Geoinformatica
https://doi.org/10.1007/s10707-019-00359-w

Two-sided online bipartite matching in spatial data: experiments and analysis

Yiming Li[1] · Jingzhi Fang[1] · Yuxiang Zeng[2] · Balz Maag[3] · Yongxin Tong[1] · Lingyu Zhang[4]

Received: 27 January 2019 / Revised: 3 April 2019 / Accepted: 16 April 2019 /
Published online: 08 May 2019

Abstract
With the rapid development of sharing economy and mobile Internet in recent years, a wide range of applications of the *Two-sided Online Bipartite Matching (TOBM)* problem in spatial data are gaining increasing popularity. To be specific, given a group of workers and tasks that dynamically appear in a 2D space, the TOBM problem aims to find a matching with the maximum cardinality between workers and tasks satisfying the spatiotemporal constraints. Many works have studied this problem, but the settings of their problems are different from each other. Moreover, no prior works have compared the performances of the algorithms tailored for different settings under a unified definition. As a result, there lacks a guideline for practitioners to adopt appropriate algorithms for various scenarios. To fill the blank in this field, we present a comprehensive evaluation and analysis of the representative algorithms for the TOBM problem in this paper. We first give our unified definition and then provide uniform implementations for all the algorithms. Finally, based on the experimental results on both synthetic and real datasets, we discuss the strengths and weaknesses of the algorithms in terms of short-term effect and long-term effect, which can be guidance on selecting appropriate solutions or designing new methods.

Keywords Online bipartite matching · Two-sided online · Spatial data

1 Introduction

With the rapid development of mobile Internet, there has been more and more study on spatiotemporal data mining [11, 21, 22], trajectory data processing [32–35] and query processing on spatiotemporal data [9, 10, 12–15, 23, 31]. And one of the fundamental research topics on spatial data in the two-sided online bipartite matching problem. Given a group of workers and tasks that dynamically appear on the platform, the *Online Bipartite Matching (OBM)* problem aims to find a maximum cardinality matching in the corresponding bipartite

✉ Yongxin Tong
yxtong@buaa.edu.cn

Extended author information available on the last page of the article.

graph of workers and tasks. In another word, the platform aims to maximize the total number of performed tasks. This problem has been widely studied in academia, e.g., database community [16, 17, 26, 43] and theory community [24, 25, 30, 46]. With the rapid development of sharing economy and mobile Internet in recent years, the OBM problem has also attracted much interest from the industry. The representative platforms include real-time taxi-dispatching platforms (e.g., Uber [3] and DiDi Chuxing [5]), online food delivery platforms (e.g., Seamless [2]), and spatial crowdsourcing platforms (e.g., Gigwalk [4]). In such applications, a large number of tasks dynamically appear on the platform and need to be assigned to the available workers based on the spatiotemporal information of both workers and tasks.

Some previous works (see survey [30]) study the OBM problem in *one-sided* online scenario, where only tasks (i.e., one side of the bipartite graph) appear on the platform dynamically. However, such works always assume that the workers appear on the platform at the very beginning, which is less practical in the aforementioned platforms. On the contrary, workers (i.e., the other side) also dynamically appear on the platform and leave after their deadlines. Inspired by real applications, many recent works [16, 24–26, 43] study the OBM problem in the *two-sided* online scenario, i.e., the *Two-sided Online Bipartite Matching (TOBM)* problem. Particularly, there are a wide range of applications of the TOBM problem and some representative works are as follows.

Task Assignment in Spatial Crowdsourcing [16, 26, 43]**.** The task assignment is one of the major challenges in spatial crowdsourcing [20, 26, 28, 36–41, 44, 45, 47]. In spatial crowdsoucring platforms like Gigwalk [4], both crowd-workers (i.e., workers) and micro-tasks (i.e., tasks) dynamically arrive and thus can represent the two sides of the bipartite graph. Besides, the assignments between workers and tasks are usually determined by the platform in real-time once they appear. As the goal of the platform is usually to maximize the number of total assignments, task assignment in spatial crowdsourcing corresponds to one of the solutions to TOBM problem.

Taxi Dispatching/Food Delivery [24, 43]**.** Taxi dispatching (e.g., Uber [3] and DiDi Chuxing [5]) and food delivery (e.g., Seamless [2]) are two representative services in the intelligent transport systems [29, 48]. In these applications, taxis and couriers can be modeled as workers, while passengers and food orders can be modeled as tasks. If a worker is able to perform a task under the spatiotemporal constraints, there will be an edge between their corresponding vertices in the bipartite graph. Similarly, all of them also dynamically arrive at the platform and the platform will dispatch suitable taxis (couriers) to pick up the passengers (food orders). In order to maximize the number of successfully served users, existing methods to the TOBM problem can also be applied in these intelligent systems.

Real Estate Agency [24]**.** In real daily life, the TOBM problem also exists in the real estate agency (e.g., RE/MAX [1]). Specifically, both landlords and tenants drop by dynamically in an online fashion during daytime. The tenants specify the type of apartments they prefer and the deadlines before which they expect to move in. The landlords also set deadlines for tenants before which they can visit the apartments. In this problem, both tenants and landlords can be represented as each side of vertices in a bipartite graph. If a tenant and a landlord mutually satisfy each other's conditions and deadline constraints, there will be an edge between their corresponding vertices. The real estate agency charges for each successful deal and thus aims to maximize the cardinality of the matching between landlords and tenants. Therefore, the real estate agency application can be addressed by the TOBM problem.

1.1 Motivation

To handle the two-sided online bipartite matching problems on spatial data, existing studies have proposed various algorithms to address the problems under different definitions. However, no existing work has compared the algorithms tailored for different settings under a unified definition. As a result, the performance of algorithms in different works cannot be compared directly and there lacks a guideline for practitioners to select appropriate algorithms for various scenarios. Prior works [16, 39] also have evaluated some other variants of online matching problems in spatial data. Specifically, Tong et al. [39] present an experimental comparison of the algorithms for the *one-sided online minimum bipartite matching* problem in spatial data. Cheng et al. [16] mainly comprehensively compare the algorithms in the batch based mode (i.e., *offline scenario*). Thus, both works focus on a less practical scenario instead of the *two-sided online* scenario. Moreover, neither work studies the TOBM problem, i.e., maximizing the cardinality of the matching.

In addition to the lack of a comprehensive evaluation of existing methods, two valuable evaluation metrics have always been ignored in existing works [16, 26, 39, 43].

- As deadline is a commonly used temporal constraint for tasks, the *average response time* of all the tasks should be considered as an evaluation metric. This factor is more practical and important in platforms like Uber [3] and Seamless [2], where the requesters of the tasks expect to be answered as soon as possible.
- Existing works mainly focus on the *short-term* (e.g., real dataset of 1 day) performances of the algorithms while the study of long-term (e.g., real dataset of 1 month) performances of the algorithms in practice is absent. As the spatiotemporal factors change over time, the stability of the matching policies should also be evaluated in terms of long-term effects.

1.2 Contributions

In this paper, we provide a fair experimental comparison study over existing algorithms of the TOBM problem. Specifically, the algorithms include Greedy, Batch-GR, Batch-LLEP, Batch-NNP, Random, ext-Ranking and POLAR-OP. Greedy assigns the new arrival worker (task) to the currently nearest unmatched task (worker). The three batch based algorithms divide the continuous time into a number of time instances and try to achieve the maximum cardinality in each batch. Random will randomly assign an unmatched neighbor to every new arrival worker or task. Ext-Ranking samples a value to represent the rank for each new arrival object, and then determines the allocation between workers and tasks when the time reaches their deadlines. The basic idea of POLAR-OP is that the new arrival workers can be guided by the platform to move to the places where future tasks may appear.

We show their strengths and weaknesses according to the experimental results. Particularly, we consider the average response time of tasks as an important evaluation metric, which has always been ignored by prior works. Moreover, we conduct experiments on large real datasets to verify the long-term performance of the tested algorithms. We summarize our contributions as follows.

- We propose a general definition for the Two-sided Online Bipartite Matching (TOBM) problem in Section 2, which can be the foundation of the future studies in this area.
- We present uniform implementations for all the representative algorithms compared in our experimental study. These implementations adopt common basic operations

and thus offer a base for the comparison of future works in this area. Furthermore, the datasets and source code used in the experiments are available in [7]. In addition to the uniform implementations, we conduct experiments not only on the synthetic datasets but also on the large real datasets to have a comprehensive study of the tested algorithms.
– We discuss the advantages and disadvantages of the representative algorithms based on the experimental evaluation in Section 4, which could provide a guideline for practitioners to select appropriate algorithms for various scenarios.

The rest of the paper is organized as follows. In Section 2, we formally define the TOBM problem and then introduce two widely used analysis models for its online solutions. In Section 3, we introduce and discuss the representative *deterministic* algorithms and *randomized* algorithms. In Section 4, we systematically evaluate and analyze the existing methods according to the experimental results on both synthetic and real datasets. We finally conclude this paper in Section 5.

2 Preliminaries

In this section, we first formally define the Two-sided Online Bipartite Matching (TOBM) problem (Section 2.1) and then introduce two widely used theoretical analysis models for the online algorithms (Section 2.2): the *adversarial order* model and the *i.i.d* model.

2.1 Problem definition

We first introduce some basic concepts and then formally define the Two-sided Online Bipartite Matching (TOBM) problem.

Definition 1 (**Task**) A task, denoted by $t = \langle l_t, s_t, d_t \rangle$, appears on the platform at the location l_t in the 2D space at time s_t and it needs to be answered within d_t time (i.e., deadline) after its arrival, otherwise it will expire.

In two-sided markets (platforms), tasks are dynamically submitted to the platforms by the requesters. In practice, the response to the requesters (tasks) can be either *acceptances* or *rejections*. An acceptance indicates that there will be a worker assigned to the task. On the contrary, a rejection indicates that the platform decides not to assign any worker. The possible reason for a rejection could be lack of workers, unachievable deadlines, etc.

Definition 2 (**Worker**) A worker, denoted by $w = \langle l_w, s_w, d_w, r_w \rangle$, appears on the platform at the **initial** location l_w in the 2D space at time s_w. He/She is expected to answer (accept) a task within d_w time (i.e., deadline) after his/her arrival, otherwise he/she will leave the platform. Besides, he/she can only perform the task in a restricted circular range of w, which takes the current location of w as the center and r_w as the radius.

In a two-sided market, workers also dynamically appear on the platforms. The major existing works [16, 24–26] usually assume that the worker holds still after his/her appearance. However, a *flexible* worker can be guided to the locations with potentially high demands (i.e., the number of tasks). Tong et al. [43] first consider this practical issue and *flexible* workers can perform the tasks far away from their initial locations (l_w). Specifically,

the flexible workers do not wait at their initial locations after their appearance. Instead, the platform can guide the workers to move to the areas where future tasks may appear.

Note that we have abused the term deadline as in [43]. For the sake of simplicity, we refer to it as the maximum amount of time that a worker or a task can wait to be answered instead of a specific timestamp. Based on the aforementioned basic definitions, we define the Two-sided Online Bipartite Matching (TOBM) problem as follows.

Definition 3 (**TOBM problem**) Given a set of workers W and a set of tasks T, where both workers and tasks dynamically appear on the platform (i.e., **two-sided online** scenario), the TOBM problem is to find a matching $\mathcal{M}$ between W and T to maximize the cardinality of the assignment (i.e., $\text{OBJ}(\mathcal{M}) = |\mathcal{M}|$), such that the following constraints are satisfied:

- **Range Constraint**: Any task assigned to a worker w must be located in the restricted circular range of w.
- **Deadline Constraint**: Every task-worker pair $(t, w) \in \mathcal{M}$ should satisfy the deadline requirements of both the worker and the task. *(1)* After the task t appears on the platform, the worker (i.e., w) assigned to this task should be determined within its deadline d_r. *(2)* After the worker w appears on the platform, the task (i.e., t) allocated to this worker should be determined within his/her deadline d_w.
- **Invariable Constraint**: Once a task t is assigned to a worker w, the allocation of (t, w) cannot be changed.

We next illustrate the TOBM problem with a toy example as follows.

Example 1 Suppose there are four workers (taxis) w_1-w_4 and four tasks (taxi-calling requests) t_1-t_4 in a real-time taxi-dispatching platform. The goal of the platform is to maximize the number of completed tasks (i.e., the cardinality of the assignment between tasks and workers). The initial locations of both tasks and workers are labeled in a $2D$ space

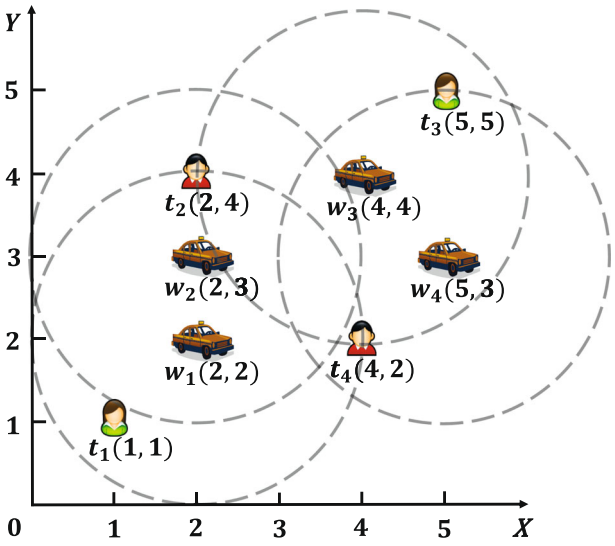


Fig. 1 An example of the TOBM problem

(X, Y) in Fig. 1. For the sake of simplicity, we assume that the restricted spatial range of each worker is a circular area with the radius of 2 unit distance as shown in Fig. 1. Table 1 shows the arrival order, arrival times and deadlines of workers and tasks.

In the two-sided online scenario, all the workers and tasks dynamically appear on the platform and should be matched before the corresponding deadline. The procedure can be further illustrated with the strategy Greedy, which will be discussed later in Section 3. Greedy always tries to allocate each new-arriving worker (task) to its currently nearest unmatched task (worker). For example, when w_1 arrives at 9:03, both t_1 and t_2 are within the spatial range of w_1. The platform then immediately assigns w_1 to t_1 as the distance of the pair (w_1, t_1) is shorter than that of (w_1, t_2). Similarly, the platform assigns w_2 to t_2 at 9:03, t_3 to w_3 at 9:05 and t_4 to w_4 at 9:06. Consequently, the cardinality of the assignment is 4.

2.2 Competitive analysis models

The performance of online algorithms is usually compared with the optimal allocation in the offline scenario and is mainly affected by the arriving orders of both tasks and workers. In the following, we introduce the evaluation standard *competitive ratio (CR)* for the online algorithms under two different analysis models: *adversarial order* model and *i.i.d* model.

Similar to the definition of the *approximation ratio* which is utilized to evaluate the approximation algorithms, the *competitive ratio* measures how good an online algorithm is compared with the optimal result of the offline scenario where all the information is provided in the beginning. There are two widely used competitive analysis models in existing works [30]: the *adversarial order* model and the *i.i.d* model, which focus on the worst-case input and the stochastic input of both tasks and workers, respectively. The corresponding competitive ratios under these two models are defined as follows.

Definition 4 (**CR under the adversarial order model**) Under the adversarial order (AO) model, the competitive ratio of an online algorithm for the TOBM problem is defined as follows:

$$CR_{AO} = \min_{(W,T)} \frac{\text{OBJ}(\mathcal{M})}{\text{OBJ}(OPT)} \tag{1}$$

where workers and tasks can have arbitrary initial locations and arrival orders respectively, $\text{OBJ}(\mathcal{M})$ is the cardinality of the matching produced by the online algorithm and $\text{OBJ}(OPT)$ is the cardinality of the optimal assignment in the offline scenario.

Table 1 Information of workers and tasks

Arrival order	Arriving time	Deadline
t_1	9 : 00	6
t_2	9 : 01	4
w_1	9 : 02	10
w_2	9 : 03	10
w_3	9 : 04	10
w_4	9 : 04	10
t_3	9 : 05	4
t_4	9 : 06	2

Definition 5 (**CR under the I.I.D model**) Under the i.i.d (IID) model, the competitive ratio of an online algorithm for the TOBM problem is defined as follows:

$$CR_{IID} = \min_{(W,T) \text{ follow } \mathcal{D}_W \text{ and } \mathcal{D}_T} \frac{\mathbb{E}[\text{OBJ}(\mathcal{M})]}{\mathbb{E}[\text{OBJ}(OPT)]} \tag{2}$$

where $\mathcal{D}_W$ and $\mathcal{D}_T$ are the spatiotemporal distributions of workers and tasks, $\mathbb{E}[\text{OBJ}(M)]$ is the expected cardinality of the matching produced by the online algorithm and $\mathbb{E}[\text{OBJ}(OPT)]$ is the expected cardinality of the optimal assignment in the offline scenario.

According to their definitions, the *i.i.d* model assumes that tasks and workers are independently and identically distributed while the *adversarial order* model has no such assumption. Specifically, under the adversarial order model, workers and tasks have arbitrary initial locations and arrival orders. Under the i.i.d model, the initial locations of workers and their arrival orders follow the spatiotemporal distribution $\mathcal{D}_W$, and the initial locations of tasks and their arrival orders follow the spatiotemporal distribution $\mathcal{D}_T$. Thus, if an online algorithm is ρ-competitive under the *adversarial order* model, it is also ρ-competitive under the *i.i.d* model.

3 Algorithms

In this section, we introduce the main ideas of the representative online algorithms evaluated in our experimental study. These algorithms can be mainly classified into two groups, *deterministic algorithms* (Section 3.1) and *randomized algorithms* (Section 3.2).

3.1 Deterministic algorithms

In the following, we introduce two deterministic algorithms, Greedy [25] and Batch [26].

Algorithm 1 Greedy.

input : a set of workers W, a set of tasks T
output: An assignment $\mathcal{M}$ between W and T

$\mathcal{M} \leftarrow \emptyset$;
foreach *new arrival task or worker* v **do**
  $Cand \leftarrow \{\forall u \mid u$ is an unmatched neighbor of v such that matching u to v satisfies all constraints$\}$;
  if $Cand \neq \emptyset$ **then**
    match v to the nearest neighbor in $Cand$;
    update $\mathcal{M}$;
  else
    let v wait to be matched until the deadline;
return $\mathcal{M}$;

3.1.1 Greedy

Greedy is first proposed by Karp et al. in [25], which is the first work to study the *online bipartite matching* problem. The **basic idea** of Greedy is to assign the new arrival worker (task) to the currently nearest unmatched task (worker).

Algorithm 1 shows the procedure of Greedy. Whenever a new object (i.e., a worker or a task) appears on the platform, Greedy first initiates a candidate set ($Cand$) of currently unmatched adjacent objects satisfying all the constraints (line 3), and then selects its nearest neighbor from the candidate set as the matching result (lines 4–6). If there is no such feasible candidate, the new arrival object will wait to be matched until its deadline (lines 7–8). The **competitive ratio** of Greedy is 0.5 under the *adversarial order* model, which is known as the barrier to break in the *online bipartite matching* problem [24, 25]. To further illustrate the Greedy algorithm, we go through the following example.

Example 2 Back to our running example in Example 1. When w_1 arrives at 9:02, both t_1 and t_2 are within the spatial range of w_1. The platform then immediately assigns w_1 to t_1 as the distance of the pair (w_1, t_1) is shorter than that of (w_1, t_2). Similarly, the platform assigns w_2 to t_2 at 9:03, t_3 to w_3 at 9:05, and t_4 to w_4 at 9:06. Consequently, the cardinality of the assignment is 4.

3.1.2 Batch

As aforementioned, task assignment in spatial crowdsourcing is one of the representative applications of the *online bipartite matching* problem. Kazemi et al. [26] first propose the challenge of task assignment in spatial crowdsourcing. They propose a *batch* based framework to solve the problem, which is further widely adopted in other works [16, 17].

The **basic idea** of the framework is to divide the continuous time into a number of time instances (i.e., batches) and tries to achieve the maximum cardinality in each batch. Specifically, the tasks and workers appear on the platform within a batch should wait to be matched at the end of the batch. In each batch, the task assignment problem can be reduced to the bipartite matching problem. Thus, the Hungarian algorithm [27] can be used to obtain the assignment with the maximum cardinality in this batch. Kazemi et al. [26] name this procedure as GR and they also consider other practical issues such as *location entropy* and *moving distance of workers*.

Least Location Entropy Priority (LLEP) [26] tries to improve the task assignment by exploiting the spatial characteristics of both tasks and workers. Since the tasks located in the areas with higher worker densities are already very likely to be performed, it assigns higher priority to the tasks located in the areas with fewer workers around to improve the chance of them being matched. Nearest Neighbor Priority (NNP) [26] considers the average moving distance of workers. Similarly, the worker closer to the task has the higher priority to be matched to the task. In the bipartite graph, the priority can be represented as the weight of the edge. Thus, Kazemi et al. [26] reduce this variant of the task assignment problem to the *minimum cost maximum flow* problem or the *minimum cost bipartite matching* problem, which can be solved by the Edmonds–Karp algorithm [19] or the Kuhn–Munkres algorithm [8]. Even though Batch is a heuristic algorithm without any theoretical guarantee, the prior work [16] has validated its effectiveness and efficiency.

3.2 Randomized algorithms

In the following, we introduce three randomized algorithms for the TOBM problem, Random [25], ext-Ranking [24] and POLAR-OP [43].

3.2.1 Random

The Random algorithm is also devised by Karp et al. in [25]. Its **basic idea** is similar to the Greedy algorithm, i.e., the algorithm will assign an unmatched neighbor (if exists) to every new arrival worker or task. The only difference is that the unmatched neighbor is sampled *uniformly and randomly* from all the candidates in the Random algorithm. However, the randomness of this algorithm does not improve its theoretical guarantee since the **competitive ratio** is still 0.5 under the *adversarial order* model.

3.2.2 ext-Ranking

In order to beat the Greedy algorithm with a higher competitive ratio, Huang et al. [24] first propose the ext-Ranking algorithm for the TOBM problem. The ext-Ranking is extended from the Ranking algorithm, which is first devised by Karp et al. [25]. The **basic idea** is to sample a value to represent the rank for each new arrival object (a worker or a task), and then determine the allocation between the workers and the tasks only at their deadlines.

The procedure of ext-Ranking is illustrated in Algorithm 2. Whenever a new object v (i.e., a worker or a task) appears on the platform, ext-Ranking will pick a value (denoted by y_v) for it from [0,1) uniformly and randomly (lines 2–3). When the time reaches the deadline of v' and if it remains unmatched, ext-Ranking first initiates a candidate set ($Cand$) of currently unmatched adjacent objects satisfying all the constraints (line 5), and then selects the one with the minimum value (i.e., the highest rank) as the matching result (lines 6–8). If there is no such feasible candidate set, the object v' will be rejected due to its expired time (lines 9–10). Besides, all the unmatched tasks and workers will wait until their deadlines.

The **competitive ratio** of ext-Ranking is 0.554 under the *adversarial order* model, which is better than the Greedy and the Random algorithm. To further illustrate the ext-Ranking algorithm, we go through the following example.

Algorithm 2 ext-Ranking.

```
   input  : a set of workers W, a set of tasks T
   output: An assignment M between W and T
1  M ← ∅;
2  foreach new arrival task or worker v do
3      pick a value (denoted by y_v) for v from [0, 1) uniformly and randomly;
4      if any unmatched task or worker v' reaches the deadline then
5          Cand ← {∀u | u is an unmatched neighbor of v' such that matching u to v'
           satisfies all constraints};
6          if Cand ≠ ∅ then
7              match v' to the neighbor with the highest rank, i.e., arg min_{u∈Cand} y_u;
8              update M;
9          else
10             let other tasks and workers wait until the deadline;
11 return M;
```

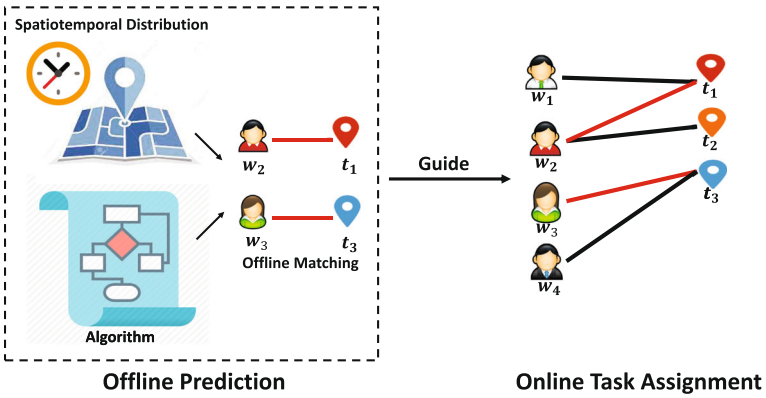


Fig. 2 The framework of POLAR-OP

Example 3 Back to our running example in Example 1. Before 9:05, two tasks t_1-t_2 and four workers w_1-w_4 appear on the platform one by one with the randomly picked values, i.e., $t_1(0.2)$, $t_2(0.4)$, $w_1(0.5)$, $w_2(0.6)$, $w_3(0.2)$ and $w_4(0.4)$. At 9:05, the time reaches the deadline of t_2 and t_2 is within the spatial range of three worker w_1-w_4. The platform then assigns t_2 to w_3 as it has the highest rank (i.e., the lowest value). At 9:06, the time reaches the deadline of t_1, and the platform assigns it to w_1. At 9:08, the time reaches the deadline of t_4, and the platform assigns it to w_4. At 9:09, the time reaches the deadline of t_3 and it becomes expired as it is not within the spatial range of the only available worker w_2. Consequently, the cardinality of the assignment is 3.

3.2.3 POLAR-OP

Tong et al. [43] devise the POLAR-OP algorithm for the Flexible Two-sided Online Task Assignment (FTOA) problem, which is a variant of the TOBM problem. All the previous algorithms assume that the workers stay in the same place and wait to be matched before their deadline. Differently, the **basic idea** of POLAR-OP is that the new arrival workers can be guided by the platform to move to the places where future tasks may appear.

Specifically, Tong et al. [43] propose a two-step framework (see Fig. 2), which consists of *offline prediction* and *online task assignment*. The framework aims to guide the online algorithm by the offline solution (i.e., offline-guide-online).

In the *offline prediction* step, the framework divides the 2D space into multiple grids and the continuous time into uniform time slots. Moreover, the spatiotemporal distributions of the subsequent workers and tasks (i.e., the number of workers and tasks respectively in each grid and each time slot) can be predicted according to the historical data [42]. Based on the predicted numbers of workers and tasks, the algorithm instantiates the same number of vertices on the left (workers $\widehat{W}$) and right (tasks $\widehat{T}$) of a bipartite graph $\widehat{G}$. Finally, the maximum cardinality matching $\widehat{\mathcal{M}}$, i.e., *offline guide*, can be acquired by the classical *offline* bipartite matching algorithms such as the Hungarian algorithm [27] or Dinic's algorithm [18].

In the *online task assignment* step, based on the offline matching ($\widehat{\mathcal{M}}$), POLAR-OP can guide the movement of flexible workers to maximize the potential number of assignments. They introduce the concept *type*. If a worker (task) has the same *type* as another worker

(task), they must appear in the same area (grid) within the same time slot. Algorithm 3 illustrates the procedure of POLAR-OP.

Algorithm 3 POLAR-OP.

input : a set of workers W, a set of tasks T, the offline guide $\widehat{\mathcal{M}}$
output: An assignment $\mathcal{M}$ between W and T

1 $\mathcal{M} \leftarrow \emptyset$;
2 **foreach** *new arrival object* v **do**
3 **if** v *is a worker* **then**
4 **if** $\widehat{\mathcal{M}}$ *exists a worker vertex with the same type as* v **then**
5 $(w, \widehat{\mathcal{M}}_w) \leftarrow$ uniformly sampled from $\widehat{\mathcal{M}}$, where w has the same type as v;
6 $Cand \leftarrow$ all unmatched tasks with the same type as $\widehat{\mathcal{M}}_w$ satisfying all constraints;
7 **if** $Cand \neq \emptyset$ **then**
8 $t \leftarrow$ randomly chosen from $Cand$;
9 assign worker v to perform task t and update $\mathcal{M}$;
10 **else**
11 dispatch worker v to the grid of $\widehat{\mathcal{M}}_w$;
12 **else**
13 **if** $\widehat{\mathcal{M}}$ *exists a task vertex with the same type as* v **then**
14 $(\widehat{\mathcal{M}}_t, t) \leftarrow$ uniformly sampled from $\widehat{\mathcal{M}}$, where t has the same type as v;
15 $Cand \leftarrow$ all unmatched workers with the same type as $\widehat{\mathcal{M}}_t$ satisfying all constraints;
16 **if** $Cand \neq \emptyset$ **then**
17 $w \leftarrow$ randomly chosen from $Cand$;
18 assign worker w to perform task v and update $\mathcal{M}$;
19 **else**
20 let v wait until its deadline;
21 **else**
22 reject v immediately;
23 **return** $\mathcal{M}$;

– When a worker v appears, POLAR-OP first selects a worker vertex w with the same type as v in the guidance $\widehat{\mathcal{M}}$, which is assigned to task $\widehat{\mathcal{M}}_w$ in the offline matching (lines 4–5). If there exist unmatched tasks ($Cand$) which have the same type as $\widehat{\mathcal{M}}_w$ and satisfy all constraints, the algorithm will uniformly sample a task for the new arrivel worker (lines 6–9). Otherwise, the new arrival worker v will be guided to move to the grid of $\widehat{\mathcal{M}}_w$ (lines 10–11).
– When a task v appears, POLAR-OP first selects a task vertex t with the same type as v in the guidance $\widehat{\mathcal{M}}$, which is allocated to worker $\widehat{\mathcal{M}}_t$ in the offline matching (lines 13–14). If there exist unmatched workers ($Cand$) that have the same type as $\widehat{\mathcal{M}}_t$ and satisfy all constraints, a worker will be uniformly sampled to perform the new arrival

Table 2 Comparisons of representative studies on the TOBM problem

Algorithm	Randomization	Time complexity	Analysis model	Ratio
Greedy [25]	Deterministic	$O(\max\{\|W\|, \|T\|\}^2)$	Adversarial Order	0.5
Batch-GR [26]	Deterministic	$O(\|B\| \cdot \max\{\|\mathcal{W}\|, \|\mathcal{T}\|\}^3)$[a]	–	Heuristic
Batch-LLEP [26]	Deterministic	$O(\|B\| \cdot \max\{\|\mathcal{W}\|, \|\mathcal{T}\|\}^3)$	–	Heuristic
Batch-NNP [26]	Deterministic	$O(\|B\| \cdot \max\{\|\mathcal{W}\|, \|\mathcal{T}\|\}^3)$	–	Heuristic
Random [25]	Randomized	$O(\max\{\|W\|, \|T\|\}^2)$	Adversarial Order	0.5
ext-Ranking [24]	Randomized	$O(\max\{\|W\|, \|T\|\}^2)$	Adversarial Order	0.554
POLAR-OP [43]	Randomized	$O(\max\{\|W\|, \|T\|\}^2)$	I.I.D	0.47

[a] $|B|$ denotes the number of batches, $|\mathcal{W}|$ and $|\mathcal{T}|$ denote the maximum number of workers and tasks among all batches respectively

task (lines 15–18). If the offline guidance $\widehat{\mathcal{M}}$ contains no such type as v, the algorithm can reject the task immediately (lines 21–22).

In the POLAR-OP algorithm, workers and tasks are assumed to be independently and identically distributed, which can be predicted by existing learning methods [42, 48]. Thus, Tong et al. [43] analyze its competitive ratio to be 0.47 under the *i.i.d* model.

3.3 Summary

Table 2 summarizes all the aforementioned online algorithms evaluated in our experimental study. In terms of competitive ratios, the ext-Ranking algorithm achieves a better theoretical guarantee than the other algorithms. Among all the algorithms, POLAR-OP has the assumption about the spatiotemporal distributions of workers and tasks with the smallest competitive ratio.

4 Experimental study

In this section, we introduce our experimental settings and then discuss the performances of the representative algorithms for the TOBM problem.

4.1 Experiment setup

Datasets We conduct experiments on both synthetic datasets and real datasets to evaluate the performance of the tested algorithms.

Synthetic datasets We generate the locations of workers and tasks on a 2D space with 200 × 200 grids, where each grid represents a 100m×100m square region. We study the effects of spatial distributions by randomly generating the locations of workers and tasks following the commonly used exponential and normal distributions [16, 39], as different distributions may affect the matching results in spatial data [39, 43]. We assume that both locations of workers and tasks follow the similar distribution, which is reasonable in real world as the workers often go to the areas where potential tasks may appear. For normal distributions, we set the variance as 15. Besides, we also vary the number of tasks, the number of workers, the deadline of tasks and workers, and the radius of restricted circular

Table 3 Parameter settings for synthetic datasets

Parameter	Settings
Mean μ of locations following normal distribution	50,75,**100**,125,150
Mean $1/\lambda$ of locations following exponential distribution	50,75,**100**,125,150
Number of task$\|T\|$	6000,8000,**10000**,12000,14000
Number of worker $\|W\|$	6000,8000,**10000**,12000,14000
Deadline of task d_t (min)	1,1.5,**2**,2.5,3
Deadline of worker d_w (min)	2,2.5,**3**,3.5,4
Radius of the restricted circular range of worker r_w (km)	0.6,0.8,**1**,1.2,1.4

range of workers. Table 3 illustrates the settings in detail and the default settings are marked in bold. Specifically, we set the deadline of tasks to 1 to 3 min. It is reasonable since in some applications, like car-hailing services, the requester will leave the platform if his/her request is not answered within several minutes.

Real datasets We use the open datasets [6] collected by the largest taxi dispatching platform DiDi Chuxing [5] in China. The datasets contain the taxi requests from 01/11/2016 to 30/11/2016 in the urban area of Chengdu, China (with latitude from 30.65° to 30.73° and longitude from 104.04° to 104.13°), including the information of spatial locations, arrival time, etc. Specifically, we use the pickup time and locations of taxi-calling orders as the arriving time and locations of tasks. We also use the drop-off time and locations of taxi-calling orders as the arriving time and locations of workers. As the information of radius of the range of workers is not included in the dataset, we vary this parameter to test the short-term performance. In order to test the long-term performance, we conduct experiments on real taxi requests from 6th to 30th. Table 4 illustrates the settings in detail and the default setting is marked in bold. As we need the datasets of first five days to predict the spatiotemporal distributions for the POLAR-OP algorithm, we only test the long-term performance with datasets of the other 25 days. Since there are 1,840,228 tasks and 1,840,228 workers in the long-term test, this test can also reflect the scalability of the algorithms.

Evaluation metrics In the experiments, we use the following metrics to evaluate the effectiveness and efficiency of the existing methods.

- **Matching Size.** The matching size represents the total number of assigned pairs, i.e., the objective of the TOBM problem.
- **Average Response Time of Tasks.** (ARTT for short) The response time of a task refers to the time it takes for t to be answered by the platform (either accept or reject). If a task is evetually not assigned to any worker, its response time will equal to its deadline.

Table 4 Parameter settings for real datasets

Parameter	Settings
Number of tasks $\|W\|$	82,171
Number of workers $\|T\|$	82,171
Radius of the restricted circular range of worker r_w (km)	0.5,1,**1.5**,2,2.5
The number of days for long-term test	1, 2, 3, $\cdots$, 24, 25

In practice, ARTT is an important indicator to reflect the satisfaction of the requesters. As a result, lower ARTT usually indicates better and faster service.
- **Running Time.** The running time represents the total execution time of an algorithm for solving the TOBM problem.
- **Memory usage.** The memory usage represents the peak space cost of an algorithm during the test. As the memory usages of all compared algorithms are below 100 MB, we omit their memory costs in the following contents since they are all efficient in terms of memory usage.

Compared algorithms and implementation We evaluate the performance of the representative algorithms, i.e., Greedy (Section 3.1.1), batch based algorithms including Batch-GR,Batch-LLEP,and Batch-NNP (Section 3.1.2), Random (Section 3.2.1), ext-Ranking (Section 3.2.2), and POLAR-OP (Section 3.2.3).

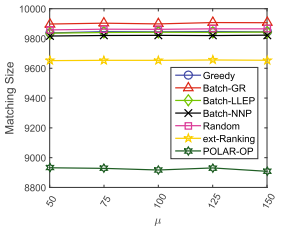
For the *batch based* algorithms, we set the batch size as 30 seconds, which could strike a balance between the matching result and the waiting time of the requesters. For the *POLAR-OP* algorithms, we set the speed of workers to be 36 km/h, which is about the average speed of motor vehicles in cities. Besides, We use two methods to predict the spatiotemporal distributions: (1) In the experiments on real datasets, we use the techniques recommended in [43] to learn the spatiotemporal distributions. (2) In the experiments on synthetic datasets, we generate the predicted distribution by adding noises. First, we generate the 2D locations of workers and tasks based on a specific distribution. These locations are considered as the real distribution of workers and tasks. Then, we randomly sample 20% objects from the real distribution and add noises to their coordinates. Specifically, for a sampled point, we randomly add 200m or -200m to its x-coordinate and y-coordinate respectively. Finally, we combine the 20% noisy points with the left 80% points as the predicted distribution. The reason is that the synthetic datasets are generated based on a specific distribution and thus the learning techniques can easily learn the distribution, which is extremely hard in real datasets.

The compared algorithms are all implemented in GNU C++. And the experiments are conducted on a server with 40 Intel(R) Xeon(R) E5 2.30GHz processors. Each experiment is repeated 30 times and the average results are reported.

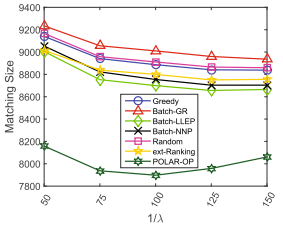
4.2 Experiment results

4.2.1 Experiments on synthetic datasets

Effect of the spatial distributions Figure 3 shows the effect of normal distribution (Fig. 3a to c) and exponential distribution (Fig. 3d to f) by varying the mean. We can observe that the mean of the normal distribution barely affects the matching size as it only changes the center of the distribution. However, with the increase of the mean in the exponential distributions, the matching size decreases since the locations of workers and tasks are more diversified so that there are potentially fewer tasks located in the spatial range of workers. Batch-GR outperforms all other algorithms in terms of matching size (i.e., objective) while POLAR-OP performs the worst due to the inaccurate predictions. As for ARTT, Greedy and Random outperform the other algorithms since they always try to assign a worker or a task the moment it appears. However, the ARTT of ext-Ranking is the highest as it determines the allocation between the workers and tasks only at their deadlines. In terms of the running time, the time costs of the batch based algorithms, Batch-GR, Batch-LLEP and Batch-NNP, are obviously higher than the others due to their high time complexities in each batch.



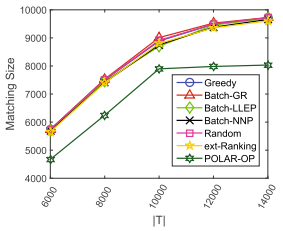
(a) Matching size of varying μ (b) ARTT of varying μ (c) Running time of varying μ



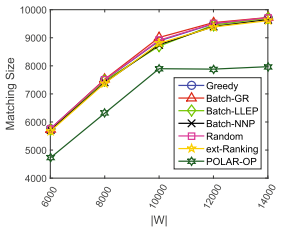
(d) Matching size of varying $1/\lambda$ (e) ARTT of varying $1/\lambda$ (f) Running time of varying $1/\lambda$

Fig. 3 Results on varying the mean of spatial distributions of the workers and tasks

Effect of the number of tasks $|T|$ Figure 4a to c show the results of varying the number of tasks. In Fig. 4a, the matching sizes of all the tested algorithms increase with the number of tasks, as there are potentially more tasks appearing in the restricted spatial range of workers. Besides, almost all the algorithms achieve similar performances in terms of the matching size while POLAR-OP still performs the worst due to the same aforementioned reason. In terms of ARTT, ext-Ranking is still the worst while both Greedy and Random are better

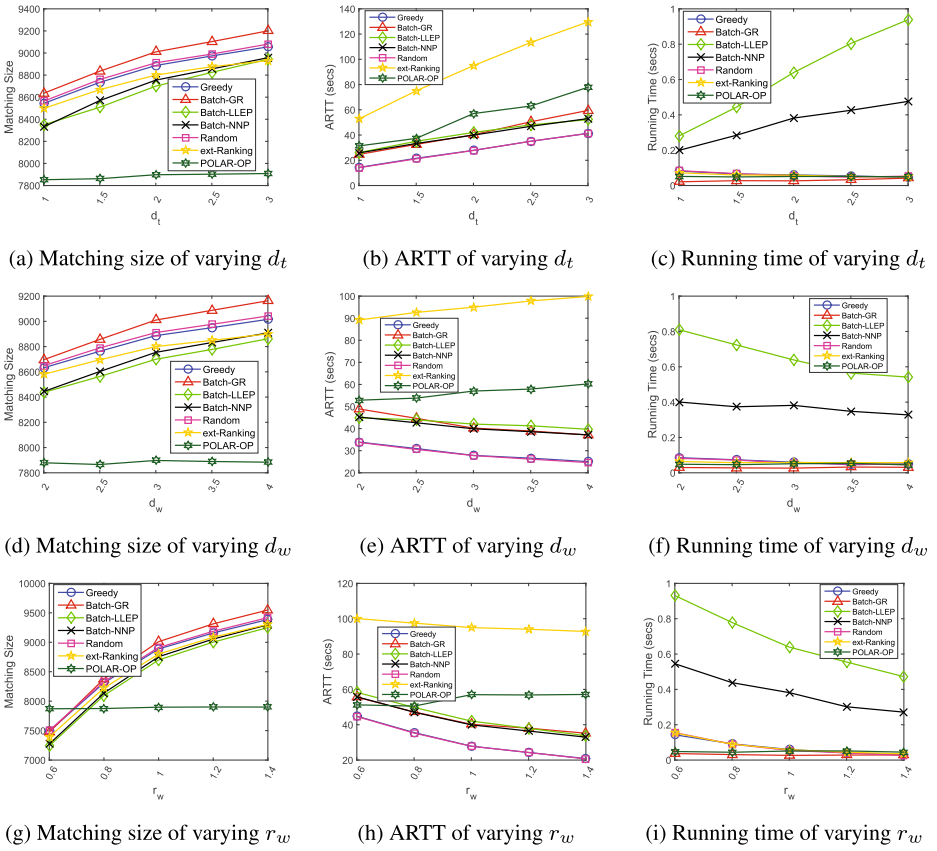


(a) Matching size of varying $|T|$ (b) ARTT of varying $|T|$ (c) Running time of varying $|T|$



(d) Matching size of varying $|W|$ (e) ARTT of varying $|W|$ (f) Running time of varying $|W|$

Fig. 4 Results on varying the number of the tasks and workers



(a) Matching size of varying d_t (b) ARTT of varying d_t (c) Running time of varying d_t

(d) Matching size of varying d_w (e) ARTT of varying d_w (f) Running time of varying d_w

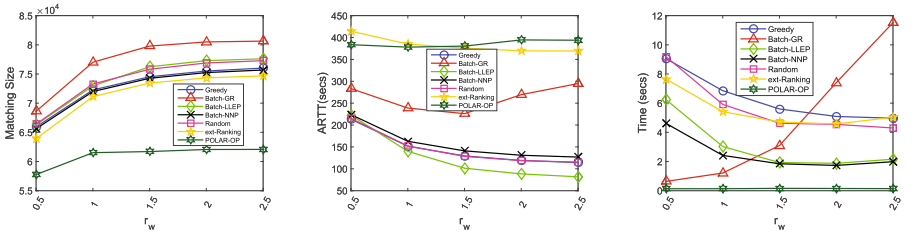
(g) Matching size of varying r_w (h) ARTT of varying r_w (i) Running time of varying r_w

Fig. 5 Results on varying the deadlines, and the restricted circular range of workers

than the others. Besides, the ARTT for all the algorithms increases with the number of tasks because more tasks can not be answered before deadlines. In terms of running time, both Batch-LLEP and Batch-NNP take more time than the others.

Effect of the number of workers $|W|$ Figure 4d to f show the experimental results when varying the number of workers. In terms of objective (i.e., matching size), it shows a similar pattern as varying the number of tasks. As for ARTT of all the algorithms, it decreases when the number of workers increases, because more workers can respond to tasks more quickly. As for running time, POLAR-OP and Batch-GR are faster than the others.

Effect of the deadline of tasks d_t The results are presented on the first row of Fig. 5. In Fig. 5a, Batch-GR still outperforms the others in terms of matching size, following by Greedy and Random. In Fig. 5b, Greedy and Random are still the best while ext-Ranking is the worst. In terms of running time, Batch-GR is the fastest, followed by POLAR-OP, ext-Ranking, Greedy and Random. Batch-LLEP and Batch-NNP are clearly slower than the others.



(a) Matching size of varying r_w (b) ARTT of varying r_w (c) Running time of varying r_w

Fig. 6 Results on varying the restricted circular range of workers

Effect of the deadline of workers d_w The results are presented on the second row of Fig. 5. As for matching size, we can see a similar pattern as varying the deadline of tasks. In Fig. 5e, the ARTT of both ext-Ranking and POLAR-OP increases with the deadline. It is reasonable for ext-Ranking since the decisions of assignments are often made at the time of deadline. For POLAR-OP, it is also reasonable since the matching size of POLAR-OP does not increase. In Fig. 5f, we can observe that the running times of all algorithms decrease with the increase of deadlines. However, both Batch-LLEP and Batch-NNP are still the slowest.

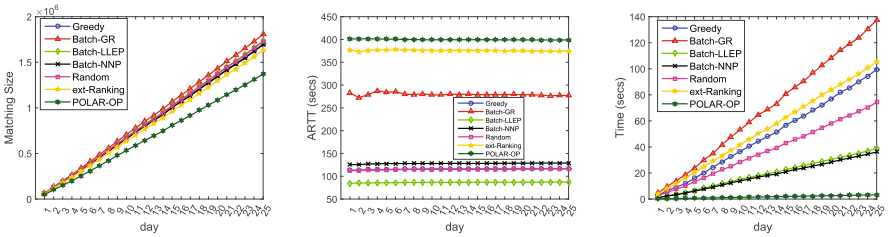
Effect of the range of workers r_w The results are presented on the last row of Fig. 5. In Fig. 5g, we can observe that matching sizes achieved by all the algorithms except POLAR-OP increase when the radius r_w increases. When the length of radius is short (e.g., 0.6), POLAR-OP outperforms the others. As for ARTT and running time, we can observe some similar patterns.

Based on the experiments on synthetic datasets, we conclude that all the evaluated algorithms except ext-Ranking are likely to satisfy the real-time requirement of the applications like car-hailing services. First, we believe that the average response time of no more than 1 min (the minimum deadline of tasks on synthetic datasets) is acceptable in most real-time applications like Uber [3]. According to the experimental results on synthetic datasets, we can see that the ARTT for all the algorithms except ext-Ranking is less than 1 min in most cases.

4.2.2 Experiments on Real datasets

Effect of the range of workers r_w Figure 6 shows the results of varying the restricted circular range of workers on the real datasets. In Fig. 6a, the matching sizes achieved by the algorithms first slightly increase and then keep stable when r_w increases from 1km to 3km. Moreover, Batch-GR still obviously outperforms other algorithms. In Fig. 6b, Batch-LLEP achieves the lowest ARTT, which is different from our observation on the synthetic dataset. POLAR-OP sometimes achieves higher average response time than ext-Ranking due to the smaller matching size. As for the running time, the time cost of Batch-GR increases drastically when r_w increases. On the contrary, POLAR-OP outperforms the others.

Long-term test We study the long-term test of the algorithms as shown in Fig. 7. In Fig. 7a, we observe that Batch-GR outperforms all the other algorithms in terms of matching size from the long-term perspective. Besides, we can see that the ranking of algorithms in descending order of matching size in Fig. 6a still holds in Fig. 7a, which means that the



(a) Matching size of varying days (b) ARTT of varying days (c) Running time of varying days

Fig. 7 Results on long-term test

performance in the short term still holds in the long term. Note that POLAR-OP is likely to have different short-term and long-term performances because the prediction accuracy of the spatiotemporal distribution in the short-term may be inconsistent with that in the long-term. However, the prediction accuracy seems to be relatively low in both short-term and long-term on real datasets so that POLAR-OP does not perform that well throughout the experiments. In Fig. 7b, we find that Batch-LLEP stably outperforms other algorithms in terms of ARTT.

As for running time, POLAR-OP is the fastest. However, Batch-GR becomes the slowest, which is contrary to the conclusion on synthetic datasets. The reason is as follows. The time complexity of Batch-GR is $O(|B| \cdot \max\{|\mathcal{W}|, |\mathcal{T}|\}^3)$, where $|B|$ denotes the number of batches, $|\mathcal{W}|$ and $|\mathcal{T}|$ denote the maximum number of workers and tasks among all batches respectively. The time complexity of other non-batch based algorithms is $O(\max\{|W|, |T|\}^2)$. Note that $|\mathcal{W}|$ is far smaller than $|W|$ and $|\mathcal{T}|$ is far smaller than $|T|$. (1) On synthetic datasets, there are only 120 batches (one hour). Batch-GR can outperform other algorithms in terms of running time because $\max\{|W|, |T|\}^2$ dominates $\max\{|\mathcal{W}|, |\mathcal{T}|\}^3$. (2) On real datasets, there are 2880 batches (one day). Batch-GR runs the slowest because $|B|$ is very large compared with that on synthetic datasets.

4.3 Experiment summary

We summarize our experimental findings as follows.

- In the short-term and long-term tests, Batch-GR outperforms other algorithms in terms of matching size.
- In the short-term test, Greedy and Random outperform other algorithms in terms of ARTT on synthetic datasets, while Batch-LLEP achieves the lowest ARTT in the long-term test. On the other hand, even though ext-Ranking achieves the highest competitive ratio, its average response time is usually the highest among all the algorithms. Moreover, we conclude that all the evaluated algorithms except ext-Ranking are likely to satisfy the real-time requirement of the applications like car-hailing services.
- In terms of running time, both Batch-LLEP and Batch-NNP are often the least efficient while POLAR-OP is often more efficient. In particular, POLAR-OP is the most efficient algorithm in the experiments on real datasets. Even though the matching size of POLAR-OP is relatively smaller, it can have better performance when the prediction becomes more accurate.

In summary, we recommend Batch-GR when the requirement of average response time is not strict. However, we recommend Greedy and Random when the requirement of average response time is strict. Finally, if the prediction of the spatiotemporal distribution of future tasks can be more accurate, we recommend the POLAR-OP algorithm since it may obtain a remarkable improvement in terms of matching size while remaining good efficiency.

5 Conclusion

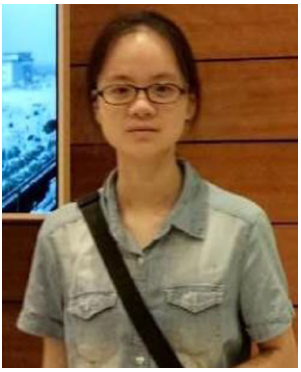
In this paper, we present a comprehensive experimental study of the representative algorithms for the *Two-sided Online Bipartite Matching (TOBM)* problem in spatial data. Specifically, we first propose a unified definition for the TOBM problem. Then we present uniform implementations for the state-of-the-art algorithms, and verify their effectiveness and efficiency on both synthetic and real datasets. Based on the experimental results, we discuss the strengths and weaknesses of the algorithms in terms of short-term effect and long-term effect, which can be a guidance for selecting appropriate solutions or designing new methods.

References

1. (1973) RE/MAX. http://www.remax.com
2. (1999) Seamless. https://www.seamless.com
3. (2009) Uber. https://www.uber.com
4. (2010) Gigwalk. http://www.gigwalk.com
5. (2012) DiDi Chuxing. https://www.didiglobal.com
6. (2016) GAIA Open Dataset. https://outreach.didichuxing.com/research/opendata
7. (2019) Source code and datasets. https://share.weiyun.com/5zX7DGs
8. Burkard RE, Dell'Amico M, Martello S (2009) Assignment problems. SIAM
9. Cao X, Chen L, Cong G, Jensen CS, Qu Q, Skovsgaard A, Wu D, Yiu ML (2012) Spatial keyword querying. In: ER, pp 16–29
10. Chen L, Cong G (2015) Diversity-aware top-k publish/subscribe for text stream. In: SIGMOD, pp 347–362
11. Chen L, Shi S, Lv J (2011) Efficient computation of measurements of correlated patterns in uncertain data. In: ADMA, pp 311–324
12. Chen L, Cui Y, Cong G, Cao X (2014) SOPS: a system for efficient processing of spatial-keyword publish/subscribe. PVLDB 7(13):1601–1604
13. Chen L, Cong G, Cao X, Tan K (2015) Temporal spatial-keyword top-k publish/subscribe. In: ICDE, pp 255–266
14. Chen L, Shang S, Zhang Z, Cao X, Jensen CS, Kalnis P (2018) Location-aware top-k term publish/subscribe. In: ICDE, pp 749–760
15. Chen Z, Cong G, Zhang Z, Fu TZJ, Chen L (2017) Distributed publish/subscribe query processing on the spatio-textual data stream. In: ICDE, pp 1095–1106
16. Cheng P, Jian X, Chen L (2018) An experimental evaluation of task assignment in spatial crowdsourcing. PVLDB 11(11):1428–1440
17. Deng D, Shahabi C, Demiryurek U (2013) Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In: GIS, pp 314–323
18. Dinitz Y (2006) Dinitz' algorithm: the original version and even's version. In: Theoretical computer science, essays in memory of Shimon even, pp 218–240
19. Edmonds J, Karp RM (1972) Theoretical improvements in algorithmic efficiency for network flow problems. J ACM 19(2):248–264
20. Gao D, Tong Y, She J, Song T, Chen L, Xu K (2017) Top-k team recommendation and its variants in spatial crowdsourcing. Data Sci Eng 2(2):136–150

21. Han J, Wen J (2013) Mining frequent neighborhood patterns in a large labeled graph. In: CIKM, pp 259–268
22. Han J, Wen J, Pei J (2014) Within-network classification using radius-constrained neighborhood patterns. In: CIKM, pp 1539–1548
23. Han J, Zheng K, Sun A, Shang S, Wen J (2016) Discovering neighborhood pattern queries by sample answers in knowledge base. In: ICDE, pp 1014–1025
24. Huang Z, Kang N, Tang ZG, Wu X, Zhang Y, Zhu X (2018) How to match when all vertices arrive online. In: STOC, pp 17–29
25. Karp RM, Vazirani UV, Vazirani VV (1990) An optimal algorithm for on-line bipartite matching. In: STOC, pp 352–358
26. Kazemi L, Shahabi C (2012) GeoCrowd: enabling query answering with spatial crowdsourcing. In: GIS, pp 189–198
27. Kuhn HW (1955) The hungarian method for the assignment problem. Nav Res Logist Q 2(1–2):83–97
28. Liu A, Wang W, Shang S, Li Q, Zhang X (2018) Efficient task assignment in spatial crowdsourcing with worker and task privacy protection. GeoInformatica 22(2):335–362
29. Liu Y, Guo B, Du H, Yu Z, Zhang D, Chen C (2017) Poster: Foodnet: optimized on demand take-out food delivery using spatial crowdsourcing. In: MobiCom, pp 564–566
30. Mehta A (2013) Online matching and ad allocation. Found Trends Theoret Comput Sci 8(4):265–368
31. Shang S, Chen L, Wei Z, Jensen CS, Wen J, Kalnis P (2016) Collective travel planning in spatial networks. IEEE Trans Knowl Data Eng 28(5):1132–1146
32. Shang S, Chen L, Jensen CS, Wen J, Kalnis P (2017) Searching trajectories by regions of interest. IEEE Trans Knowl Data Eng 29(7):1549–1562
33. Shang S, Chen L, Wei Z, Jensen CS, Zheng K, Kalnis P (2017) Trajectory similarity join in spatial networks. PVLDB 10(11):1178–1189
34. Shang S, Chen L, Wei Z, Jensen CS, Zheng K, Kalnis P (2018) Parallel trajectory similarity joins in spatial networks. VLDB J 27(3):395–420
35. Shang S, Chen L, Zheng K, Jensen CS, Wei Z, Kalnis P (2018) Parallel trajectory-to-location join. IEEE Trans Knowl Data Eng 30(1):1–1
36. She J, Tong Y, Chen L, Cao CC (2016) Conflict-aware event-participant arrangement and its variant for online setting. IEEE Trans Knowl Data Eng 28(9):2281–2295
37. Song T, Tong Y, Wang L, She J, Yao B, Chen L, Xu K (2017) Trichromatic online matching in real-time spatial crowdsourcing. In: ICDE, pp 1009–1020
38. Tong Y, Zhou Z (2018) Dynamic task assignment in spatial crowdsourcing. SIGSPATIAL Special 10(2):18–25
39. Tong Y, She J, Ding B, Chen L, Wo T, Xu K (2016) Online minimum matching in real-time spatial data: experiments and analysis. PVLDB 9(12):1053–1064
40. Tong Y, She J, Ding B, Wang L, Chen L (2016) Online mobile micro-task allocation in spatial crowdsourcing. In: ICDE, pp 49–60
41. Tong Y, Chen L, Shahabi C (2017) Spatial crowdsourcing: challenges, techniques, and applications. PVLDB 10(12):1988–1991
42. Tong Y, Chen Y, Zhou Z, Chen L, Wang J, Yang Q, Ye J, Lv W (2017) The simpler the better: a unified approach to predicting original taxi demands based on large-scale online platforms. In: KDD, pp 1653–1662
43. Tong Y, Wang L, Zhou Z, Ding B, Chen L, Ye J, Xu K (2017) Flexible online task assignment in real-time spatial data. PVLDB 10(11):1334–1345
44. Tong Y, Chen L, Zhou Z, Jagadish HV, Shou L, Lv W (2018) SLADE: a smart large-scale task decomposer in crowdsourcing. IEEE Trans Knowl Data Eng 30(8):1588–1601
45. Tong Y, Wang L, Zhou Z, Chen L, Du B, Ye J (2018) Dynamic pricing in spatial crowdsourcing: a matching-based approach. In: SIGMOD, pp 773–788
46. Wang Y, Wong SC (2015) Two-sided online bipartite matching and vertex cover: beating the greedy algorithm. In: ICALP, pp 1070–1081
47. Zeng Y, Tong Y, Chen L, Zhou Z (2018) Latency-oriented task completion via spatial crowdsourcing. In: ICDE, pp 317–328
48. Zhang L, Hu T, Min Y, Wu G, Zhang J, Feng P, Gong P, Ye J (2017) A taxi order dispatch model based on combinatorial optimization. In: KDD, pp 2151–2159

Yiming Li is currently an undergraduate student in CSE at Beihang University. His current research interests include spatio-temporal data processing and game theory.



Jingzhi Fang is currently an undergraduate student in CSE at Beihang University. Her current research interests include spatio-temporal data processing and game theory.

Yuxiang Zeng is currently working toward the Ph.D. degree in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His major research interests are crowdsourcing and spatio-temporal data management.

Balz Maag received his M.Sc. degree in electrical engineering and information technology from ETH Zurich in 2014. He is currently a Ph.D. student in the Computer Engineering and Networks Laboratory (TIK) at ETH Zurich. His research interests include the development and optimization of algorithms for wireless sensor networks and crowd-sensing applications.

Yongxin Tong received the Ph.D. degree in Computer Science and Engineering from the Hong Kong University of Science and Technology, in 2014. He is currently a Distinguished Research Fellow in the State Key Laboratory of Software Development Environment of the School of Computer Science and Engineering at Beihang University. His research interests include crowdsourcing, spatio-temporal data processing and analysis, uncertain data mining and management, and social network analysis. He has published more than 50 academic papers in highly refereed journals and conference proceedings and has served on the program committees of more than two dozen international conferences and workshops. He is an associate editor of IEEE Transactions on Big Data (TBD). He is an Alibaba DAMO Academy Young Fellow (2018) and a member of the ACM, the IEEE and the CCF.

Lingyu Zhang joined Didi Chuxing in 2013 and is a research fellow in AI Labs. His research focuses on applying machine learning and data mining algorithms in ride sharing services. He designed and initiated several large intelligent systems in Didi, such as an optimized taxi dispatch system, destination predicting system. He has over 20 patents in intelligent transportation and his research has been published at KDD.

Affiliations

Yiming Li[1] · Jingzhi Fang[1] · Yuxiang Zeng[2] · Balz Maag[3] · Yongxin Tong[1] · Lingyu Zhang[4]

Yiming Li
ymli@buaa.edu.cn

Jingzhi Fang
15231133@buaa.edu.cn

Yuxiang Zeng
yzengal@cse.ust.hk

Balz Maag
balz.maag@tik.ee.ethz.ch

Lingyu Zhang
zhanglingyu@didichuxing.com

1 SKLSDE Lab, School of Computer Science and Engineering and IRI, Beihang University, Beijing, China

2 Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong SAR, China

3 Department of Information Technology and Electrical Engineering, ETH Zurich, Zurich, Switzerland

4 Didi Chuxing, Beijing, China