Hong Kong SAR, China
[§] Microsoft Research, Redmond, WA, USA
[¶] Didi Research Institute, Didi Chuxing, Beijing, China
[†]{yxtong,lbwang,kexu}@buaa.edu.cn, [*]zimu.zhou@tik.ee.ethz.ch, [§]leichen@cse.ust.hk,
[‡]bolind@microsoft.com, [¶]yejieping@didichuxing.com

ABSTRACT

The popularity of Online To Offline (O2O) service platforms has spurred the need for online task assignment in real-time spatial data, where streams of spatially distributed tasks and workers are matched in real time such that the total number of assigned pairs is maximized. Existing online task assignment models assume that each worker is either assigned a task immediately or waits for a subsequent task at a fixed location once she/he appears on the platform. Yet in practice a worker may actively move around rather than passively wait in place if no task is assigned. In this paper, we define a new problem *Flexible Two-sided Online task Assignment* (FTOA). FTOA aims to guide idle workers based on the prediction of tasks and workers so as to increase the total number of assigned worker-task pairs. To address the FTOA problem, we face two challenges: (i) How to generate guidance for idle workers based on the prediction of the spatiotemporal distribution of tasks and workers? (ii) How to leverage the guidance of workers' movements to optimize the online task assignment? To this end, we propose a novel two-step framework, which integrates *offline prediction* and *online task assignment*. Specifically, we estimate the distributions of tasks and workers per time slot and per unit area, and design an online task assignment algorithm, *Prediction-oriented Online task Assignment in Real-time spatial data* (POLAR-OP). It yields a 0.47-competitive ratio, which is nearly twice better than that of the state-of-the-art. POLAR-OP also reduces the time complexity to process each newly-arrived task/worker to $\mathcal{O}(1)$. We validate the effectiveness and efficiency of our methods via extensive experiments on both synthetic datasets and real-world datasets from a large-scale taxi-calling platform.

1. INTRODUCTION

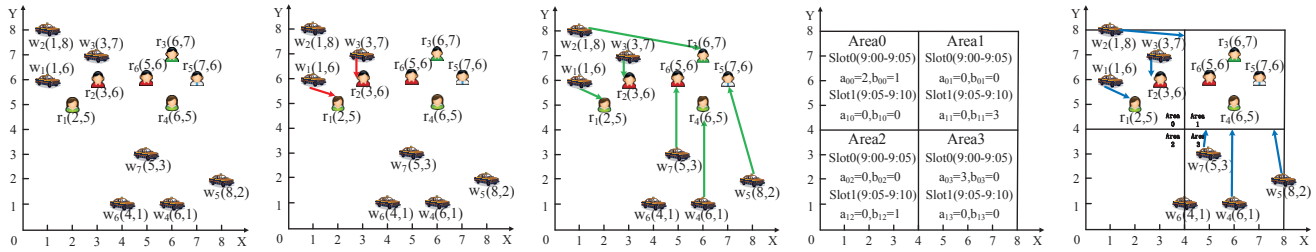
With the rapid development of mobile Internet and sharing economy, Online To Offline (O2O) service platforms are gaining increasing popularity, which demands for new massive real-time spatial data processing techniques. Representative O2O platforms include real-time taxi-calling services, *e.g.*, Uber [4] and Didi [1], product placement checking services in supermarkets, *e.g.*, Gigwalk [2], and on-wheel meal-ordering services, *e.g.*, GrubHub [3]. Large numbers of tasks appear on these platforms dynamically and need be assigned to certain workers in real time based on the spatial information provided by the workers' mobile devices. That is, these O2O platforms perform *online task assignment for real-time spatial data* with the aim to maximize the number of total assigned pairs [26].

Previous studies usually reduce the problem of online task assignment for real-time spatial data to the online maximum cardinality bipartite matching problem [26, 28] in *dynamic online scenarios*, where the spatiotemporal information of tasks and workers are unknown before they appear on the platforms. Although existing online task assignment models (online models for short) can deal with dynamic online scenarios, they make a strict assumption on the workers. Once a worker appears on the platform, she/he is immediately assigned a task or waits at her/his initial location for a task till her/his deadline [25, 26]. This assumption has two shortcomings. *(i)* In real-world O2O applications, it is impractical for a worker to passively stay in place. Instead, she/he tends to actively move around in the hope for suitable tasks elsewhere. *(ii)* The wait-in-place restriction on workers ignores the *potential* number of assigned pairs (*i.e.*, edges in the online bipartite graph). More task-worker pairs can be assigned if an idle worker can be guided in advance to the location where a new task may arrive.

Imagine the following scenario. David is a part-time Uber taxi driver (worker). When he logs in the Uber platform, he finds no taxi-calling tasks nearby. After waiting for 20 minutes without task assignment, he faces a dilemma whether to keep on waiting in place or try his luck elsewhere. However, existing online models can neither tell David whether he should stay or not, nor provide guidance on where he should go to get a task. Such a dilemma is common for workers on many O2O platforms, which raises a problem that most online models face: how to integrate the mobility of workers as a new decision option in a flexible model and guide workers to other locations such that the cardinality of the online assignment is maximized? To further illustrate this motivation, we go through the following toy example.

EXAMPLE 1. *Assume a real-time taxi-calling platform has six taxi-calling tasks ($r_1 - r_6$) by six requesters, and seven workers ($w_1 - w_7$),* i.e.*, seven taxis. The initial locations of the tasks and the workers are labeled in the 2D space (X, Y)*

Proceedings of the VLDB Endowment, Vol. 10, No. 11



(a) Initial locations of the tasks and the workers

(b) The online optimal result without workers' movement

(c) The offline optimal assignment allowing workers' movement

(d) A predictive number of the tasks/workers in each area and time slot

(e) The workers' movement based on the guide in Figure 1d

Figure 1: The results of different online models based on time in Table 1

Table 1: Arrival time of real-time taxi-calling tasks and workers (taxis)

9:00	9:00	9:01	9:01	9:02	9:03	9:03	9:03	9:04	9:05	9:06	9:07	9:08
w_1	r_1	w_2	w_3	r_2	w_4	w_5	w_6	w_7	r_3	r_4	r_5	r_6

in Figure 1a. The goal of the platform is to maximize the total number of valid assigned pairs (tasks and taxis) such that the assigned taxis can arrive at the locations of the requesters before their deadlines. Table 1 shows the arrival time of each task and worker. We assume the deadlines for tasks and workers are 2 minutes and 30 minutes, respectively, and the speed of workers is one unit per minute. If workers can only wait in place as in the existing online models, the online optimal result is shown by the red arrows in Figure 1b, where workers w_1 and w_3 are assigned to tasks r_1 and r_2, respectively, and the cardinality of the online optimal assignment is 2. If the platform assigns tasks offline, where the full spatiotemporal information of the tasks/workers is known in advance, then it can ask each worker to move to the location of the task assigned to her/him once the worker appears on the platform. Figure 1c shows the offline optimal assignment result and the movement of workers (green arrows). For example, worker w_6 appears on the platform at 9:03 and is guided to the location of task r_3 because the platform knows that r_3 will appear there at 9:05. Consequently, the cardinality of the offline optimal assignment is 6. Although it is impossible to achieve the offline optimal assignment without the full knowledge of the future tasks/workers, we observe the potential benefits by allowing workers to move around, which motivates us to predict future tasks/workers and guide workers to potential tasks in advance.

Motivated by the above example, we propose a flexible online model that takes the mobility of workers as a new decision option. Specifically, we formulate a new problem called *Flexible Two-sided Online task Assignment* (FTOA). FTOA allows workers to either wait in place or be guided to other locations if she/he is not assigned a task on arriving at the platform. To address the FTOA problem, a natural idea is to first predict the spatiotemporal distributions of subsequent tasks and workers and then utilize the predications to guide workers' movement and online task assignment. To achieve this goal, two challenges need to be addressed. *(i) How to effectively generate the guide for idle workers by predicting the spatiotemporal distribution of tasks and workers? (ii) How to leverage the guide of the workers' movement to optimize the online task assignment?* To the best of our knowledge, this is the first work that studies the FTOA problem. The main contributions of this work are summarized as follows.

- We propose and formulate FTOA, a new problem of online task assignment in real-time spatial data that is fit for practical O2O applications where workers are allowed to move around if no task is assigned.
- To address the FTOA problem, we develop a novel two-step framework, which integrates offline prediction and online task assignment.
- We devise a prediction-based online task assignment algorithm which achieves a 0.47-constant competitive ratio and includes a series of optimization techniques to improve its efficiency and effectiveness. In particular, it not only improves the competitive ratio by nearly two times than that of the state-of-the-art but also reduces the time complexity of processing each newly-arrived task/worker to $\mathcal{O}(1)$.
- We verify the effectiveness and efficiency of our methods on both synthetic datasets and datasets from a large-scale taxi-calling platform.

In the rest of this paper, we formulate the FTOA problem in Section 2 and provide an overview of the two-step framework in Section 3. Then we detail the new two-sided online task assignment algorithms and its optimizations in Section 4 and Section 5. Section 6 presents the performance evaluations, Section 7 reviews related work and Section 8 concludes this paper.

2. PROBLEM STATEMENT

In this section, we first introduce the basic concepts, and then formally define the FTOA problem. We also present a baseline approach called SimpleGreedy, which is extended from a state-of-the-art online task assignment model [26].

2.1 Preliminaries and Definitions

DEFINITION 1 (WORKER). *A worker, denoted by $w =< L_w, S_w, D_w >$, appears on the platform with an initial location L_w in the 2D space at time S_w and waiting time D_w. In other words, the worker w no longer provides services on the platform after the deadline $S_w + D_w$.*

DEFINITION 2 (TASK). *A task, denoted by $r =< L_r, S_r, D_r >$, is released on the platform at time S_r and at location*

Table 2: Summary of symbol notations

Notation	Description
$\widehat{W}$,$\widehat{R}$	The set of predicted workers and tasks
$m = \lvert\widehat{W}\rvert$	The cardinality of the set of predicted workers
$n = \lvert\widehat{R}\rvert$	The cardinality of the set of predicted tasks
W,R	The set of workers and tasks in real scenario
L_w, L_r	The locations of a worker and a task
S_w, S_r	Start times of a worker and a task
D_w, D_r	Deadlines of a worker and a task
a_{ij},b_{ij}	The predicted numbers of workers and tasks in time Slot$i$ and Area$j$
$d(L_w, L_r)$	Travel time from L_w to L_r
OPT	The optimal solution and also its value

L_r in the 2D space, and it needs to be served within D_r time. In other words, the task r will disappear from the platform if it is not assigned to a worker or the assigned worker fails to arrive at the location L_r before the time $S_r + D_r$.

Definition 3 (Travel Cost). *The travel cost, denoted by $d(w, r)$, is the time cost to travel from L_w to L_r.*

Notice that each worker has a velocity. Thus the travel cost from the location of a worker w to the location of a task r is the ratio of the Euclidean distance between w and r over the velocity. For simplicity, we assume the same velocity for all workers. Different velocities can be transformed into the same velocity by adjusting the travel costs.

Definition 4 (FTOA Problem). *Given a set of workers W and a set of tasks R, where workers and tasks can dynamically appear on the platform one by one at any time, the FTOA problem finds an assignment M among W and R to maximize the number of assigned pairs $MaxSum(M) = \sum_{w \in W, r \in R} I(w, r)$, where $I(w, r) = 1$ if the pair (w, r) is matched in the assignment M, and otherwise $I(w, r) = 0$, such that the following constraints are satisfied:*

- *Deadline constraint: For any worker-task pair (w, r), it should satisfy the following two deadline conditions. (1) The task should appear before the worker leaves the platform (*i.e., *$S_r < S_w + D_w$). (2) The worker should be able to arrive at the location of the assigned task before the deadline of the task (*i.e., *$D_r - (S_w - S_r) - d(L_w, L_r) \geq 0$).*
- *Worker's decision constraint: Once a new worker appears, she/he has the following three options: (i) She/He is assigned a task by the platform before her/his deadline; (ii) She/He stays in place and waits for a subsequent task before her/his deadline; (iii) She/He moves to a specific region instructed by the platform.*
- *Task's decision constraint: Once a new task is released, its location is fixed. The task is either assigned to a worker currently on the platform with the above deadline constraint satisfied or waits for one future worker who can meet the above deadline constraint.*
- *Invariable constraint: Once a task r is assigned to a worker w, the assignment of (w, r) cannot be revoked.*

Table 2 lists the notations used throughout the paper.

2.2 A Baseline Approach

The FTOA problem is hard because it needs to maximize the number of assigned pairs without knowing the future locations of tasks and workers. In this subsection, we first introduce a baseline solution called SimpleGreedy. The main idea of SimpleGreedy is that for any new object (a worker or a task), it picks those objects (tasks or workers) in the other set with the deadline constraint satisfied, and select one with the shortest distance.

Example 2. *Back to Example 1 in Figure 1, suppose all the tasks have $D_r = 2$ minutes and the workers move one unit per minute. SimpleGreedy will achieve a matching size 2, since every worker will stay in place and wait for new tasks. Only r_1 and r_2 will be finished before deadline.*

However, the optimal solution which knows the locations of future tasks can dispatch workers w_2, w_4, w_5 and w_6 in advance when they appear on the platform. The workers will move to the top right area to finish the four tasks. Finally, the OPT algorithm achieves a matching size of 6.

3. OVERVIEW OF OUR APPROACH

In this section, we introduce an overview of our solution framework and the evaluation metrics to assess the effectiveness of different online task assignment algorithms.

3.1 Overview of the Two-Step Framework

Our framework consists of two steps: *offline prediction* and *online task assignment*.

3.1.1 Offline Prediction

As it is hard to predict the exact spatiotemporal information of each task/worker, a reasonable relaxation is to predict the number of the tasks/workers in *a specific spatiotemporal range*. We use two concepts, "grid area" (a specific spatial range) and "time slot" (a specific temporal range), to partition the spatial and temporal dimensions, respectively. We illustrate the two concepts by the following example.

Example 3. *Back to Example 1 in Figure 1, the 2D space in Figure 1a is divided into four grid areas (Area 0 - Area 3), and the whole timeline in Table 1a is partitioned to two time slots (Slot 1 [9:00-9:05] and Slot 2 [9:06-9:10]). Figure 1d shows an illustrative prediction of the numbers of tasks and workers for each grid area and each time slot, where a_{ij} and b_{ij} denote the predicted number of workers and tasks in time slot i and grid area j, respectively. For example, the number of predicted tasks and workers in Area 3 and Slot 0 are 0 and 3, respectively.*

It is a well-studied topic to predict the number of moving objects per grid area and per time slot based on historical data such as demand-supply prediction of taxicabs [19, 31] and rents-returns prediction of sharing bikes [17]. In this work, we evaluate representative prediction methods on real-world datasets and select the most accurate one for our offline prediction (see Section 6.3).

3.1.2 Online Task Assignment

Based on the offline prediction results, we first generate an *offline guide* (Section 4). The offline guide transforms the predicted numbers of tasks and workers in each grid over multiple (future) time slots into pre-computed matched pairs of tasks and workers for the entire region. The offline guide serves as the basis for dispatching workers to other locations where tasks are likely to occur in the future, and

saves the time to process a newly-arrived worker or task, because a pseudo assignment is already computed.

With the offline guide, the platform refers to the pseudo assignment of a new object (a task or a worker), and makes the corresponding decisions. In case the actual workers and tasks deviate from the predictions, we propose a set of optimizations that can still ensure the theoretical guarantee on the total number of assigned worker-task pairs of the online task assignment (Section 5).

3.2 Evaluation Metrics of Online Algorithms

Online task assignment algorithms are the core in large-scale O2O service platforms, and it is essential to select the algorithms with the best theoretical guarantees on the total number of assigned worker-task pairs. The theoretical guarantees of different online algorithms are usually assessed by *competitive ratio (CR)*, which measures the differences between the outputs of an online algorithm and the optimal results under the assumption of knowing all the future locations of tasks and workers. Depending on the assumptions of the arrival of tasks and workers, the competitive ratio needs to be analyzed using different online input models [18]. Since our framework is built upon an offline prediction model that assumes independently and identically distributed (i.i.d) tasks and workers, we also adopt the i.i.d input model [7] to calculate the competitive ratio. Specifically, given α time slots and β grid areas, the spatiotemporal distribution of workers (tasks), $\mathcal{D}_{\mathcal{W}}$ ($\mathcal{D}_{\mathcal{R}}$), is defined by $Pr_a[i][j] = \frac{a_{ij}}{\sum_i \sum_j a_{ij}}$ ($Pr_b[i][j] = \frac{b_{ij}}{\sum_i \sum_j b_{ij}}$), where $1 \le i \le \alpha$, $1 \le j \le \beta$, and a_{ij} (b_{ij}) means the predicted number of workers (tasks) in time slot i and grid area j. There will be $m = \sum_i \sum_j a_{ij}$ trials for workers and $n = \sum_i \sum_j b_{ij}$ trials for tasks. The i.i.d model assumes that the dynamically arrived workers and tasks follow the aforementioned spatiotemporal distributions and the competitive ratio under the i.i.d model is defined as follows.

Definition 5 (CR in the i.i.d Model[7]). *The competitive ratio in the i.i.d model of an online algorithm for the FTOA problem is the minimum ratio of the result of the online algorithm over the optimal result under all possible arrival orders generated by the spatiotemporal distributions of the tasks and the workers $\mathcal{D}_{\mathcal{R}}$ and $\mathcal{D}_{\mathcal{W}}$,*

$$CR_{i.i.d} = min_{\forall v \in V \text{ follows } \mathcal{D}_{\mathcal{R}} \text{ and } \mathcal{D}_{\mathcal{W}}} \frac{MaxSum(M)}{MaxSum(OPT)}$$

where V is the set of all possible input orders of tasks and the workers, v is one order in V, $MaxSum(M)$ is the cardinality of assignment by the online algorithm, and $MaxSum(OPT)$ is the optimal cardinality of the offline scenario.

To sum up, as discussed in Section 3.1, the offline prediction results can be used to guide and optimize the online task assignment. As next, we present a novel prediction-oriented online task assignment approach, which consists of two parts: *offline guide generation* in Section 4 and *online task assignment* in Section 5.

4. OFFLINE GUIDE GENERATION

The offline guide generation component takes the predicted number of tasks and workers per grid area and per time slot as input, and outputs a pseudo assignment among these predicted tasks and workers in the form of a maximum bipartite matching. Based on the predicted number of tasks and workers, the offline generation component first instantiates the same number of *nodes* on the left (workers) and right (tasks) of a bipartite graph. The set of nodes on the left (right) is denoted by $\widehat{W}$ ($\widehat{R}$). The nodes from the prediction in the same grid area and time slot are labeled with the same *type*. Then we add an edge between a pair of nodes in the bipartite graph, *i.e.*, (w, r) ($w \in \widehat{W}, r \in \widehat{R}$), if the pair satisfies the deadline constraint in Definition 4. Finally, we adopt the classical maximum bipartite matching algorithm on the above bipartite graph to generate a maximum bipartite matching, *i.e.*, the offline guide.

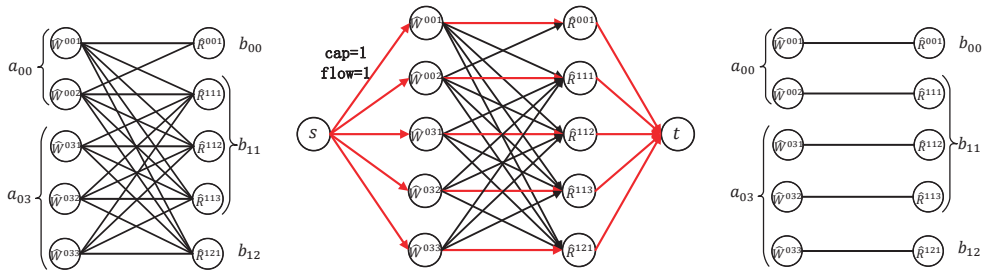
Algorithm 1: Offline Guide Generation Algorithm

```
   input  : Ŵ, R̂
   output: The offline guide Ĝ_f
1  Create source node s, sink node t;
2  foreach node w ∈ Ŵ do
3      add_edge(s,w,1);
4  foreach node r ∈ R̂ do
5      add_edge(r,t,1);
6  foreach node w ∈ Ŵ do
7      foreach node r ∈ R̂ do
8          if D_r − (S_w − S_r) − d(L_w, L_r) ≥ 0 and
             S_r < S_w + D_w then
9              add_edge(w,r,1)
10 Ĝ_f ← Maxflow(s, t);
11 return Ĝ_f
```

Algorithm 1 illustrates the generation of the offline guide. All predicted numbers of workers are instantiated to the set of left nodes, $\widehat{W}$, in a bipartite graph, while all predicted numbers of tasks are instantiated to the set of right nodes, $\widehat{R}$, in the same graph. In lines 1-5, we create one additional source node and one additional sink node, and add edges between the source node and each left node, as well as between the sink node and each right node, respectively. Then in lines 6-9, we add an edge between a pair of nodes if the corresponding predicted worker can complete the corresponding predicted task on time. In line 10, we apply the Ford-Fulkerson algorithm [5] to obtain a maximum cardinality bipartite matching, *i.e.*, the offline guide $\widehat{G_f}$, which is returned in line 11.

We make two notes here. (1) Any other max-flow algorithm is applicable to generate the offline guide. (2) We can further add a cost representing the travel cost between a worker and a task to the corresponding edge and apply any mincost-maxflow algorithm to derive a maximum cardinality bipartite matching with minimum travel cost.

Example 4. *Back to Example 3, Algorithm 1 will create nodes as depicted in Figure 2a. For instance, $a_{00} = 2$ means that the predicted number of workers in Slot 0 and Area 0 is 2, and there will be two nodes ($\widehat{W}^{001}$ and $\widehat{W}^{002}$) created on the left side representing the two predicted workers. $\widehat{W}^{ijk}$ ($\widehat{R}^{ijk}$) represents the k-th predicted worker (task) in Slot i and Area j. Since the worker in Slot 0, Area 0 can serve the*



(a) Network construction (b) Maxflow result (c) Final guide

Figure 2: Illustrated example of offline guide generation

task in Slot 0 and Area 0, two edges with $\widehat{R}^{001}$ will be added. After running a max flow algorithm, the maximum bipartite matching $\widehat{G_f}$ is created and the red edges in Figure 2b represent the edges carrying one unit of flow. Based on the max flow result, the final offline guide is presented in Figure 2c.

Complexity Analysis. The time complexity of offline guide generation depends on the implementation of line 10. An implementation such as the Ford-Fulkerson algorithm takes $\mathcal{O}(min(m,n)|E|)$ $(m = \sum_i \sum_j a_{ij}, n = \sum_i \sum_j b_{ij})$ time, where searching a new augmenting path takes $\mathcal{O}(|E|)$ time and there are at most $min(m,n)$ augmenting paths. Note that the guide is generated offline and thus it does not affect the efficiency of the online task assignment.

5. ONLINE ASSIGNMENT ALGORITHMS

In this section, we elaborate on our online task assignment algorithms based on the offline guide. We first present *prediction-oriented on-line task assignment in real-time spatial data* (POLAR), which has a competitive ratio of 0.4. We further propose POLAR-OPtimization (POLAR-OP), which has a better competitive ratio of 0.47. In the following, we distinguish a *node*, which is instantiated by a *predicted* worker (task) and refers to a node in the bipartite graph, and an *object*, which is an *actual* worker (task) during online task assignment.

5.1 POLAR Algorithm

POLAR is inspired by the solution to the one-sided online matching problem [7]. The main idea is that when a new *object* (an actual worker/task in the online task assignment process) appears, we let the object *occupy* a corresponding node in the offline guide $\widehat{G_f}$ and perform matching based on $\widehat{G_f}$. Here "corresponding" means that the object and the node it occupies are the same *type*. We use "*occupy*" to represent that each node in $\widehat{G_f}$ can only be used by at most one object.

Algorithm 2 summarizes the procedure of POLAR. In lines 2-4, when a new object o arrives, which could be either a worker or a task, POLAR makes the object occupy an unoccupied node w (or r) of its type (*i.e.*, the same indices of grids and time slots) in $\widehat{G_f}$. If no node can be occupied, the object is ignored. This situation occurs when the offline prediction underestimates the number of objects. Also note that in POLAR, each node can be occupied by one object at most. In lines 5-13, POLAR refers to the offline guide to perform the assignment for the object. Specifically, in line 6, POLAR checks whether the paired node r (or w) of node w (or r) in $\widehat{G_f}$ has been occupied by an object. If yes, it

Algorithm 2: POLAR Algorithm

input : $W, R, \widehat{G_f}$(the offline guide from Algorithm 1)
output: A feasible assignment M

1 $M \leftarrow \emptyset$; Mark all the nodes in $\widehat{G_f}$ unoccupied;
2 **foreach** *new arrival object* o **do**
3 w(or r)$\leftarrow$ an unoccupied node of o's type in $\widehat{G_f}$;
4 mark w(or r) occupied;
5 find the edge (w, r) in $\widehat{G_f}$;
6 **if** *r(or w) is occupied* **then**
7 assign o to the corresponding object of r(or w);
8 update M;
9 **else**
10 **if** *o is a worker* **then**
11 dispatch o to go to the area of r;
12 **else**
13 let o wait until its deadline;
14 **return** M

immediately assigns the newly arrived object o to the object occupying the paired node and updates the assignment. Otherwise in lines 9-13, POLAR dispatchs o in advance if it is a worker or lets o wait until its deadline if it is a task.

Example 5. *We continue to use Example 1 to illustrate POLAR in Figure 3a and refer to the guide in Figure 2c. In Slot0 and Area0, the worker w_1 who occupies $\widehat{W}^{001}$ arrives on the platform and is assigned to the task r_1 which occupies $\widehat{R}^{001}$ in Slot 0 and Area 0. The second worker w_2 who occupies $\widehat{W}^{002}$ is dispatched to area 1 since she/he matches the node $\widehat{R}^{111}$ in the guide, and she/he will serve the task r_3 in Slot1 and Area1. When the third worker w_3 in Slot 0 and Area 0 arrives, because there is no unoccupied node, POLAR just ignores it. In Slot 0 and Area 3, two workers (who occupy $\widehat{W}^{031}$ and $\widehat{W}^{032}$) will move to Area 1 and serve two tasks r_4 and r_5 (which occupy $\widehat{R}^{112}$ and $\widehat{R}^{113}$) in Slot 1 and Area 1. The worker w_6 who occupies $\widehat{W}^{033}$ in Area 3 is dispatched to Area 2 but fails to get a task because of the inaccurate prediction. The last worker appears in Area3 fails to occupy a node. The POLAR algorithm finally achieves a matching size of 4.*

Next we derive the competitive ratio of POLAR by giving the lower bound of the matching size and the upper bound of the *OPT*. We assume each pair matched based on the offline guide can be matched in reality. The assumption is reasonable because the offline guide complies with all the

constraints in Definition 4. Although the use of discrete time slots and areas in Algorithm 1 may affect slightly the inequalities, such differences can be ignored.

LEMMA 1. *We define E as the set of edges added in line 9 of Algorithm 1, which are also the edges in the bipartite graph, and $E^* = \{e \in E : f_e = 1\}$, which is the set of edges that carry the flow of one unit. With a probability of at least $1 - 2e^{-\Omega(m+n)}$, for any $\epsilon \geq 0$, the matching size returned by POLAR is at least $(1-\frac{1}{e})^2|E^*|-\epsilon(m+n)$, where $m = \sum_i \sum_j a_{ij}$ and $n = \sum_i \sum_j b_{ij}$.*

PROOF. *For any edge $e \in E^*$, it is chosen by the algorithm if and only if the two nodes, w and r, on this edge are occupied in the real online scenario. For a type of worker in some Sloti and in some Areaj, the new worker of this type comes with the probability of $\frac{a_{ij}}{m}$, and the new worker occupies one node with probability of $\frac{1}{a_{ij}}$. In the analysis, we assume the algorithm selects one node from all nodes of this type uniformly at random and lets the new object occupy in this node. The assumption only reduces the probability of occupying a node and does not affect the correctness of the analysis. Then the new worker occupies one node with probability of $\frac{1}{m}$. The probability that a node w is not occupied by any worker is $(1 - \frac{1}{m})^m$. Hence the probability that the node w is occupied by a worker is at least $1 - \frac{1}{e}$. Similarly, the probability that a node r is occupied by a task is $1 - (1 - \frac{1}{n})^n \geq 1 - \frac{1}{e}$. Note in the i.i.d. model the arrival of one type of nodes is independent from the other. We conclude that the probability that the edge e is chosen by the algorithm is at least $(1 - \frac{1}{e})^2$. Let ALG be the random variable denoting the matching size output by POLAR. We will have $E[ALG] \geq (1 - \frac{1}{e})^2|E^*|$. Note that ALG is a function of all the coming nodes. Let X_i be the node that the ith arrived object occupies. The series $Z_i = E[ALG|X_1, ...X_i]$ will form a Doob martingale. Here Z_0 is the expectation of ALG and Z_{m+n} is the final output from the algorithm. We also claim that ALG is a function of $m+n$ random variables, which satisfies the Lipschitz condition with bound 1. Formally, for any two series $\{X'_1, ..., X'_i, ..., X'_{m+n}\}$ and $\{X'_1, ..., X''_i, ..., X'_{m+n}\}$ different in only one position X'_i and X''_i, $|ALG(X'_1, ..., X'_i, ..., X'_{m+n}) - ALG(X'_1, ..., X''_i, ..., X'_{m+n})| \leq 1$. Therefore by Azuma-Hoeffding inequality, we obtain $\Pr(|Z_{m+n} - Z_0| \geq \epsilon(m + n)) \leq 2e^{-\frac{\epsilon^2(m+n)}{2}}$. After we replace $E[ALG]$ and ALG with Z_0 and Z_{m+n}, respectively,*

$$\Pr(ALG \leq E[ALG] - \epsilon(m + n)) \leq 2e^{-\frac{\epsilon^2(m+n)}{2}}. \quad (1)$$

Taking the opposite, we conclude that with a probability of at least $1-2e^{-\Omega(m+n)}$, for any $\epsilon \geq 0$, the POLAR algorithm achieves its matching size at least $(1-\frac{1}{e})^2|E^*|-\epsilon(m+n)$. □

We next derive the upper bound of OPT.

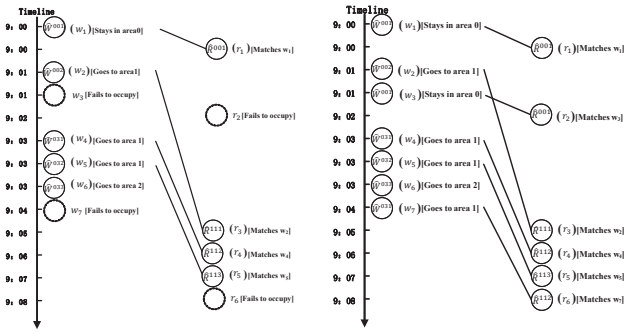
LEMMA 2. *Let OPT be the optimal matching size. With a probability of at least $1-e^{-\Omega(m+n)}$, for any $\epsilon > 0$, $OPT \leq |E^*| + \epsilon(m + n)$.*

PROOF. *In a network graph, the size of any cut is always an upper bound of any maxflow. We obtain the bound of OPT in the real online scenario by constructing a cut which uses the offline guide.*

In Algorithm 1, we get a residual network G_f where we could construct a mincut using the canonical "reachability" feasible flow [5]. Specifically, s with all the nodes s can reach G_f forms the set S, and the set T contains the remaining nodes. (S, T) forms the mincut of G.

We define $\widehat{W_S} = \widehat{W} \cap S$, $\widehat{W_T} = \widehat{W} \cap T$, $\widehat{R_S} = \widehat{R} \cap S$ and $\widehat{R_T} = \widehat{R} \cap T$. By contradiction, there is no edge between W_S and R_T. Hence, there are only edges in s to $\widehat{W_T}$ and $\widehat{R_S}$ to t in the mincut. Also note that mincut equals maxflow indicates that $|E^*| = |\widehat{W_T}| + |\widehat{R_S}|$. We will use this observation in the following proof.

The real online scenario (W, R) can achieve its OPT by constructing a network and using a maxflow algorithm. After a maxflow algorithm, we get G_f. Any cut in the residual network G_f is an upper bound on OPT. For a worker $w \in \widehat{W}$, we define $D(w)$ as the online workers that try to occupy the node w and $W_S = \cup_{w \in \widehat{W_S}} D(w)$, similarly W_T, R_S, R_T. The $s-t$ cut we define in G_f is $(W_S \cup R_S, W_T \cup R_T)$. Since there is no edge between W_S and R_T, there will be no edge between W_S and R_T. The cut size equals $|W_T| + |R_S|$. For any worker w, it occupies a node in W_T with a probability of $\frac{\sum_{w \in \widehat{W_T}} a_{S_w L_w}}{m}$, independent of other workers. Using a Chernoff bound, for any $\epsilon > 0$, with a probability of $1 - e^{-\Omega(m)}$, $|W_T| \leq |\widehat{W_T}| + \epsilon m$. Together with the observation above, we have $OPT \leq |E^*| + \epsilon(m + n)$. □



(a) POLAR (Size=4) (b) POLAR-OP (Size=6)

Figure 3: Illustrated examples of POLAR and POLAR-OP

THEOREM 1. *The POLAR algorithm achieves its competitive ratio $(1 - \frac{1}{e})^2 \approx 0.4$ with high probability.*

PROOF. *According to Lemma 1, with high probability, for any $\epsilon > 0$, we have ALG $>= (1 - \frac{1}{e})^2|E^*| - \epsilon(m + n)$. In Lemma 2, with high probability, for any $\epsilon > 0$, we have $OPT \leq |E^*| + \epsilon(m + n)$. Combining the two results, the theorem holds.* □

Complexity Analysis. It takes $\mathcal{O}(1)$ time to process each newly arrived object by referring to the offline guide.

5.2 Optimizing the POLAR Algorithm

Note that the workers and tasks that are not predicted by the offline prediction component are simply ignored by POLAR. In this subsection, we propose POLAR-OPtimization (POLAR-OP), which enhances the POLAR algorithm to handle such unexpected workers/tasks and has a better competitive ratio of 0.47. The main idea of POLAR-OP is that when an new object (an actual worker/task) appears, the platform lets the object *associate* a corresponding node in the offline guide $\widehat{G_f}$ and perform an online matching based

Algorithm 3: POLAR-OP Algorithm

input : $W, R, \widehat{G_f}$(the offline guide from Algorithm 1)
output: A feasible assignment M

```
1  M ← ∅;
2  foreach new arrival object o do
3      w(or r)← a node of o's type in Ĝ_f;
4      find the edge (w, r) in Ĝ_f;
5      if r(or w) is associated then
6          assign o to any object associated to r(or w);
7          update M;
8      else
9          if o is a worker then
10             dispatch o to go to the area of r;
11         else
12             let o wait until its deadline;
13 return M;
```

on $\widehat{G_f}$. Note that "*associate*" means that each node in $\widehat{G_f}$ can be reused by multiple objects.

Algorithm 3 illustrates the main procedure of POLAR-OP. The main difference between POLAR-OP and POLAR is that POLAR-OP allows objects to reuse the nodes in $\widehat{G_f}$. Specifically, in line 3 POLAR-OP tries to *associate* the object to a node w (or r) of its type. In contrast to POLAR, where one node can only be *occupied* by one object, a node in POLAR-OP can be associated to multiple objects. We only ignore the object when there is no node of its type. The rest of POLAR-OP is similar to POLAR. In line 4, we find the paired node r (or w) of the node w (or r) in $\widehat{G_f}$. In lines 5-12, if there are objects associated to the paired node, we immediately assign the newly arrived object o to *any* object associated to the paired node. Otherwise, we dispatch o based on the offline guide if it is a worker or let it wait in place if it is a task.

Example 6. *Back to our running example in Example 1. Figure 3b illustrates the process of the POLAR-OP Algorithm that can reuse the matched edges in the offline guide. In Slot 0 and Area 0, the third worker w_3 who is associated to $\widehat{W}^{001}$ again can be matched to the second task r_2 that is associated to $\widehat{R}^{001}$ in Area 0. The fourth worker w_7 who appears in Slot 0 and Area 3 will be associated to $\widehat{W}^{031}$'s position again, and will then go to Area 1 and serve the fourth task r_6 that could not be associated to a node at first. Finally the algorithm achieves a matching size of 6, which is equal to the optimal solution.*

Similar to previous theorems, we first derive the lower bound of the matching size with the following lemma and then the upper bound of OPT.

Lemma 3. *The POLAR-OP achieves a matching size of at least $0.47|E^*| - \epsilon(m+n)$ with high probability.*

Proof. *Due to the symmetry of two sides' nodes, we focus on the worker set nodes. For any worker w, it selects a certain node to occupy with probability of $1/m = 1/\sum_i \sum_j a_{ij}$. So for any node, in the whole procedure there are k tasks trying to occupy the same node with probability $\binom{m}{k}\frac{1}{m}^k(1-\frac{1}{m})^{m-k} \approx \frac{1}{ek!}$. When k is small compared with m, the random variable follows the Poisson distribution with $\mu = 1$, and we can use Poisson distribution to approximate the real distribution based on the widely studied balls into bins problem. Now we have two such random variables W_e and R_e which represent the number of two sides' balls into the edge e in E^*, respectively. The number of pairs matched in this edge equals the smaller one. Let M_e indicate the number of pairs matched in one edge e. We have*

$$\begin{aligned} E[ALG] &= \sum_{e\in E^*}\sum_i i\Pr[M_e = i] \\ &= |E^*|\sum_i i(2\Pr[R_e = i]\sum_{j=i}^{min(m,n)}\Pr[W_e = j] \\ &\quad - \Pr[R_e = i]\Pr[W_e = i]) \\ &= |E^*|\sum_i i[2\frac{1}{ei!}\sum_{j=i}^{min(m,n)}\frac{1}{ej!} - \frac{1}{(ei!)^2}]. \end{aligned} \tag{2}$$

Notice the general term of i is decreasing quickly and we only take the first three terms and the result turns out to be $E[ALG] \geq 0.47|E^*|$. Also using the Azuma-Hoeffding inequality about Doob martingale we have, with high probability, $ALG \geq 0.47|E^*| - \epsilon(m+n)$. □

Theorem 2. *The POLAR-OP algorithm achieves a competitive ratio of approximately 0.47.*

Proof. *Since the concentration result about OPT still holds, the theorem follows straightaway.* □

Complexity Analysis. Similarly, the time complexity of POLAR-OP is still $\mathcal{O}(1)$.

Table 3: Real dataset

City	$\|W\|$	$\|R\|$	D_w	D_r	t	g
Beijing	50637	54129	2	[0.5,0.75,1,1.25,1.5]	12	600
Hangzhou	49324	48507				

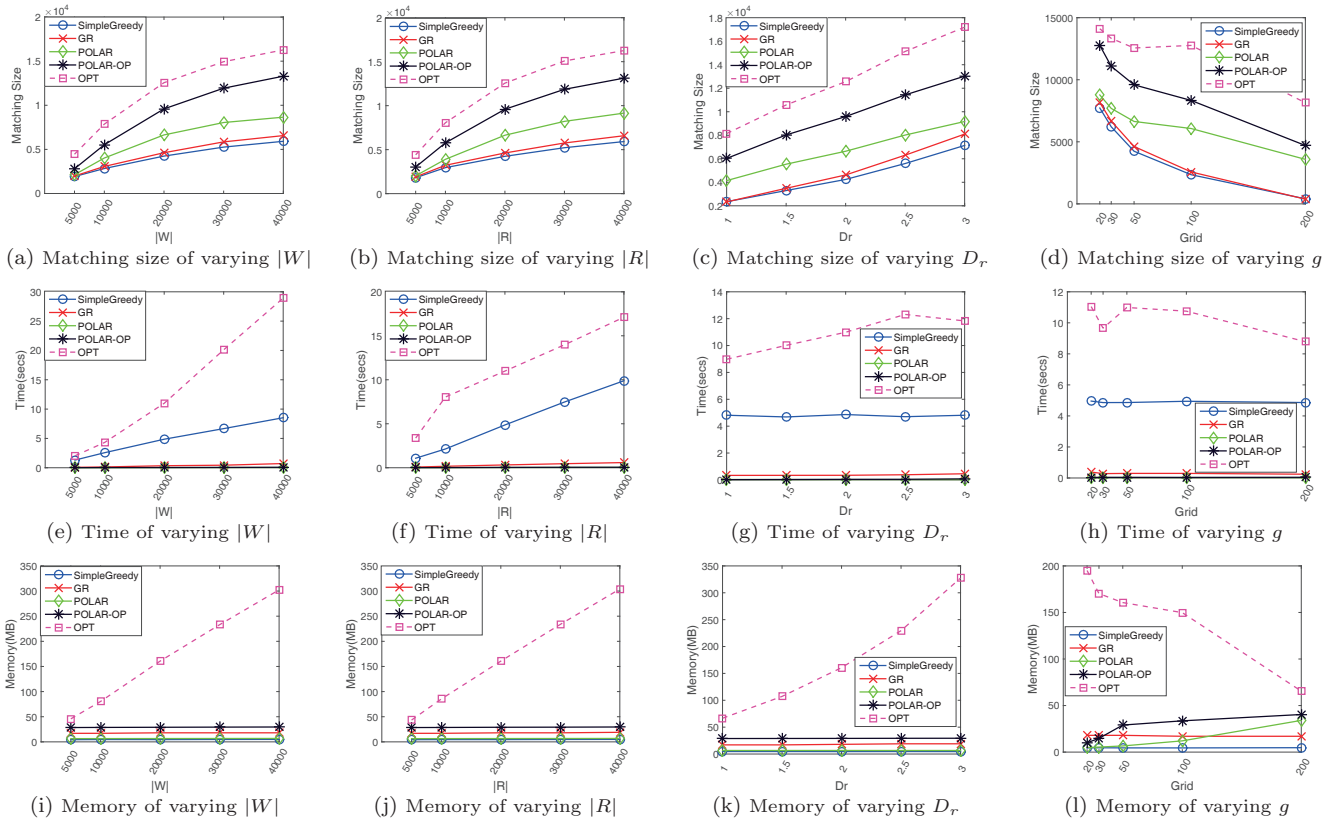
Table 4: Synthetic dataset

Factor	Setting
$\|W\|$	5000, 10k, **20k**, 30k, 40k
$\|R\|$	5000, 10k, **20k**, 30k, 40k
D_r	1.0, 1.5, **2.0**, 2.5, 3.0
$g = x \times y$(The number of grids)	20×20,30×30 **50×50**,100×100,200×200
t(The number of time slots)	12, 24, **48**, 96, 144
μ(Mean of temporal distribution obeying normal distribution)	0.25, 0.375, **0.5**, 0.625, 0.75
σ(Variance of temporal distribution obeying normal distribution)	0.25, 0.375, **0.5**, 0.625, 0.75
$mean$(Mean of spatial distribution obeying normal distribution)	0.25, 0.375, **0.5**, 0.625, 0.75
cov(Covariance of spatial distribution obeying normal distribution)	0.25, 0.375, **0.5**, 0.625, 0.75
Scalability	$\|W\| = \|R\|$ = 200k,400k, 600k,800k,1000k

6. EXPERIMENTAL STUDY

6.1 Experiment Setup

Synthetic Datasets. We generate 20000 workers and 20000 tasks on a 50×50 2D grid. We vary the number of workers, tasks, grid areas, and time slots to mimic a wide scale of real-world application scenarios. One slot represents 15 minutes and one grid represents a 0.01 (longitude) $\times$ 0.01 (latitude) square, and we set the global velocity of a



(a) Matching size of varying $|W|$ (b) Matching size of varying $|R|$ (c) Matching size of varying D_r (d) Matching size of varying g

(e) Time of varying $|W|$ (f) Time of varying $|R|$ (g) Time of varying D_r (h) Time of varying g

(i) Memory of varying $|W|$ (j) Memory of varying $|R|$ (k) Memory of varying D_r (l) Memory of varying g

Figure 4: Results on varying $|W|, |R|, D_r$ and the number of grids g

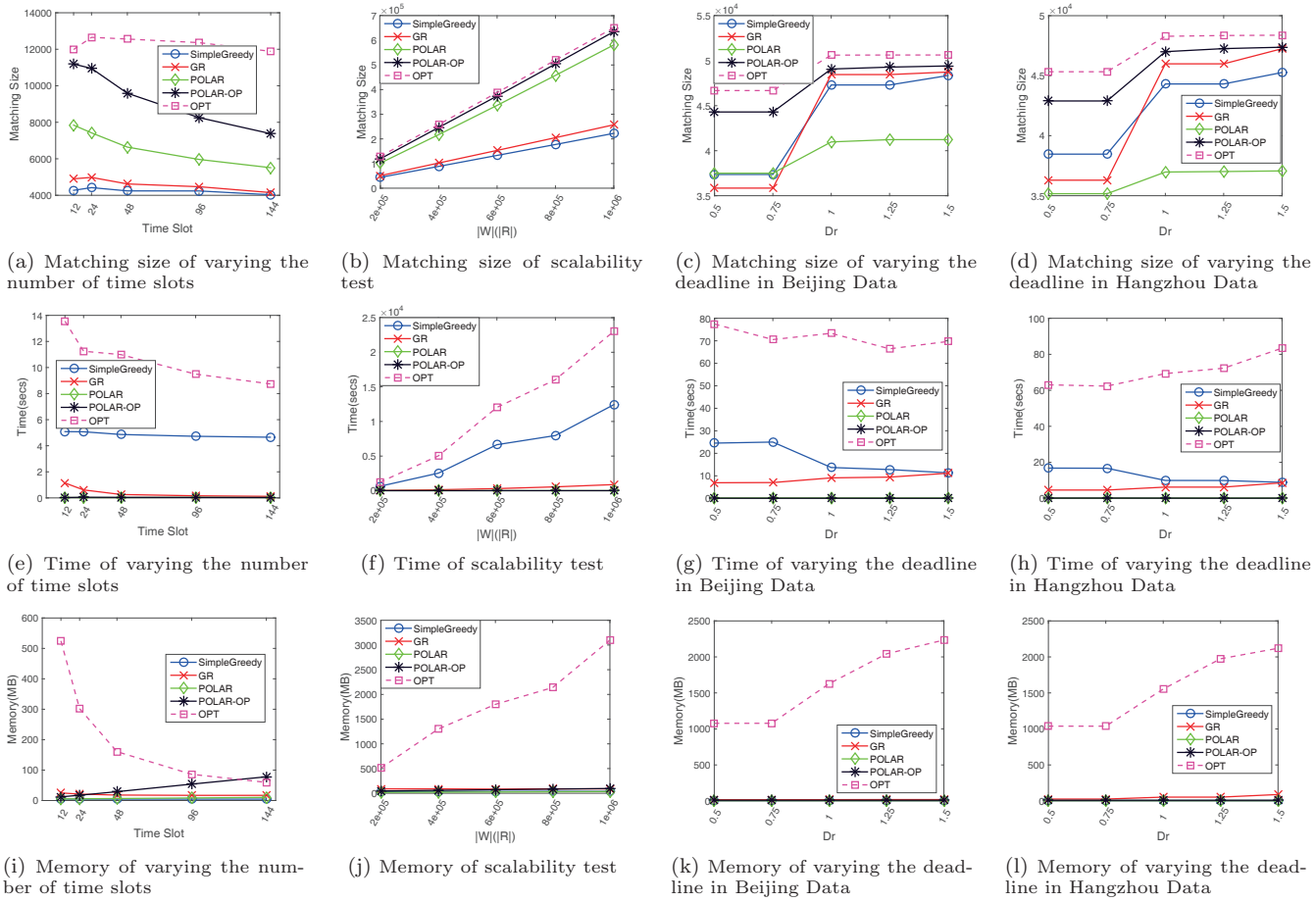
worker as 5 grids per slot, which is about 40 km/h. The tasks are usually urgent and hence we set the range D_r from 1 to 3. We simulate the temporal distribution and spatial distribution by Normal distribution, which is verified by recent studies [13]. The parameter μ in the temporal distribution is the value in the table times t, and σ can be computed in a similar way. By default, $\mu = 0.5 \times 48 = 24$ and $\sigma = 0.5 \times 48 = 24$. Since the spatial distribution has two dimensions, we use a multivariate normal distribution. Specifically, the mean is the value in the table times (x, y), and the covariance is the value in the table times the matrix $\begin{pmatrix} x & 0 \\ 0 & y \end{pmatrix}$. Since there is no correlation between x and y, we set the values along the diagonal of the matrix to be the square of their variance and the remaining positions as 0. Table 4 illustrates the configuration in detail and the default settings are marked in bold. The experiments using the synthetic datasets are performed on a machine with Intel (R) Core (TM) i7 3.80GHz CPU and 4GB main memory.

Real Datasets. We collected the taxi-calling data sampled in proportion from July 2016 to December 2016 in two cities, Beijing and Hangzhou, by a large-scale online taxi-calling platform in China [1]. Table 3 summarizes the default configurations. Notice that $|W|$ represents the number of the occurrence of workers. That is, a taxi that appears multiple times on the platform is counted as the number of times it occurs rather than 1. We test different values of D_r, since the parameter determines whether an assignment is feasible or not. We test the performance of the algorithms on each day. Thus, if one slot lasts for 15 minutes, there are 96 slots in total per day. We use $20 \times 30 = 600$ grids to cover the cities and one grid represents a 0.01 (longitude) $\times$ 0.01 (latitude) square. We ignore the data points beyond the scope of the rectangle. The experiments with the real datasets are conducted on a server with 32 Intel Xeon E5 2.4GHz processors with Hyper-Threading enabled and 128GB memory.

Compared Algorithms. We compare POLAR, POLAR-OP with two baselines, SimpleGreedy and GR [24], and OPT in terms of total matching size, running time and memory cost, and study the effect of varying parameters. GR [24] is one of the state-of-the-art dynamic task assignment algorithms, which gathers all objects within a time window and performs an assignment for the objects in each window. We omit the running time of the offline preprocessing. Note that OPT consumes time at the same level of the preprocessing. The algorithms are implemented in C++.

6.2 Experiment Results on Synthetic Datasets

Effect of $|W|$. The first column of Figure 4 shows the results of varying $|W|$. The matching size increases with the increase of $|W|$ for all the algorithms, because of more edges in the bipartite graph. POLAR-OP performs best. Both SimpleGreedy and GR perform worse than POLAR because they do not take potential matching pairs into account. GR marginally outperforms SimpleGreedy because GR assigns tasks after collecting a batch of objects while SimpleGreedy conducts task assignment on the arrival of each new object. All the algorithms need more time when $|W|$ increases. The running times of POLAR and POLAR-OP are close to 0, since they process an object in $\mathcal{O}(1)$ time. GR also runs fast since it only needs to match limited amounts of objects in each time window. SimpleGreedy takes notably longer



(a) Matching size of varying the number of time slots

(b) Matching size of scalability test

(c) Matching size of varying the deadline in Beijing Data

(d) Matching size of varying the deadline in Hangzhou Data

(e) Time of varying the number of time slots

(f) Time of scalability test

(g) Time of varying the deadline in Beijing Data

(h) Time of varying the deadline in Hangzhou Data

(i) Memory of varying the number of time slots

(j) Memory of scalability test

(k) Memory of varying the deadline in Beijing Data

(l) Memory of varying the deadline in Hangzhou Data

Figure 5: Results on varying the number of time slots, scalability test and Real Data

time because it has to retrieve all the objects when starting to process a new object. All except OPT have a low memory cost. The memory cost does not increase with $|W|$ because the size of the data structure is fixed.

Effect of $|R|$. The second column in Figure 4 shows the results of varying $|R|$. We observe similar performance trends when varying $|W|$, indicating that the role of workers and tasks are symmetrical in our FOTA problem.

Effect of deadline D_r. The third column in Figure 4 presents the results of varying D_r. Although we set a global deadline for all the tasks, the impact of D_r on the performances is the same as that of different deadlines for different tasks. The matching size of all the algorithms increases with the increase of D_r. This is because D_r represents the slackness of the inequality in the real-time constraint. The increase of D_r leads to more edges in the graph, and thus a larger matching size. POLAR-OP yields the largest matching size. All except OPT have constant running time and memory cost, indicating that OPT is sensitive to D_r.

Effect of the number of grids g. The fourth column in Figure 4 shows the results of different numbers of grids. The x-axis is the number of grids in one dimension. The matching size decreases as the number of grids increases. This is because by dividing the same region (and with the same number of workers and tasks) into more grids, the number of objects per grid will decrease. Consequently, the spatial overlap between workers and tasks per grid also drops, which leads to fewer edges in the graph. Nevertheless, POLAR-OP still outperforms the other online algorithms. The running time of all the algorithms remain relatively stable. The inflection point of OPT might be due to certain accidental factors. Since we construct a model for each grid, the memory cost increases with the increase of the number of grids.

Effect of the number of time slots t. The first column in Figure 5 depicts the results with different numbers of time slots. The impact of the number of time slots is similar to that of the number of grids. The matching size also decreases with the increase of time slots. The running time of SimpleGreedy is still the longest in all the online algorithms, and the memory cost of all the online algorithms slightly increase with more time slots.

Scalability. The second column in Figure 5 plots the results by increasing $|R|$ and $|W|$ simultaneously at the same scale. As OPT does not scale with the simultaneous increase of $|R|$ and $|W|$, we omit its time and memory results. For the matching size, POLAR-OP performs almost the same as OPT. SimpleGreedy performs the worst in running time, because it takes a long time to find the nearest neighbours. The running times of the other algorithms remain low and only increase slightly as the scale of the data increases. All algorithms are scalable in terms of memory.

Effect of the temporal distribution of tasks. The first column in Figure 6 shows the results when varying μ in the normal-distributed temporal distributions. μ reflects

the distance between the dense part of workers and that of tasks, since the temporal distribution of workers is fixed. The intention is to change the scale of the overlap from small to large. We observe that the matching size is insensitive to the temporal distribution of tasks.

The second column in Figure 6 shows the results when varying σ in the normal-distributed temporal distributions. σ determines how long the dense part of the tasks spans in time slots. We observe that when σ is greater than 0.25, the matching size remains almost unchanged, because the tasks' $\mu - \sigma$ can still reach the workers' $\mu = 0.25$. However, for a smaller σ, such as 0.125, the overlap drops quickly, since now $\mu - 2*0.125$ is lower than the workers' default $\mu = 0.25$. This will lead to a decrease in the matching size.

Effect of the spatial distribution of tasks. The third and fourth columns in Figure 6 shows the effect of different spatial distributions by varying the mean and covariance of the normal-distributed spatial distributions. The mean of the spatial distribution affects the matching size. As the center of tasks gets farther from the center of workers, the matching size decreases. However, when then mean of the spatial distribution of tasks is $(0.25x, 0.25y) = (12.5, 12.5)$, which is the same as that of workers, SimpleGreedy and GR seem to perform well. In this case, there is no need to dispatch workers in advance, and workers only need to wait in place for the tasks to appear in the same area. Similar results can be observed when varying the covariance of the normal distribution.

6.3 Experiment Results on Real Datasets

We first compare seven representative spatiotemporal prediction methods and choose the best as the offline prediction technique in our framework. We then evaluate the performance of the proposed approaches. Both evaluations are performed on a large-scale real-time taxi-calling platform.

6.3.1 Evaluation of Offline Prediction

Representative Prediction Approaches.

- Historical Average (HA): using the average of the history in the same time slot and the same grid area in the same day of week.
- Auto-Regressive Integrated Moving Average (ARIMA): using the well-known time-series model [31].
- Gradient Boosted Regression Tree (GBRT): using non-parametric regression, which is one of the most effective statistical learning models for prediction [8].
- Predictive Aggregation Queries (PAQ): using aggregation queries with moving object trajectories in the 6 latest hours [11, 22].
- Linear Regression (LR): using a linear regression model with the the numbers of tasks and workers of the 15 most recent corresponding periods.
- Neural Network (NN): using a neural network with the numbers of tasks and workers of the 15 most recent corresponding periods and other features *e.g.*, the weather condition [31].
- HP-MSI: using the state-of-the-art method to predict the number of bikes to be rent from or returned to each bike station [17].

Evaluation Metrics. We use two common metrics: *Error Rate (ER)* and *Root Mean Squared Logarithmic Error (RMLSE)* for evaluation, where $ER = \frac{1}{t}\sum_{i=1}^{t}\frac{\sum_{j=1}^{g}|a_{ij}-\widehat{a_{ij}}|}{\sum_{j=1}^{g}a_{ij}}$ and $RMLSE = \frac{1}{t}\sum_{i=1}^{t}\sqrt{\frac{1}{g}\sum_{j=1}^{g}(log(a_{ij}+1)-log(\widehat{a_{ij}}+1))^2}$.

Table 5: Prediction evaluation on real datasets

	Customer (Task)				Taxi (Worker)			
	Beijing		Hangzhou		Beijing		Hangzhou	
	RMLSE	ER	RMLSE	ER	RMLSE	ER	RMLSE	ER
HA	0.418	0.270	0.506	0.266	0.425	0.266	0.414	0.260
ARIMA	0.411	0.280	0.497	2.364	0.417	0.277	0.403	0.282
GBRT	0.399	0.251	0.493	0.225	0.405	0.245	0.395	0.255
PAQ	0.400	0.252	0.490	0.230	0.411	0.241	0.390	0.256
LR	0.403	0.253	0.489	0.245	0.414	0.250	0.400	0.253
NN	**0.396**	0.250	**0.479**	0.235	0.403	0.248	0.393	0.258
HP-MSI	0.399	**0.248**	0.489	**0.217**	**0.400**	**0.239**	**0.386**	**0.242**

Comparison of Offline Prediction Approaches. Table 5 summarizes the results of the compared prediction methods with respect to the two metrics using the two real datasets. Note that smaller values of the two metrics indicate more accurate prediction. From the results, we make the following observations. The three simple prediction approaches, HA, LR and ARIMA, perform poorly on both datasets. The reason may be that they cannot combine some complicated features such as weather condition, or reflect potential non-linear relationships between the features and the predictions. The three non-linear prediction models, NN, GBRT and PAQ, are competitive. A possible reason is that they take more features into consideration and model complicated relationships. Finally, HP-MSI achieves the best overall performance because it is tailored for spatiotemporal prediction. Thus we choose HP-MSI for the offline prediction component in our framework.

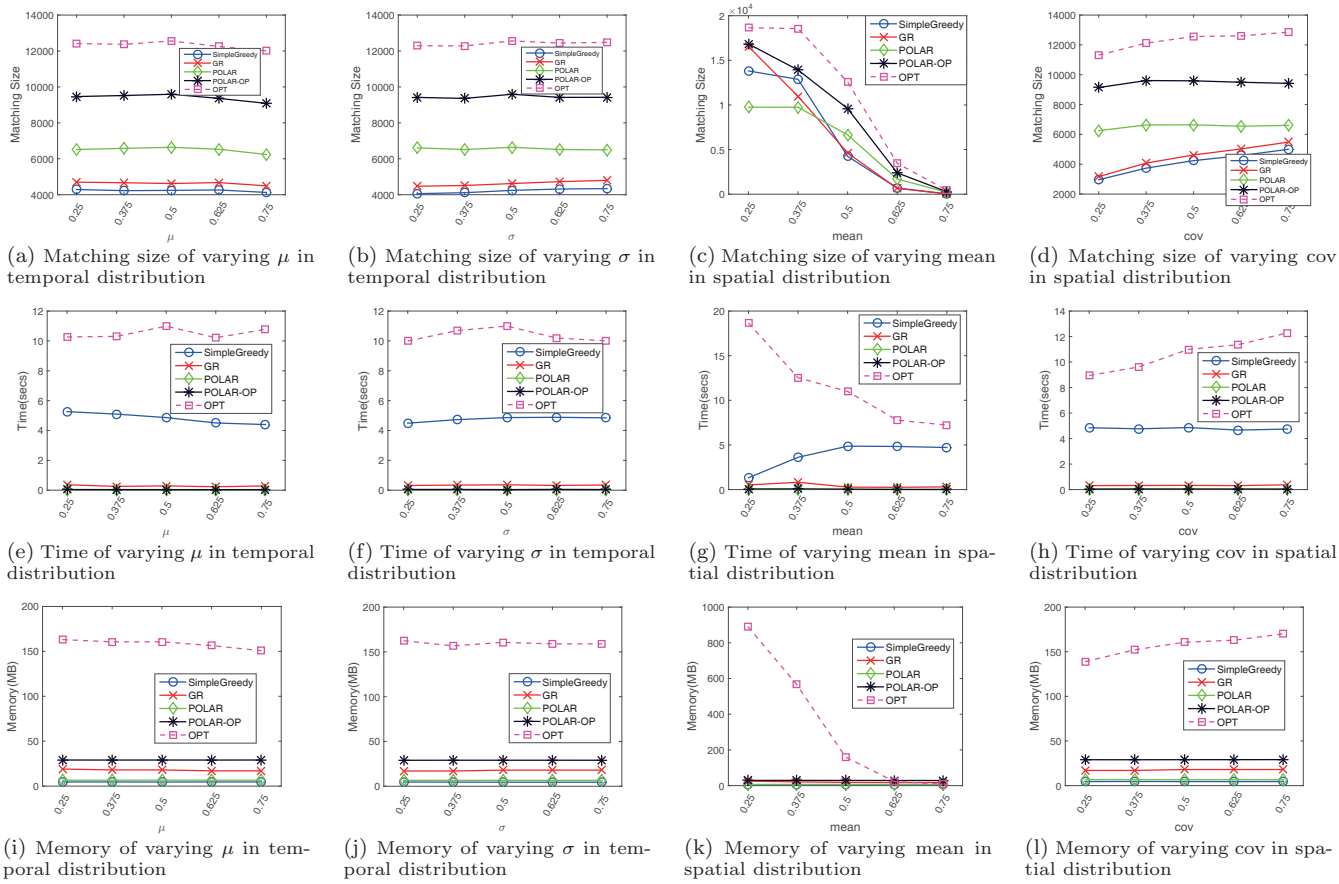
6.3.2 Evaluation of Online Task Assignment

We finally evaluate the performance of the online task assignment algorithms on the two datasets collected from a large-scale taxi-calling platform in the last two columns of Figure 5, where D_r is varied. We can observe that the performance of both POLAR and POLAR-OP is better than that of SimpleGreedy. In other words, the offline prediction affects the final matching size. Using the same prediction approach, the performances of the online task assignment algorithms exhibit similar patterns to the results of varying D_r on two real datasets. An interesting finding is that SimpleGreedy performs better than POLAR (Figure 5(c)-(d)). This may be explained by the supply-demand imbalance. Specifically, when there are more supplies than demands, it may not be beneficial to dispatch workers to certain areas in advance. Furthermore, since POLAR relies solely on the offline guide to assign tasks, the prediction errors will decrease the matching size and thus SimpleGreedy performs better. The explanation is also supported by Figure 6c. POLAR-OP performs the best on both the Beijing and the Hangzhou real-world datasets.

6.4 Summary on Experiment Results

We finally summarize our findings.

- Both algorithms based on the two-step framework are efficient in time and space, and they also visit more edges of the bipartite graph due to prediction, which results in a larger matching size.
- POLAR-OP is the most effective among all the algorithms and is also efficient.
- Both algorithms based on the two-step framework are scalable.
- The proposed prediction-based framework also works in practice on real-world datasets.



(a) Matching size of varying μ in temporal distribution

(b) Matching size of varying σ in temporal distribution

(c) Matching size of varying mean in spatial distribution

(d) Matching size of varying cov in spatial distribution

(e) Time of varying μ in temporal distribution

(f) Time of varying σ in temporal distribution

(g) Time of varying mean in spatial distribution

(h) Time of varying cov in spatial distribution

(i) Memory of varying μ in temporal distribution

(j) Memory of varying σ in temporal distribution

(k) Memory of varying mean in spatial distribution

(l) Memory of varying cov in spatial distribution

Figure 6: Results on varying μ and σ in temporal distribution and mean and cov in spatial distribution

7. RELATED WORK

Our work is related to the research on task assignment in spatial data and online maximum bipartite matching.

7.1 Task Assignment in Spatial Data

The problem of task assignment in spatial data is also called spatial matching problem [27, 30], which aims to match two sets of objects based on their spatial locations, where different optimization goals on the distance between the matched objects are proposed, *e.g.*, total sum [27] and stable marriage [30]. However, these studies cannot perform dynamic task assignment in spatial data.

With the development of smartphones, the problem of task assignment in spatial data becomes one of the core issues in various applications in O2O services and spatial crowdsourcing [6, 9, 15, 16, 20, 21, 23, 24]. For example, [15] is the first work about task assignment on spatial crowdsourcing platforms, with the goal to maximize the number of assigned tasks. Although the aforementioned works study the problem of task assignment on spatial data, they mostly focus on *offline scenarios*, where the spatiotemporal information of all the tasks and workers is known in advance.

There are two closely related works [26, 28]. The main difference between our work and [28] is that this research only considers the one-sided online task assignment and does not support the two-sided online task assignment. The major difference between our work and [26] lies in the online models. In the online model of [26], a worker has to wait in place instead of moving to other places if no task is assigned to her/him immediately. Our work proposes a more flexible online model, which not only provides a 0.47-competitive ratio, twice better than that of [26], but also reduces the time complexity of processing each task/worker to $\mathcal{O}(1)$.

7.2 Online Bipartite Matching

The classic online bipartite matching problem has been widely studied [14]. The majority focuses on one-sided online matching. Different assumptions on the input models have been proposed to the following three categories:

Adversarial model. This model assumes that an adversary plays against an online algorithm by specifying the graph and the arrival order of nodes. The first work [14] is proposed with a competitive ratio of 0.632. In the two-sided online version, recent work [29] proposes an algorithm with a ratio of 0.526. Our problem is more general in the way that the workers are all flexible.

Random Order Model. In this model, the arriving order of nodes is randomly chosen while the graph can still be arbitrarily bad [10]. In the two-sided online version, recent work [26] proposes the TGOA algorithm with a ratio of 0.25. However, their problems are different from our FTOA and their algorithms cannot solve our problem.

I.I.D Model. The I.I.D model, first proposed by [7], assumes that there is an underlying distribution of the coming objects. Given the distribution as prior, they design a competitive algorithm with ratio of 0.729 [12]. However, these studies are used to address one-sided online matching, which is inapplicable to our problem.

8. CONCLUSION

In this paper, we propose *Flexible Two-sided Online task Assignment* (FTOA), a new problem of online task assignment in real-time spatial data for practical O2O applications, where workers are allowed to move around if no task is assigned immediately. To address the FTOA problem, the key insight is to leverage the big historical records on O2O platforms to predict future spatiotemporal distributions of tasks and workers, base on which we devise a novel two-step framework, which integrates offline prediction with online task assignment. Given predictions as an offline guide, we develop the POLAR algorithm, which conducts online task assignment based on the offline guide and yields a competitive ratio of 0.4. To deal with the cases where the number of the actual tasks/workers exceeds the predicted estimates, we propose the POLAR-OP algorithm, which achieves a competitive ratio of 0.47. We verify the effectiveness, efficiency and scalability of the proposed algorithms on both synthetic datasets and datasets from a large-scale real-time taxi-calling platform. We envision our work as a leap towards more flexible and intelligent online task assignment for a wide spectrum of practical O2O platforms.

Acknowledgment

Yongxin Tong's work is supported in part by National Grand Fundamental Research 973 Program of China under Grant 2014CB340300, NSFC Grant No. 61502021 and 71531001, and SKLSDE Open Program SKLSDE-2016ZX-13.

Lei Chen's work is supported in part by the Hong Kong RGC Project 16202215, Science and Technology Planning Project of Guangdong Province, China, No. 2015B010110006, NSFC Grant No. 61328202 and 61300031, Microsoft Research Asia Collaborative Grant and NSFC Guang Dong Grant No. U1301253.

9. REFERENCES

[1] Didi chuxing. *https://en.wikipedia.org/wiki/Didi_Chuxing*.

[2] Gigwalk. *http://www.gigwalk.com*.

[3] Grubhub. *https://www.grubhub.com/*.

[4] Uber. *https://www.uber.com/*.

[5] R. Ahuya, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[6] D. Deng, C. Shahabi, and U. Demiryurek. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *GIS 2013*.

[7] J. Feldman, A. Mehta, V. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *FOCS 2009*.

[8] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*.

[9] D. Gao, Y. Tong, J. She, T. Song, L. Chen, and K. Xu. Top-k teams recommendation in spatial crowdsourcing. In *WAIM 2016*.

[10] G. Goel and A. Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA 2008*.

[11] A. M. Hendawi and M. F. Mokbel. Predictive spatio-temporal queries: a comprehensive survey and future directions. In *MobiGIS 2012*.

[12] P. Jaillet and X. Lu. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research*, 2014.

[13] Z.-Q. Jiang, W.-J. Xie, M.-X. Li, B. Podobnik, W.-X. Zhou, and H. E. Stanley. Calling patterns in human communication dynamics. *Proceedings of the National Academy of Sciences*, 2013.

[14] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC 1990*.

[15] L. Kazemi and C. Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *GIS 2012*.

[16] L. Kazemi, C. Shahabi, and L. Chen. Geotrucrowd: trustworthy query answering with spatial crowdsourcing. In *GIS 2013*.

[17] Y. Li, Y. Zheng, H. Zhang, and L. Chen. Traffic prediction in a bike-sharing system. In *GIS 2015*.

[18] A. Mehta. Online matching and ad allocation. *Theoretical Computer Science*, 2012.

[19] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas. Predicting taxi-passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems*, 2013.

[20] L. Pournajaf, L. Xiong, V. Sunderam, and S. Goryczka. Spatial task assignment for crowd sensing with cloaked locations. In *MDM 2014*.

[21] J. She, Y. Tong, L. Chen, and C. C. Cao. Conflict-aware event-participant arrangement and its variant for online setting. *IEEE Transactions on Knowledge and Data Engineering, 2016*.

[22] J. Sun, D. Papadias, Y. Tao, and B. Liu. Querying about the past, the present, and the future in spatio-temporal databases. In *ICDE 2004*.

[23] H. To, G. Ghinita, and C. Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. *PVLDB 2014*.

[24] H. To, C. Shahabi, and L. Kazemi. A server-assigned spatial crowdsourcing framework. *ACM Transactions on Spatial Algorithms and Systems*, 2015.

[25] Y. Tong, J. She, B. Ding, L. Chen, T. Wo, and K. Xu. Online minimum matching in real-time spatial data: Experiments and analysis. *PVLDB 2016*.

[26] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen. Online mobile micro-task allocation in spatial crowdsourcing. In *ICDE 2016*.

[27] L. H. U, M. L. Yiu, K. Mouratidis, and N. Mamoulis. Capacity constrained assignment in spatial databases. In *SIGMOD 2008*.

[28] U. ul Hassan and E. Curry. A multi-armed bandit approach to online spatial task assignment. In *UIC 2014*.

[29] Y. Wang and S. C. Wong. Two-sided online bipartite matching and vertex cover: Beating the greedy algorithm. In *ICALP 2015*.

[30] R. C.-W. Wong, Y. Tao, A. W.-C. Fu, and X. Xiao. On efficient spatial matching. In *VLDB 2007*.

[31] K. Zhao, D. Khryashchev, J. Freire, C. Silva, and H. Vo. Predicting taxi demand at high spatial resolution: approaching the limit of predictability. In *ICBD 2016*.