Online Mobile Micro-Task Allocation in Spatial Crowdsourcing

Yongxin Tong [†], Jieying She [‡], Bolin Ding [#], Libin Wang [†], Lei Chen [‡]
[†]SKLSDE Lab, School of Computer Science and Engineering and IRC, Beihang University, China
[‡]The Hong Kong University of Science and Technology, Hong Kong SAR, China
[#]Microsoft Research, Redmond, WA, USA
[†]{yxtong, lbwang}@buaa.edu.cn, [‡]{jshe, leichen}@cse.ust.hk, [#]bolind@microsoft.com

***Abstract*—With the rapid development of smartphones, spatial crowdsourcing platforms are getting popular. A foundational research of spatial crowdsourcing is to allocate micro-tasks to suitable crowd workers. Most existing studies focus on offline scenarios, where all the spatiotemporal information of micro-tasks and crowd workers is given. However, they are impractical since micro-tasks and crowd workers in real applications appear dynamically and their spatiotemporal information cannot be known in advance. In this paper, to address the shortcomings of existing offline approaches, we first identify a more practical micro-task allocation problem, called the *Global Online Micro-task Allocation in spatial crowdsourcing* ($GOMA$) problem. We first extend the state-of-art algorithm for the online maximum weighted bipartite matching problem to the GOMA problem as the baseline algorithm. Although the baseline algorithm provides theoretical guarantee for the worst case, its average performance in practice is not good enough since the worst case happens with a very low probability in real world. Thus, we consider the average performance of online algorithms, a.k.a online random order model. We propose a two-phase-based framework, based on which we present the TGOA algorithm with $\frac{1}{4}$-competitive ratio under the online random order model. To improve its efficiency, we further design the TGOA-Greedy algorithm following the framework, which runs faster than the TGOA algorithm but has lower competitive ratio of $\frac{1}{8}$. Finally, we verify the effectiveness and efficiency of the proposed methods through extensive experiments on real and synthetic datasets.**

I. Introduction

In recent years, crowdsourcing has attracted much attention of the industry and the research communities with the blossom of some successful crowdsourcing platforms, such as Amazon Mechanical Turks (AMT), oDesk, etc. Particularly, with the unprecedented development of smartphones and mobile Internet, crowdsourcing marketplaces are switching from traditional crowdsourcing markets to spatial crowdsourcing (a.k.a mobile crowdsourcing) markets, where crowd workers (workers for short in this paper) are paid to perform micro-tasks (tasks for short in this paper) using their mobile phones [1]. For example, on Gigwalk[1] and Gmission[2], consulting companies recruit crowd workers to check the prices of products in supermarkets, and Waze[3] uses crowd workers to collect real-time information of traffic or remaining parking lots.

Similar to the studies on traditional crowdsourcing markets, most existing research on spatial crowdsourcing focuses on the task allocation (a.k.a task assignment) problem [2], [3], [4], [5], [6], which aims to assign tasks to suitable workers such that the total number of assigned tasks or the total weighted value of the assigned pairs of tasks and workers is maximized. However, these existing works take the *offline scenario* assumption, where the spatiotemporal information of all the tasks and crowd workers is known before task allocation is conducted. Therefore, they are infeasible in real-time dynamic environments, where each task and crowd worker may appear anywhere at anytime and requires immediate responses from spatial crowdsourcing platforms. Imagine the following scenario. At noon on weekends, Tony usually wants to know how crowded his favorite restaurants around his home are so that he can decide which restaurant he should go to for lunch to avoid long queues. Thus, Tony posts a task on a spatial crowdsourcing platform, e.g. Gigwalk and Gmission, and asks the crowd workers to take photos of the waiting queues at the restaurants. Tony hopes to receive immediate response rather than waiting long. In fact, such tasks arrive dynamically and require real-time response, and so do crowd workers. Therefore, it raises a problem that most spatial crowdsourcing platforms encounter: *how to allocate the tasks to suitable workers in real-time dynamic environments (a.k.a online scenarios) and model such online scenarios*?

As introduced in [2], under offline scenarios, the task allocation problem in spatial crowdsourcing can be solved by being reduced to the problem of maximum weighted bipartite matching[7], where the tasks and workers correspond to the two disjoint sets of vertices in a bipartite graph, and there is an edge between two vertices from the two disjoint sets if the corresponding task locates in the restricted range of the corresponding worker, whose weight is the corresponding utility value of the pair of task and worker. Although the reduction can still be performed under online scenarios, the offline solutions become infeasible since the arrival orders of tasks and workers are unknown in dynamic environments. To further illustrate this motivation, we go through a toy example as follows.

Example 1: Suppose we have six micro-tasks $t_1 - t_6$ and three crowd workers $u_1 - u_3$ on a spatial crowdsourcing platform, whose initial locations are shown in a 2D space (X, Y) in Fig. 1. Each worker has a spatial restricted activity range, indicating that the worker can only conduct tasks that locate within the range, which is shown as a dotted circle in Fig. 1. Each user also has a capacity, which is the maximum number of tasks that can be assigned to him/her. In this

[1]http://www.gigwalk.com
[2]http://gmissionhkust.com/
[3]http://www.waze.com/

TABLE I: Utility between Micro Tasks and Crowd Workers

	t_1	t_2	t_3	t_4	t_5	t_6
w_1 (1)	**7**	1	2	3	2	1
w_2 (3)	5	1	1	2	1	2
w_3 (2)	6	2	**9**	**1**	1	1

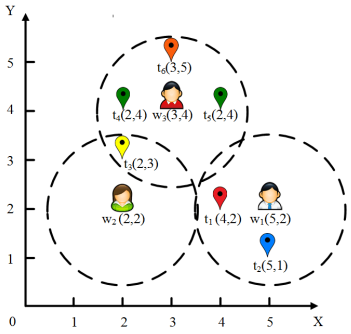


Fig. 1: Initial Locations of Micro-Tasks and Crowd Workers

example, $u_1 - u_3$ have capacities of 1, 3, and 2, respectively (in brackets). TABLE I presents the utility values between each pair of task and worker, which depends on the payoff of the task and the success ratio of the worker that can be inferred from how well this worker performs other tasks in history[2], [8]. Under the offline scenario, the total utility of the optimal task allocation, which is shown in bold font in TABLE I, is 17. However, in dynamic online scenarios, the offline solutions are not applicable since each task needs to be promptly assigned to a worker who has already arrived and vice versa, and we never know what the next arrival task or worker is in advance. For example, if the tasks and workers arrive following the "1st order" as shown in Table II, w_1 can be randomly assigned to t_1 when it arrives, and w_2 is assigned to t_3 after they arrive. Note that when w_3 arrives, it can only be assigned to two of $t_4 - t_6$, and thus the total utility is $7 + 1 + 1 + 1 = 10$. However, if the tasks and workers arrive following the "2nd order", $(t_1, w_1), (t_3, w_3)$ and (t_4, w_3) can be allocated respectively, resulting in total utility of 17, which is exactly the optimal allocation under the offline scenario. It indicates that the effectiveness of an online task allocation significantly depends on the arrival orders of tasks and workers.

As discussed above, we propose a new task allocation problem in dynamic environments, called the *Global Online Micro-task Allocation in spatial crowdsourcing* ($GOMA$) problem. As the example above indicates, the arrival order of tasks and workers significantly affects the performance of the algorithms. Notice that existing online studies generally focus on the performance of online algorithms on the worst-case arrival order, a.k.a online adversarial model. However, such algorithms may not perform well in practice because the worst-case order happens with a very low probability in real world. Therefore, we study the average performance of online algorithms in GOMA, a.k.a online random order model. In addition, a close branch of research is the online maximum weighted bipartite matching (OMWBM) problem [9], [10], [11], where the information of the left-hand vertices in a bipartite graph is known, and the right-hand vertices arrive dynamically. However, our GOMA problem differs from the OMWBM problem in that both the tasks and the workers

TABLE II: Arrival Time of Micro-Tasks and Crowd Workers

Arrival Time	8:00	8:01	8:02	8:07	8:08	8:09	8:09	8:15	8:18
1st Order	t_1	t_2	w_1	w_2	t_3	t_4	w_3	t_5	t_6
2nd Order	w_1	t_1	t_2	t_3	w_3	t_4	w_2	t_6	t_5

in GOMA are dynamic. Hence, the OMWBM problem is a special case of the GOMA problem. Therefore, we claim that the GOMA problem is *global online* or *two-sided online*. To the best of our knowledge, as discussed later, this is the first work that studies the GOMA problem particularly under the random order model and thus we should design efficient algorithms specifically for our problem. We make the following contributions.

- We identify a new online micro-task allocation problem, the Global Online Micro-task Allocation in spatial crowdsourcing ($GOMA$) problem, that has extensive spatial crowdsourcing applications. We extend the state-of-art algorithm for the OMWBM problem under the online adversarial model to the GOMA problem as the baseline algorithm and show that it has competitive ratio of $2eln(1+U_{max})$, where U_{max} is the maximum utility of a task-worker pair.
- We clarify that the baseline algorithm performs not well enough in practice since the worst case under the adversarial model happens with a very low probability in real world. We propose a two-phase-based framework that has competitive ratio of $\frac{1}{4}$ under the online random order model, which describes the average performance of online algorithms. Following the framework, we further propose a more efficient algorithm with slightly lower competitive ratio of $\frac{1}{8}$.
- We verify the effectiveness and efficiency of the proposed methods through extensive experiments on real and synthetic datasets.

The rest of the paper is organized as follows. In Section II, we formally formulate our problem. In Section III, we review related works. In Section IV, we present an online algorithm and prove its competitive ratio under the online adversarial model. Two effective algorithms with better theoretically guaranteed competitive ratios under the online random order model are proposed in Section V. Extensive experiments on both synthetic and real datasets are presented in Section VI. We finally conclude this paper in Section VII.

II. Problem Statement

We first formally define the *global online micro-task allocation in spatial crowdsourcing* problem, and then introduce two types of online models. Finally, an optimal solution which can be achieved under the offline scenario is proposed.

A. Problem Definitions

Definition 1 (Micro-Task): A micro-task ("task" for short), denoted by $t =< \boldsymbol{l}_t, a_t, d_t, p_t >$, at the location $\boldsymbol{l}_t$ in a 2D space is posted on the platform at time a_t and is either allocated to a crowd worker who arrives on the platform before the response deadline d_t or cannot be allocated thereafter. Furthermore, p_t is the payoff of the task t.

Definition 2 (Crowd Worker): A crowd worker ("worker" for short), denoted by $w =< \boldsymbol{l}_w, a_w, d_w, r_w, c_w, \delta_w >$, arrives at the platform with initial location $\boldsymbol{l}_w$ in a 2D space at time a_w and either performs several tasks which arrive at the platform before its response deadline d_w or does not conduct any task. r_w is the radius of the restricted circular range of w, whose center is $\boldsymbol{l}_w$. In addition, capacity c_w is the maximum number of tasks that w intends to finish, and $\delta_w \in (0, 1]$ is the success ratio of w according to the historical records of w completing tasks on the platform.

We then define the utility value that a worker is allocated to a task as follows.

Definition 3 (Utility Value): The utility that a worker w performs a task t is measured by $U(t, w) = p_t \times \delta_w$.

Note that the micro-tasks in this paper are usually simple and trivial, e.g. taking a photo by mobile phones, checking prices in the supermarket, etc. On one hand, we assume that the crowd workers do not need any specific skill to conduct a micro-task, so the success ratio of a worker represents his/her reliability. On the other hand, we can observe that the rewards of different micro-tasks have no much difference in real applications. Since most spatial crowdsourcing platforms make a profit in a fixed proportion of the payoff of the allocated tasks, the aforementioned utility function not only maximizes the profits of spatial crowdsourcing platforms but also guarantees the reliability of the assigned workers. Finally, we define our problem as follows.

Definition 4 (GOMA Problem): Given a set of micro-tasks T, a set of crowd workers W, and a utility function $U(.,.)$ on a spatial crowdsourcing platform, which has no task or worker initially and allows that each task and worker can arrive one by one at any time, the GOMA problem is to find an allocation M among the tasks and the workers to maximize the total utility $MaxSum(M) = \sum_{t \in T, w \in W} U(t, w)$ such that the following constraints are satisfied:

- Deadline constraint: after a task t arrives, it is either assigned to a worker w who arrives at the platform before the response deadline of t, d_t, or is not allocated thereafter, and vice versa.
- Invariable constraint: once a task t is allocated to a worker w, the allocation of (t, w) cannot be changed.
- Capacity constraint: the number of tasks assigned to a worker w cannot exceed w's capacity c_w.
- Range constraint: any task assigned to a worker w must locate in the restricted range of w.

B. Online Input Models

The performance of online algorithms is usually compared with the optimal allocation of the offline scenario and heavily depends on the arriving orders of tasks and workers. In the following, we first provide the corresponding offline version of the GOMA problem and then formally introduce the online arriving models of the GOMA problem.

The offline version of the GOMA problem is identical to the online GOMA problem except that the first two constraints are excluded. In other words, the spatiotemporal information of all the spatial tasks and the crowd workers is known before the task allocation is conducted. Therefore, we can obtain the optimal result under the offline GOMA problem.

Moreover, different from traditional approximation algorithms for which approximation ratios are utilized to measure the approximation quality, for online algorithms, *competitive ratios (CR)* are used to evaluate their performance. In particular, the competitive ratio measures how good an online algorithm is compared with the optimal result of the offline model where all the information is provided. Based on different assumptions on the arrival order of the tasks and workers, we use two types of online models, called the adversarial model and the random order model, which focus on the worst-case arrival order and the stochastic arrival order of all the tasks and workers, respectively. The corresponding competitive ratios of the two types of online models are defined as follows.

Definition 5 (CR in the Adversarial Model): The competitive ratio in the the adversarial model of a specific online algorithm for the GOMA problem is the following minimum ratio between the result of the online algorithm and the optimal result over all possible arrival orders of the tasks and the workers,

$$CR_A = min_{\forall G(T,W,U) \text{ and } \forall v \in V} \frac{MaxSum(M)}{MaxSum(OPT)} \quad (1)$$

where $G(T, W, U)$ is an arbitrary input of tasks, workers and their utilities, V is the set of all possible input orders, v is one order in V, $MaxSum(M)$ is the total utility produced by the online algorithm, and $MaxSum(OPT)$ is the optimal total utility of the offline scenario.

Definition 6 (CR in the Random Order Model): The competitive ratio in the the random order model of a specific online algorithm for the GOMA problem is the following ratio,

$$CR_{RO} = min_{\forall G(T,W,U)} \frac{\mathbb{E}[MaxSum(M)]}{MaxSum(OPT)} \quad (2)$$

where $G(T, W, U)$ is an arbitrary input of tasks, workers and their utilities, $\frac{\mathbb{E}[MaxSum(M)]}{MaxSum(OPT)}$ is the expectation of the ratio of the total utility produced by the online algorithm and the optimal total utility of the offline scenario over all possible arrival orders.

The optimal solution of the offline problem is introduced in the next subsection.

C. Offline Algorithm

In this subsection, we introduce an optimal algorithm for the offline GOMA problem. Since the offline GOMA problem can be reduced to the maximum weighted bipartite matching (MWBM) problem as explained shortly, the main idea of the offline algorithm is to first transform an offline GOMA problem instance to an MWBM instance and then adopts classical flow algorithms, e.g. Hungarian algorithm [12], to calculate the optimal result.

Specifically, given an instance of the offline GOMA problem, which includes a set of spatial tasks T, a set of crowd workers W, and a utility function $U(.,.)$, we construct a flow network $G = (V, A)$ as follows. $V = W \cup T \cup \{s, d\}$, where s is a source node and d is a sink node. For each pair $t \in T, w \in W$, there is a weighted directed arc $a_{w,t} \in A$ with $a_{w,t}.weight = U(w, t)$ from w to t if the time interval $[a_w, d_w]$ of w overlaps with the time interval $[a_t, d_t]$ of t, namely

$(a_w \leq d_t) \wedge (a_t \leq d_w)$. In addition, for every $w \in W$, there is a directed arc $a_{s,w} \in A$ from s to w with $a_{s,w}.weight = 0$. Similarly, for every $t \in T$, there is also a directed arc $a_{t,d} \in A$ from t to d with $a_{t,d}.weight = 0$. The total amount of flow in the network is $\min(|T|, |W|)$. So far, the flow network instance has been constructed. Then, we use existing flow algorithms to obtain the optimal result of the MWBM instance. Then to obtain the allocation result of the offline GOMA problem, we simply assign a crowd worker w to a task t if there is a non-zero flow between them in the MWBM result.

III. Related Work

In this section, we review related works from two categories, spatial crowdsourcing and online maximum bipartite matching.

A. Spatial Crowdsourcing

In recent years, a wide spectrum of fundamental data-driven operations for general crowdsourcing platforms are well studied, such as filtering[13], entity resolution[14], data cleaning[15], topic discovery[16], and a couple of crowd-powered database systems have been successfully developed, such as CrowdDB[17], Deco[18] and Qurk[19]. Particularly, task assignment is one of the most important issues. [20], [21] study the task assignment problem in offline scenarios by learning the quality of crowd workers. [22] propose online task assignment in crowdsourcing, where only tasks are dynamic. In particular, all these studies focus on data management for traditional crowdsourcing rather than spatial crowdsourcing.

With the development of smartphones, more real applications of crowdsourcing are switching to spatial crowdsourcing [1]. [2] is the first on task allocation on spatial crowdsourcing platforms, whose goal is to maximize the number of assigned tasks. [8] generalizes the model of [2] by adding a score function to each pair of task and worker and aims to maximize the total score of the assignment. [3] integrates the reliability of crowd workers into the model of [2], and [6] studies the problem of maximizing the reliability and the spatiotemporal diversity of workers concurrently. [5], [23] focus on protecting the location privacy of crowd workers in the assignment process. The route planning problem for a crowd worker is also proposed, which aims at maximizing the number of completed tasks [3], and its corresponding online version is also studied [24], whose route optimization problem is quite different from ours. In addition, a general spatial crowdsourcing platform, gMission, is developed [25]. Although the aforementioned works study the task allocation problem on spatial crowdsourcing, they mostly focus on *offline scenarios*, where the spatiotemporal information of all the tasks and workers is known before the task allocation is conducted. In other words, the above studies do not address the task allocation problem for tasks and workers that arrive dynamically, and are thus impractical for dynamic environments in real spatial crowdsourcing applications.

Some recent works on spatial matching [26], [27], [28] focus on matching 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], stable marriage[26] and min-max[28]. Note that these studies also take the offline scenario assumption.

In particular, a closely related research [29], the online spatial task assignment problem to maximize the number of assigned tasks, has been proposed recently. The biggest difference between [29] and our work lies in the problem definitions. First, in our online model, all the tasks and workers can dynamically appear anywhere at anytime, but in [29] only tasks are dynamic. Also, our goal is to maximize the total utility value of the allocated pairs of tasks and workers, but [29] is to maximize the number of assigned tasks. In addition, the solutions of [29] fail to provide any competitive ratio guarantee, but our proposed approaches provide competitive analysis and theoretical guarantees under both the adversarial model and the random order model, respectively.

B. Online Maximum Bipartite Matching

The OMWBM problem is the online version of the maximum weighted bipartite matching problem [7]. The input of the OMWBM problem is a weighted bipartite graph $G = (L, R, E, U)$, whose left-hand vertices L are known beforehand, but the right-hand vertices R are unknown and arrive one by one. Once a right-hand vertex $r \in R$ arrives, the edges $(l, r) \in E$ incident to r and their corresponding weights $U(l, r) \in U$ are revealed, and r must either match a left-hand vertex l or remain unmatched thereafter. The goal of the OMWBM problem is to maximize the total sum of the weights of the matched edges [30]. In particular, different assumptions on the arrival orders of R lead to different online models, which can be simply divided to two categories: the online adversarial model and the online random order model.

1) Adversarial Model: In this model, we assume that the information of R, E and U and the arrival order of R is unknown and can be arbitrarily bad. In other words, the adversarial model focuses on the worst-case arrival order of R. Karp et al. first defines the problem of online maximum UNWEIGHTED bipartite matching under the adversarial model and proposes the Ranking algorithm with competitive ratio of $1 - \frac{1}{e}$. Recently, for the OMWBM problem under the adversarial model, [11] proposes the state-of-the-art randomized algorithm. Note that the aforementioned works only focus on the case where only one single side of vertices arrive online. However, in our GOMA problem, both sides of vertices arrive online. Furthermore, a closely related research, the two-sided online maximum unweighted bipartite matching problem [31] is studied. However, there are two major differences between our GOMA problem and [31]. First, our GOMA problem considers the weighted cases, but [31] can only address the unweighted case. Second, as discussed later, our main contribution is to devise effective and efficient online algorithms under the random order model, which is not considered in [31].

2) Random Order Model: The major characteristics of the random order model is the assumption that the arrival order of vertices follows a uniform random permutation, hence focuses on the average or expected performance of online algorithms. [32] first proposes the online maximum unweighted bipartite matching problem under the random order model and proves that the competitive ratio of the simple greedy algorithm can be enhanced from $\frac{1}{2}$ under the adversarial model to $1 - \frac{1}{e}$. Then, [33] and [34] propose two randomized algorithms with competitive ratios of $\frac{1}{8}$ and $\frac{1}{e}$ for the OMWBM problem under the random order model, respectively. However, all these

existing studies only address the case where one single side of vertices arrive online. Therefore, our GOMA problem, where both the tasks and workers (two sides of vertices) arrive online, cannot be solved directly by any existing solution.

IV. Baseline Algorithm

In this section, we extend the Greedy-RT algorithm [11] as a baseline algorithm. The Greedy-RT algorithm [11] has the state-of-art competitive ratio of $\frac{1}{2e\ln(1+U_{max})}$ under the online adversarial model for the OMWBM problem, where only one single side of vertices in the weighted bipartite graph arrive online and U_{max} is the maximum weight. The main idea of Greedy-RT [11] is to first randomly choose a threshold on the weights of edges, and then for each new arrival vertex, an arbitrary edge incident to it whose weight is no less than the chosen threshold is added to the match result if such edge exists. We extend the Greedy-RT algorithm to the GOMA problem, called Extended-Greedy-RT, whose framework is similar to that of Greedy-RT, but both sides of vertices (i.e. tasks and workers) of the weighted bipartite graph arrive dynamically, and the deadline constraint of our GOMA problem is considered. Note that in the Extended-Greedy-RT algorithm, for each worker with capacity c_w, we treat it as c_w duplicated workers who arrive at the same time. In other words, when a worker with capacity c_w arrives, we take it as c_w duplicated workers, each of whom has capacity 1, and process each of them in order.

The procedure of Extended-Greedy-RT is illustrated in Algorithm 1. In lines 1-2, Extended-Greedy-RT randomly chooses a threshold (e^k) on the weights of edges according to the estimated maximum weight U_{max}, which can be learned from historical records on the spatial crowdsourcing platform. When a new vertex arrives, which may be a task or a worker, Extended-Greedy-RT then adds an edge among the ones whose weights are no less than the threshold and that satisfy all the constraints to the match result in lines 3-9. Note that when a worker with capacity c_w arrives, Extended-Greedy-RT takes it as c_w duplicates of w that arrive at the same time and processes them one by one. We then explain Extended-Greedy-RT with a running example and further show that Extended-Greedy-RT achieves the same competitive ratio as Greedy-RT does following the analysis similar to [11].

Example 2 (Extended Greedy-RT): Back to our running example in Example 1. The Extended Greedy-RT algorithm sets $\theta = \lceil \ln(9+1) \rceil = 3$, so $k \in \{0, \cdots, 3\}$. If k is chosen as 0, the threshold is $e^0 = 1$. According to the 1st arrival order in Table II, when w_1 arrives, the candidate set $Cand = \{t_1, t_2\}$, and the algorithm assign t_1 to w_1. Similarly, t_3 and t_4 are allocated to w_2 and w_3, respectively. Thus, when $k = 0$, the total utility score is 10. Since the Extended Greedy-RT algorithm is a randomized algorithm on choosing k, the expectation of the total utility score for all possible k is $\frac{10+16+16+0}{4} = 10.5$.

Lemma 1: The competitive ratio of Extended-Greedy-RT is $\frac{1}{2e\lceil \ln(U_{max}+1) \rceil}$.

Proof: According to the offline algorithm in Section II-C, we can obtain the corresponding weighted bipartite graph of the GOMA input, denoted as G. Let $G_{[e^i, e^{i+1})}$ be a subgraph of G, which only contains the edges whose utilities lie in the interval $[e^i, e^{i+1})$, and $OPT_{[e^i, e^{i+1})}$ and $M_{[e^i, e^{i+1})}$ be the optimal match and the one returned by Extended-Greedy-RT on $G_{[e^i, e^{i+1})}$, respectively. For any edge in $OPT_{[e^i, e^{i+1})}$, at least one of the two vertices of this edge must be matched in $M_{[e^i, e^{i+1})}$. Therefore, $|M_{[e^i, e^{i+1})}| \geq \frac{1}{2}|OPT_{[e^i, e^{i+1})}|$. Then, we have

$$\begin{aligned}
\mathbb{E}[MaxSum(M)] &= \frac{1}{\theta}\sum_{i=0}^{\theta} MaxSum(M_{[e^i, U_{max})}) \\
&\geq \frac{1}{\theta}\sum_{i=0}^{\theta} e^i |M_{[e^i, e^{i+1})}| \\
&\geq \frac{1}{\theta}\sum_{i=0}^{\theta} e^i \frac{|OPT_{[e^i, e^{i+1})}|}{2} \\
&\geq \frac{1}{2e\theta}\sum_{i=0}^{\theta} MaxSum(OPT_{[e^i, e^{i+1})}) \\
&\geq \frac{1}{2e\theta} MaxSum(OPT)
\end{aligned}$$

Thus, the competitive ratio of Extended-Greedy-RT is

$$CR = \frac{\mathbb{E}[MaxSum(M)]}{MaxSum(OPT)} = \frac{1}{2e\theta} = \frac{1}{2e\lceil \ln(U_{max}+1) \rceil}$$

since $\theta = \lceil \ln(U_{max}+1) \rceil$, where U_{max} is the maximum utility of the GOMA input. ■

Algorithm 1: Extended Greedy-RT

input : $T, W, U(.,.)$
output: A feasible allocation M
1 $\theta \leftarrow \lceil \ln(U_{max}+1) \rceil$;
2 $k \leftarrow$ randomly choosing an integer from $\{1, \cdots, \theta\}$ with probability $\frac{1}{\theta}$;
3 **foreach** *new arrival task or worker* v **do**
4 $Cand \leftarrow \{\forall u | u$ is an unmatched neighbor of v such that $U(u,v) \geq e^k$ and it satisfies all constraints$\}$;
5 **if** $Cand = \emptyset$ **then**
6 continue;
7 **else**
8 $u^\star \leftarrow$ an arbitrary item is chosen from $Cand$;
9 $M \leftarrow M \cup (u^\star, v)$;
10 **return** M

Complexity Analysis. For the each new arrival task or worker, the time and space complexity of the Extended-Greedy-RT algorithm are both $O(\max(|T|, |W|))$, respectively.

Although Extended-Greedy-RT provides a theoretical guarantee for the worst case, the worst case only appears with probability $\frac{1}{n!}$ in online scenarios if the total number of tasks and workers is n. That is because the worst case is only one among $n!$ different arrival orders of n tasks and workers. Therefore, if an online algorithm focuses on its performance on the worst case, it does not make much sense in real applications, which pay more attention to the average performance of the algorithms. In order to address this issue, we adopt the competitive ratio on the random order model

Algorithm 2: TGOA

input : $T, W, U(.,.)$
output: A feasible allocation M

1 $m \leftarrow |T|, n \leftarrow \sum_{i=1}^{|W|} c_i, k \leftarrow \lfloor \frac{m+n}{2} \rfloor$;
2 Multiset $W^{\Delta} \leftarrow \emptyset, T^{\Delta} \leftarrow \emptyset$;
3 **for** *each new arrival task or worker* v **do**
4 **if** $|T^{\Delta}| + |W^{\Delta}| < k$ **then**
5 **if** $v \in W$ **then**
6 $t \leftarrow$ the task with the highest utility that is unmatched and satisfies all constraints;
7 **if** t *exists* **then**
8 $M \leftarrow M \cup (t, v)$;
9 **else**
10 $w \leftarrow$ the worker with the highest utility that is unmatched and satisfies all constraints;
11 **if** w *exists* **then**
12 $M \leftarrow M \cup (v, w)$;
13 **else**
14 $M_v \leftarrow Hungarian(T^{\Delta} \cup W^{\Delta} \cup \{v\})$;
15 **if** v *is matched in* M_v **then**
16 **if** $v \in W$ **then**
17 $t \leftarrow$ the task assigned to v in M_v;
18 **if** t *is unmatched in* M *and satisfies all constraints* **then**
19 $M \leftarrow M \cup (t, v)$;
20 **else**
21 $w \leftarrow$ the worker assigned v in M_v;
22 **if** w *is unmatched in* M *and satisfies all constraints* **then**
23 $M \leftarrow M \cup (v, w)$;
24 **if** $v \in W$ **then**
25 $W^{\Delta} \leftarrow W^{\Delta} \cup v$;
26 **else**
27 $T^{\Delta} \leftarrow T^{\Delta} \cup v$;
28 **return** M

The Hungarian method is a combinatorial optimization algorithm that solves the assignment problem in polynomial time and which anticipated later primal-dual methods

as the new evaluation standard, which measures the average performance of online algorithms, and propose a more effective algorithm framework in the next section.

V. A Two-Phase-Based Framework

As discussed in Section IV, the global adversarial model only focuses on the worst case, which actually happens with a very low probability in real world. Therefore, we propose a more effective algorithm framework, a two-phase-based framework, which can obtain better theoretical guarantees – constant competitive ratios under the global random order model in this section. Based on the framework, we present a *Two-phase-based Global Online Allocation (TGOA)* algorithm with competitive ratio of $\frac{1}{4}$. To improve its time efficiency, we further present the TGOA-Greedy algorithm following the framework, which is faster than the TGOA algorithm but leads to a slightly lower competitive ratio of $\frac{1}{8}$.

A. TGOA Algorithm

Inspired by the approaches solving the classic secretary problem [35], we present the TGOA algorithm. The main idea of TGOA is to first divide all the vertices, each of which can either be a task or a worker, into two equal groups according to their arrival orders and adopt different strategies on them. That is, the first half of vertices arrive first while the second half of vertices arrive afterwards. Note that the total number of tasks and workers on the platform in a time interval, e.g. 24 hours, can be estimated according to the historical records of the platform. Then for the first half of tasks and workers, TGOA conducts a greedy strategy to assign each new arrival task (worker) to the corresponding worker (task) with the highest utility and satisfying all the constraints. For the second half of tasks and workers, when a new task or worker arrives, TGOA tries to perform a global optimal match w.r.t. the second-half vertices that have arrived if possible in order to reduce the possibility of being trapped to the local optimal of the greedy strategy on the first half of tasks and workers. Similar to the Extended Greedy-RT algorithm, for each worker with capacity c_w, we treat it as c_w duplicated workers who arrive at the same time and process them one by one.

More specifically, let $k \leftarrow \lfloor \frac{m+n}{2} \rfloor$, where m is the number of tasks and $n \leftarrow \sum_{i=1}^{|W|} c_i$. Then the first k items (tasks or workers) arrive belong to the first half of vertices, and the others belong to the second half of vertices. Then for the first half of tasks or workers, when a new task (or worker) arrives, we simply assign the corresponding worker (or task) with the largest utility value and satisfying all the constraints to the new arrival task (or worker). Then starting from the $(k+1)$-th new arrival task or worker, we adopt a more optimal strategy. That is, among the second-half vertices that have arrived, including the new arrival one v, we hypothetically find a global optimal match M_v using the Hungarian algorithm. If v is matched in the global optimal match M_v, we assign v to the corresponding vertex that is matched to v in M_v if such vertex has not been assigned previously and satisfies all the constraints.

Algorithm 2 illustrates the procedure of TGOA. In line 1, we calculate the number of the first half of vertices k, which can be estimated according to historical records. In line 2, we initialize two sets, which are used to store the vertices that have arrived. Note that W^{Δ} is a multiset. Then in lines 3-27, we iteratively process each new arrival task or worker. Particularly, we adopt a greedy strategy on the first half of new arrival vertices in lines 4-12 and adopt a more optimal strategy on the second half of new arrival vertices in lines 14-23. For each of the first half of vertices, we either assign a task with the highest utility value that satisfies all the constraints to it if it is a worker in lines 6-8 or assign a worker with the largest utility that satisfies all the constraints to it if it is a task in lines 10-12. Then for each of the second half of vertices, we first run the Hungarian algorithm on the vertices that have arrived to obtain a current global optimal match M_v in line 14. If v is matched in M_v, we either assign to v the corresponding task that is matched to v in M_v if v is a worker in lines 17-19 or assign to v the corresponding worker that is matched to v in M_v is v is a task if such allocation is feasible in lines 21-23. Finally, the sets T^{Δ} and W^{Δ} are updated accordingly in lines 24-27.

Example 3 (TGOA Algorithm): Back to our running example in Example 1. The TGOA algorithm first estimates $m = 6, n = 1 + 3 + 2 = 6, k = \frac{6+6}{2} = 6$. Based on the 1st arrival order in Table II, for the first half of vertices, namely the first 6 arrival tasks and duplicates of workers, the algorithm allocates t_1 to w_1. For the second half of vertices, t_3 and t_4 are assigned to w_3 since the Hungarian algorithm only match the second half of vertices to each other. Therefore, the total utility score is 7+9+1=17, which is better than the expectation of the total utility score, 10, of the Extended Greedy-RT algorithm.

Competitive Ratio. We next study the competitive ratio of TGOA, which is inspired by [35] and extended to our problem. For convenience, we propose a variant of TGOA, TGOA-Filter, which directly ignores the first half of arrival tasks and workers and only performs lines 14-27 of TGOA for the second half of arrival tasks and workers. Therefore, the total utility of TGOA-Filter must be no greater than that of TGOA. In other words, if the competitive ratio of TGOA-Filter is α, the competitive ratio of TGOA must be at least α. Note that the following analysis assumes that TGOA-Filter filters out the first $\lfloor \frac{m+n}{\epsilon} \rfloor$ arrival items. As discussed later, the optimal value of ϵ is 2.

The following analysis about TGOA-Filter relies on the fact that, for each new arrival item v on the platform, the possibility of $v \in T$ (v being a task) is equal to that of $v \in W$ (v being a worker). For the first $\lfloor \frac{m+n}{\epsilon} \rfloor$ arrival items that are filtered out by TGOA-Filter, it can be regarded as that $\lfloor \frac{m+n}{2\epsilon} \rfloor$ tasks and $\lfloor \frac{m+n}{2\epsilon} \rfloor$ workers are filtered out, respectively. When TGOA-Filter processes the i-th arrival item v ($i \in [\lfloor \frac{m+n}{\epsilon} \rfloor + 1, m+n]$), $|T^{\Delta}|$ tasks and $|W^{\Delta}|$ workers have arrived on the platform. Hence, we have the following lemmas.

Lemma 2: $\mathbb{E}[MaxSum(M_v)] \geq \frac{|T^{\Delta}||W^{\Delta}|}{mn} MaxSum(OPT)$

Proof: Let $OPT_{(|T^{\Delta}|,n)}$ be the optimal matching generated by the Hungarian algorithm for the offline weighted bipartite graph, which includes $|T^{\Delta}|$ tasks and n workers, so $MaxSum(OPT_{(|T^{\Delta}|,n)})$ is the total utility of $OPT_{(|T^{\Delta}|,n)}$. Due to the randomness of the random order model, W^{Δ} can be considered as the set of $|W^{\Delta}|$ workers who are uniformly chosen from W. Therefore, we have

$$\mathbb{E}[MaxSum(M_v)] \geq \frac{|W^{\Delta}|}{n} MaxSum(OPT_{(|T^{\Delta}|,n)})$$

Similarly, T^{Δ} can also be considered as the set of $|T^{\Delta}|$ tasks which are uniformly chosen from T, then we have

$$MaxSum(OPT_{(|T^{\Delta}|,n)}) \geq \frac{|T^{\Delta}|}{m} MaxSum(OPT)$$

Therefore,

$$\mathbb{E}[MaxSum(M_v)] \geq \frac{|T^{\Delta}|}{m} \frac{|W^{\Delta}|}{n} MaxSum(OPT)$$

■

According Lemma 2, we can achieve the following lemma.

Lemma 3:

$$\begin{cases} \mathbb{E}[U(v,w)] \geq \frac{|W^{\Delta}|}{mn} MaxSum(OPT) & v \in T \\ \mathbb{E}[U(t,v)] \geq \frac{|T^{\Delta}|}{mn} MaxSum(OPT) & v \in W \end{cases}$$

Proof: If $v \in T$, when v arrives at the platform and is added into T^{Δ}, which is the set of tasks that have already arrived, v can be considered as a task that is uniformly chosen from T^{Δ}. Thus, the expectation of the utility of edge $(v,w) \in M_v$ is

$$\mathbb{E}[U(v,w)] = \frac{1}{|T^{\Delta}|} MaxSum(M_v)$$

According to Lemma 2, we have

$$\begin{aligned} \mathbb{E}[U(v,w)] &= \frac{1}{|T^{\Delta}|} MaxSum(M_v) \\ &\geq \frac{|W^{\Delta}|}{mn} MaxSum(OPT) \end{aligned}$$

Similarly, we can also obtain the expectation of the utility of edge $(t,v) \in M_v$ if $v \in W$,

$$\begin{aligned} \mathbb{E}[U(t,v)] &= \frac{1}{|W^{\Delta}|} MaxSum(M_v) \\ &\geq \frac{|T^{\Delta}|}{mn} MaxSum(OPT) \end{aligned}$$

■

The aforementioned two lemmas provide a bound on the expectation of the utility of edge $(v,w) \in M_v$ (or $(t,v) \in M_v$) under the random order model, which is added into the final matching if and only if w (or t) is not matched before the i-th item v arrives. We further analyze the probability that the task or worker matched to v is unmatched when the $(\lfloor \frac{m+n}{\epsilon} \rfloor + 1)$-th to $(i-1)$-th items arrive.

Lemma 4:

$$\begin{cases} \mathbb{P}\{w \text{ is unmatched in steps } [\lfloor \frac{m+n}{\epsilon} \rfloor + 1, |T^{\Delta}| - 1]\} = \frac{\lfloor \frac{m+n}{2\epsilon} \rfloor}{|T^{\Delta}|-1} & v \in T \\ \mathbb{P}\{t \text{ is unmatched in steps } [\lfloor \frac{m+n}{\epsilon} \rfloor + 1, |W^{\Delta}| - 1]\} = \frac{\lfloor \frac{m+n}{2\epsilon} \rfloor}{|W^{\Delta}|-1} & v \in W \end{cases}$$

Proof: If $v \in T$, we construct $T' = \{t_{\lfloor \frac{m+n}{2\epsilon} \rfloor + 1}, \cdots, t_{|T^{\Delta}|-1}\}$, which is the set of the tasks that arrive after the $(\lfloor \frac{m+n}{2\epsilon} \rfloor)$-th task but before v. Let $t_j \in T'$ be the j-th task arriving at the platform, then the worker w is assigned to t_j with probability at most $\frac{1}{j}$ since at least the first $j-1$ arrival tasks cannot be matched to w. In other words, for the j-th arrival task, u is unmatched with probability $\frac{j-1}{j}$. Therefore, we have

$$\begin{aligned} &\mathbb{P}\{w \text{ is unmatched in steps } [\lfloor \frac{m+n}{\epsilon} \rfloor + 1, |T^{\Delta}| - 1]\} \\ &= \prod_{j=\lfloor \frac{m+n}{\epsilon} \rfloor + 1}^{|T^{\Delta}|-1} \frac{j-1}{j} = \frac{\lfloor \frac{m+n}{2\epsilon} \rfloor}{|T^{\Delta}| - 1} \end{aligned}$$

Similarly, if $v \in W$, we have

$$\mathbb{P}\{t \text{ is unmatched in steps } [\lfloor \frac{m+n}{\epsilon} \rfloor + 1, |W^{\Delta}| - 1]\} = \frac{\lfloor \frac{m+n}{2\epsilon} \rfloor}{|W^{\Delta}| - 1}$$

■

According Lemmas 3 and 4, we have the following lemma.

Lemma 5:

$$\begin{cases} \mathbb{E}[U(v,w) \in M] \geq \frac{\lfloor \frac{m+n}{2\epsilon} \rfloor}{|T^{\Delta}|-1} \frac{|W^{\Delta}|}{mn} MaxSum(OPT) & v \in T \\ \mathbb{E}[U(t,v) \in M] \geq \frac{\lfloor \frac{m+n}{2\epsilon} \rfloor}{|W^{\Delta}|-1} \frac{|T^{\Delta}|}{mn} MaxSum(OPT) & v \in W \end{cases}$$

Proof: If $v \in T$, the edge $(v, w) \in M_v$ can be added to the final match, M, of TGOA-Filter if and only if w is not matched when the the $(\lfloor \frac{m+n}{\epsilon} \rfloor + 1)$-th to $(i-1)$-th items arrive. According to Lemmas 3 and 4, we can obtain the expectation of the utility $U(v, w) \in M$

$$\mathbb{E}[U(v,w) \in M] \geq \frac{\lfloor \frac{m+n}{2\epsilon} \rfloor}{|T^{\Delta}| - 1} \times \frac{|W^{\Delta}|}{mn} MaxSum(OPT)$$

Similarly, we can also obtain the expectation of the utility $U(t, v) \in M$ if $v \in T$,

$$\mathbb{E}[U(v,w) \in M] \geq \frac{\lfloor \frac{m+n}{2\epsilon} \rfloor}{|W^{\Delta}| - 1} \times \frac{|T^{\Delta}|}{mn} MaxSum(OPT)$$

■

Theorem 1: The competitive ratio of the TGOA algorithm under random order model is $\frac{1}{4}$.

Proof: Let U_v denote the utility contributed by v in the final match M. According to Lemma 5, it is easy to see that $\mathbb{E}[U_v] = \frac{1}{2}\mathbb{E}[U(v, w) \in M]$ if $v \in T$, vice versa. In addition, in the online random order model, an arbitrary order must have a corresponding symmetrical order. In other words, in an arbitrary order, if a new arrival item v is a task, in the corresponding symmetrical order, there must be an item with the same arrival order that is a worker. Therefore, we have

$$\begin{aligned}
&\mathbb{E}[MaxSum(M)] = \frac{1}{2}\mathbb{E}[\sum_{v=1}^{m+n} U_v] \\
&\geq \frac{1}{2} \times \frac{1}{2} \sum_{v=\lceil \frac{m+n}{\epsilon} \rceil}^{m+n} (\frac{\lfloor \frac{m+n}{2\epsilon} \rfloor}{|W^{\Delta}| - 1} \frac{|T^{\Delta}| MaxSum(OPT)}{mn} + \\
&\frac{\lfloor \frac{m+n}{2\epsilon} \rfloor}{|T^{\Delta}| - 1} \frac{|W^{\Delta}| MaxSum(OPT)}{mn}) \\
&\geq \frac{1}{2} \sum_{i=\lceil \frac{m+n}{\epsilon} \rceil}^{m+n} \frac{\frac{m+n}{2\epsilon}}{mn} MaxSum(OPT) \\
&= \frac{\lfloor \frac{m+n}{2\epsilon} \rfloor}{2mn} (m + n - \lceil \frac{m+n}{\epsilon} \rceil + 1) MaxSum(OPT) \\
&\geq \frac{1}{\epsilon}(1 - \frac{1}{\epsilon}) MaxSum(OPT)
\end{aligned}$$

In order to maximize the $\frac{1}{\epsilon}(1 - \frac{1}{\epsilon})$, we can easily know that

$$\frac{1}{\epsilon}(1 - \frac{1}{\epsilon}) \leq (\frac{\frac{1}{\epsilon} + 1 - \frac{1}{\epsilon}}{2})^2 = \frac{1}{4}$$

Therefore, $\mathbb{E}[MaxSum(M)] \geq \frac{1}{4} MaxSum(OPT)$. ■

Complexity Analysis. For each of the first half of new arrival tasks or workers, the time and space complexity of TGOA are $O(\max(|T|, |W|))$ and $O(1)$, respectively. For each of the second half of new arrival tasks or workers, the time and space complexity of TGOA are $O(\max(|T^{\Delta}|^3, |W^{\Delta}|^3))$ and $O(\max(|T^{\Delta}|^2, |W^{\Delta}|^2))$, respectively.

B. TGOA-Greedy Algorithm

Notice that though the TGOA algorithm adopts a more optimal strategy on the second half of vertices to ensure higher quality of the allocation results, it is not efficient enough since it takes cubic time complexity to find the current global optimal match for each of the second-half vertices. Therefore, in this subsection, we improve the efficiency of TGOA by replacing the optimal Hungarian algorithm with a greedy strategy, which leads to a slightly lower competitive ratio.

Algorithm 3: Greedy-Match

input : A baipartite graph
output: An offline allocation M_v^{Greedy}

1 $M_v^{Greedy} \leftarrow \emptyset$;
2 **while** *True* **do**
3 $(t, w) \leftarrow$ the task-worker pair with the highest utility that is unmatched and satisfies all constraints;
4 **if** (t, w) *exists* **then**
5 $M_v^{Greedy} \leftarrow M_v^{Greedy} \cup (t, w)$;
6 **else**
7 break;
8 **return** M_v^{Greedy}

More specifically, we replace the Hungarian algorithm with the following greedy algorithm in line 14 of Algorithm 2 to find the hypothetical match M_v. Among the second-half vertices that have arrived, we iteratively find an unmatched edge between them with the largest utility value that satisfies all the constraints and add such edge into M_v if it exists.

The procedure is illustrated in Algorithm 3. In lines 3-7, we iteratively add an unmatched edge with the largest utility value into M_v if such edge exits. If no more such edge exists, M_v is returned as the result. And the whole procedure of the TGOA-Greedy algorithm is similar to Algorithm 2, except that line 14 is replaced by "$M_v^{Greedy} \leftarrow Greedy - Match$(second-half vertices in $T^{\Delta} \cup W^{\Delta} \cup \{v\}$)".

Example 4 (TGOA-Greedy Algorithm): Similar to the TGOA algorithm, based on the 1st arrival order in Table II, the TGOA-Greedy algorithm has same result of the TGOA algorithm the for first half of vertices. For the second half of vertices, TGOA-Greedy also allocates t_3 and t_4 to w_3. Thus, the total utility score of the TGOA-Greedy algorithm is 7+9+1=17 as well.

Competitive Ratio. We next analyze the competitive ratio of the TGOA-Greedy algorithm. As introduced above, the TGOA-Greedy algorithm follows the framework of Algorithm 2, except that line 14 calls Algorithm 3 instead of the Hungarian algorithm. The greedy strategy affects M_v^{Greedy} when v arrives at the platform. Therefore, we have the following lemma and theorem.

Lemma 6: $\mathbb{E}[MaxSum(M_v^{Greedy})] \geq \frac{|T^{\Delta}||W^{\Delta}|}{2mn} MaxSum(OPT)$

Proof: Let $M_{(|T^{\Delta}|,n)}^{Greedy}$ be the matching returned by Algorithm 3 for the offline weighted bipartite graph. According to [12], we know that $MaxSum(OPT_{(|T^{\Delta}|,n)}) \geq \frac{1}{2} MaxSum(M_{(|T^{\Delta}|,n)}^{Greedy})$, and thus we have

$$\mathbb{E}[MaxSum(M_v)] \geq \frac{|T^{\Delta}||W^{\Delta}|}{2mn} MaxSum(M_{(|T^{\Delta}|,n)}^{Greedy})$$

■

Theorem 2: The competitive ratio of the TGOA-Greedy algorithm under random order model is $\frac{1}{8}$.

TABLE III: Real Dataset

Platform	$\|T\|$	$\|W\|$	c_w	r_w	δ	due	p_t
gMission	713	532	[1,2,3]	1km	0.8	5 min	10.45
EverySender	4036	517	[1,2,$\cdots$,20]	1km	0.6	10 min	5.24

Proof: According to Lemmas 5 and 6, the final match returned by the TGOA-Greedy algorithm satisfies that

$$\begin{cases} \mathbb{E}[U(v,w) \in M] \geq \frac{\lfloor \frac{m+n}{2\epsilon} \rfloor}{2(|T^{\Delta}|-1)} \frac{|W^{\Delta}|}{mn} MaxSum(OPT) & v \in T \\ \mathbb{E}[U(t,v) \in M] \geq \frac{\lfloor \frac{m+n}{2\epsilon} \rfloor}{2(|W^{\Delta}|-1)} \frac{|T^{\Delta}|}{mn} MaxSum(OPT) & v \in W \end{cases}$$

Based on Theorem 1, we have $\mathbb{E}[MaxSum(M)] \geq \frac{1}{8} MaxSum(OPT)$. ■

Complexity Analysis. For each of the first half of new arrival tasks or workers, the time and space complexity of TGOA are $O(\max(|T|, |W|))$ and $O(1)$, respectively. For each of the second half of new arrival tasks or workers, the time and space complexity of TGOA are $O(|T^{\Delta}||W^{\Delta}| \log(|T^{\Delta}||W^{\Delta}|))$ and $O(\max(|T^{\Delta}|^2, |W^{\Delta}|^2))$, respectively.

VI. Experimental Study

A. Experimental Setup

We use two real datasets, the gMission dataset [25] and the EverySender dataset [4]. gMission is a research-based general spatial crowdsourcing platform. In the gMission dataset, every task has a task description, a location, a release time, a deadline and payoff. Each worker is also associated with a location, an available time, a deadline, the maximum activity range, and a success ratio based on his/her historical records on completing tasks. EverySender is a spatial crowdsourcing expressage platform on campus, where everyone on campus can post micro-tasks, e.g. buying some drinks or collecting a package, or conduct tasks as a crowd worker. Similar to the gMission dataset, each task and worker in the EverySender dataset also includes its corresponding information. Since capacity of workers is not given in the datasets, we generate the capacity of workers following uniform distribution. TABLE III presents the statistics of both real datasets. We also use synthetic dataset for evaluation. We generate the utility values following uniform and Normal distributions respectively. Furthermore, the arrival order of all tasks and workers follows uniform distribution following the online random order model. Statistics of the synthetic dataset are shown in TABLE IV, where we mark our default settings in bold font.

We evaluate the Greedy-RT, TGOA, and TGOA-Greedy algorithms in terms of total utility score, running time and memory cost, and study the effect of varying parameters on the performance of the algorithms. In each experiment, we repeatedly test 50 different online arrival orders of tasks and workers and report the average results. The algorithms are implemented in Visual C++ 2010, and the experiments were performed on a machine with Intel(R) Core(TM) i5 2.40GHz CPU and 4GB main memory.

B. Experiment Results

Effect of cardinality of W. The results of varying $|W|$ are shown in the first column of Fig. 2. For total utility score, we can first observe that the value increases as $|W|$ increases, which is reasonable since there are more matched edges as $|W|$ increases. Second, we can observe that the TGOA-based algorithms return better matching results than the baseline Extended Greedy-RT does. Finally, TGOA-Greedy returns slightly worse results than TGOA does. As for running time, we can first see that it reasonably increases when $|W|$ increases for all the algorithms. Another observation is that the TGOA algorithm is inefficient due to its cubic time complexity compared with the other two algorithms whose running time only increases slightly as $|W|$ increases. Finally, for memory consumption, we can see that it reasonably increases as $|W|$ increases. Also, the TGOA-based algorithms are less efficient than the baseline algorithm since they consume more space in storing results of a hypothetical global matching. Particularly, the TGOA algorithm is the most inefficient since it consumes more space when adopting the Hungarian algorithm.

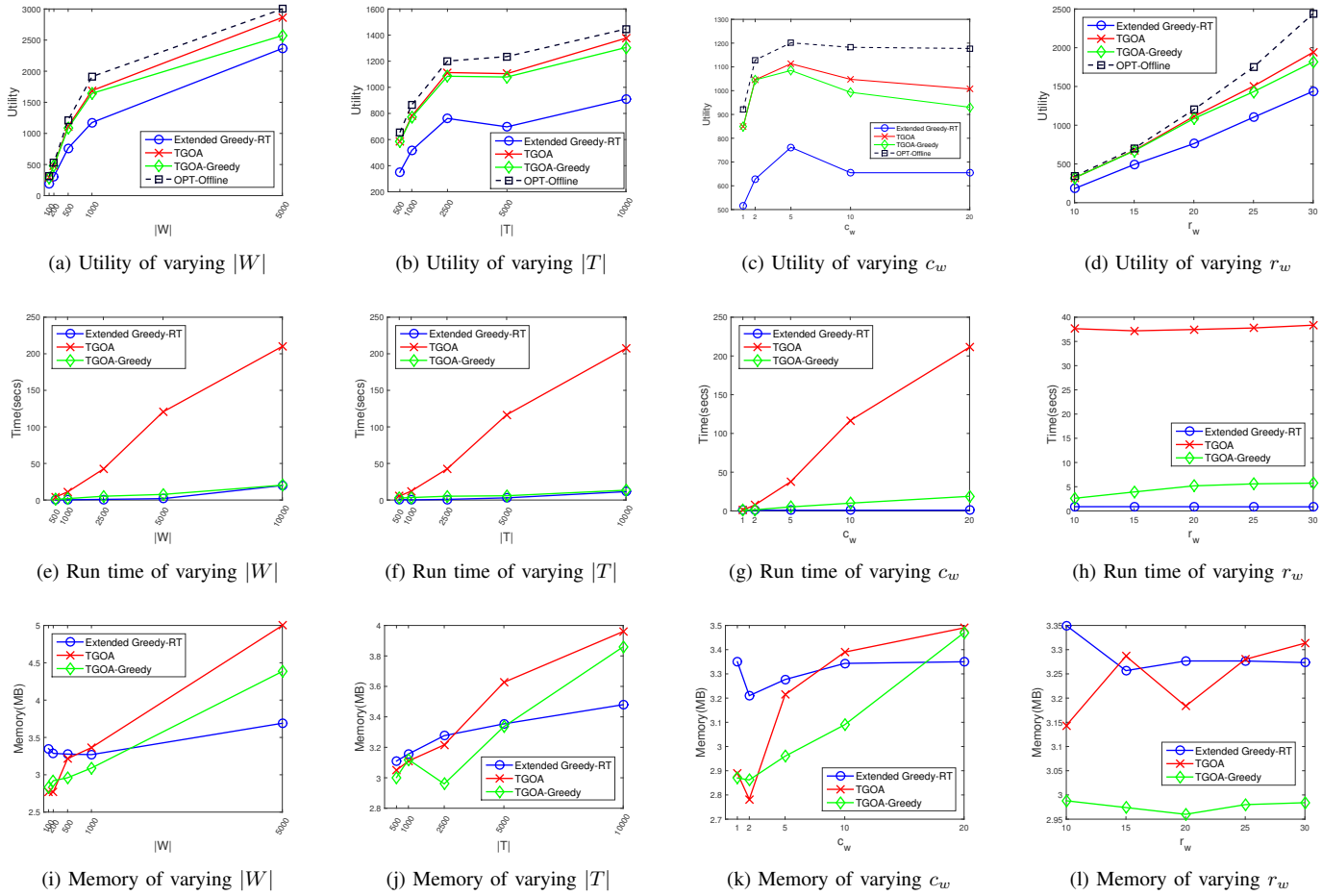
TABLE IV: Synthetic Dataset

Factor	Setting
$\|T\|$	500, 1000, **2500**, 5000, 10000
$\|W\|$	100, 200, **500**, 1000, 5000
c_w	1, 2, **5**, 10, 20
r_w	1.0, 1.5, **2.0**, 2.5, 3.0
δ_w	0.1, 0.3, **0.5**, 0.7, 0.9
Deadline	2, 4, **6**, 8, 10
p_t	2, 5, **10**, 20, 50
Scalability	$\|T\|$ = 10K, 20K, 30K, 40K, 50K, 100K $\|W\|$ = 500, 1000, 2500

Effect of cardinality of T. The results of varying $|T|$ are shown in the second column of Fig. 2. First for total utility score, we can observe that it generally increases when $|T|$ increases. And again, we can see that the TGOA-based algorithms perform much better, and the TGOA-Greedy algorithm is only slightly worse than the TGOA algorithm. For running time, we can observe similarly that the TGOA algorithm is inefficient compared with the TGOA-Greedy algorithm and the baseline. Finally, as for memory consumption, we can again see that the TGOA-based algorithms consume more space to store the results of the hypothetical global matching and all the three algorithms consume more space as $|T|$ increases.

Effect of capacity. The results of varying c_w are presented in the third column of Fig. 2. For total utility score, our first observation is that the value increases at first when c_w increases from 1 to 5 but then decreases when c_w is larger than 5. The reason is that when $c_w \leq 5$, the number of tasks is still larger than that of workers and thus more tasks can be assigned when c_w increases, resulting in larger total utility. However, when $c_w > 5$, there are more workers than tasks and thus the total utility cannot be larger as the total number of tasks is fixed. Also, when the capacity of workers increases, the workers who arrive earlier will be allocated more tasks whose utility may be small, and the workers who arrive late may not be allocated tasks with large utility. Therefore, the total utility score decreases when c_w increases. Again, the TGOA-based algorithms perform better than the baseline algorithm. As for running time, we can again observe that TGOA is the most inefficient. Finally, for memory consumption, all the algorithms generally consume more space as c_w increases. Notice that Extended Greedy-RT consumes more space than TGOA-Greedy does since it needs more space to store the

[4]http://www.dajiasong.com/



(a) Utility of varying $|W|$ (b) Utility of varying $|T|$ (c) Utility of varying c_w (d) Utility of varying r_w

(e) Run time of varying $|W|$ (f) Run time of varying $|T|$ (g) Run time of varying c_w (h) Run time of varying r_w

(i) Memory of varying $|W|$ (j) Memory of varying $|T|$ (k) Memory of varying c_w (l) Memory of varying r_w

Fig. 2: Results on varying $|T|$, $|W|$, capacity c_w, and radius r_w.

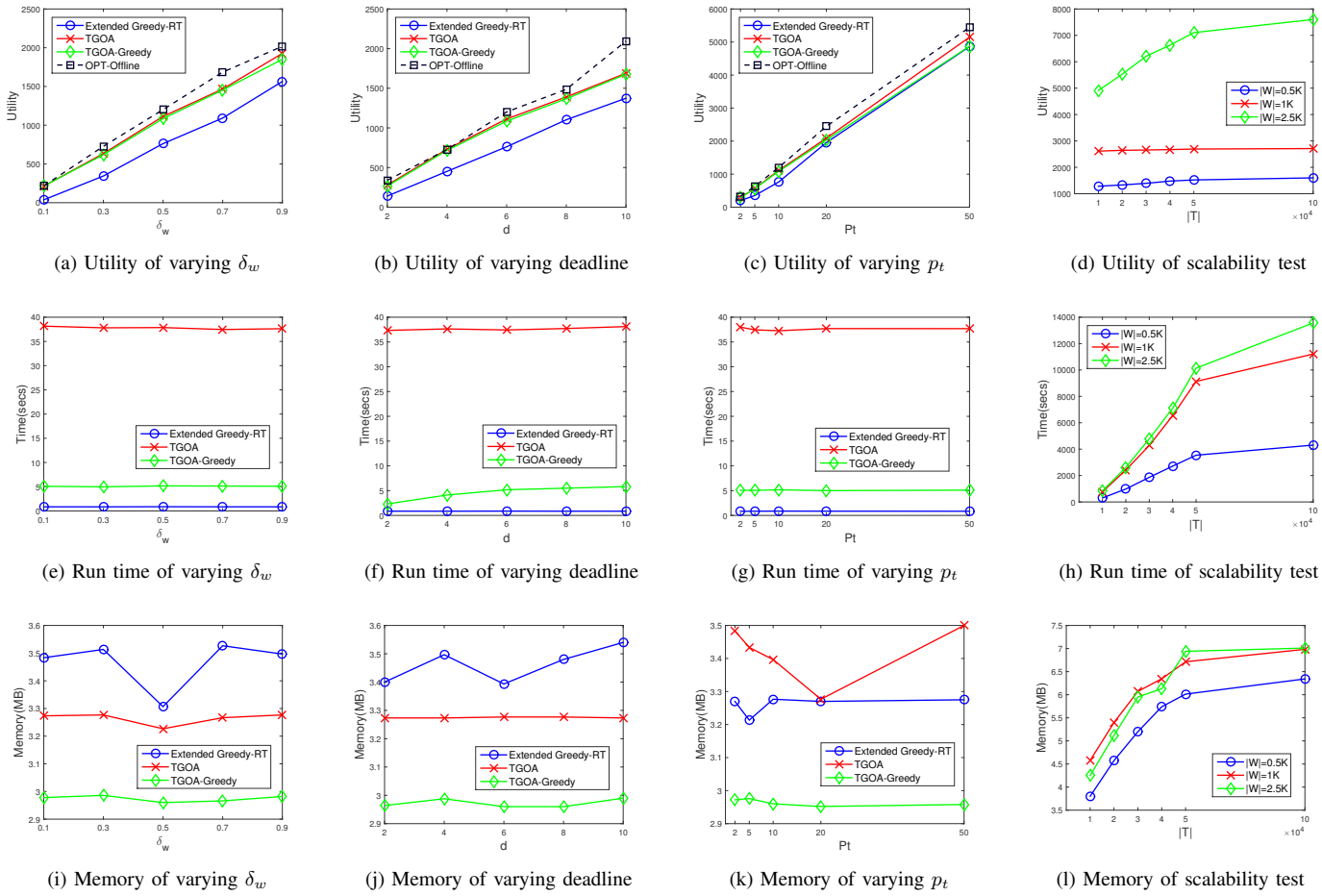
candidate neighbors when c_w increases and thus the number of unmatched neighbors also increases.

Effect of radius of restricted range. The results of varying the restricted range are shown in the last column of Fig. 2. For total utility score, we can first observe that the value increases when the restricted range increases, which is reasonable as workers can conduct more tasks. Also, we can again observe that the TGOA-based algorithms are more effective than the baseline algorithm and TGOA-Greedy is only slightly worse than TGOA. As for running time, again we can see that TGOA is the most inefficient. Note that the baseline algorithm is more efficient than the TGOA-Greedy algorithm. However, increasing restricted range of workers does not affect the running time of the algorithms too much. Finally, for memory consumption, we again observe that the TGOA-based algorithms consume more memory.

Effect of success ratio. The results of varying the success ratio of workers are presented in the first column of Fig. 3. For total utility score, we can see that the value increases with increasing success ratio of workers, which is reasonable as the utility of each pair of task and worker becomes larger. Also, the TGOA-based algorithms are again more effective than Extended Greedy-RT. For running time, we can again observe that TGOA is the most inefficient while Extended Greedy-RT is the most efficient. However, increasing success ratio of workers does not affect the running time of the algorithms. Finally, for memory consumption, we can see that TGOA consumes the most and the memory consumption of the three algorithms does not vary too much as the success ratio of workers varies.

Effect of deadline. The results of varying deadline are presented in the second column of Fig. 3. For total utility, our first observation is that the value increases when the deadline becomes longer, which is reasonable as more task-worker pairs can be matched. Also, again the TGOA-based algorithms perform better than the baseline, while TGOA-Greedy is nearly as good as TGOA. For running time, we can see that it does not vary too much when the response deadline increases. Again, TGOA is very inefficient compared with the other two algorithms. Finally, for memory consumption, TGOA is again the most inefficient and the memory consumption of all the algorithms does not vary much with varying deadline.

Effect of payoff. The results of varying the payoff of tasks are presented in the third column of Fig. 3. For total utility, we can first observe that its value increases with increasing payoff of tasks, which is reasonable as the utility of each task-worker pair becomes larger. Also, we can see that the TGOA algorithm performs slightly better than the other two algorithms. For running time, we can again see that TGOA is the slowest among the three. Also, varying the payoff of tasks does not affect the running time of the algorithms. Finally, for memory consumption, we can again see that TGOA is the



(a) Utility of varying δ_w (b) Utility of varying deadline (c) Utility of varying p_t (d) Utility of scalability test

(e) Run time of varying δ_w (f) Run time of varying deadline (g) Run time of varying p_t (h) Run time of scalability test

(i) Memory of varying δ_w (j) Memory of varying deadline (k) Memory of varying p_t (l) Memory of scalability test

Fig. 3: Results on varying success ratio δ_w, deadline, payoff p_t, and scalability.

most inefficient and all the three algorithms do not vary too much in memory when the payoff of tasks varies.

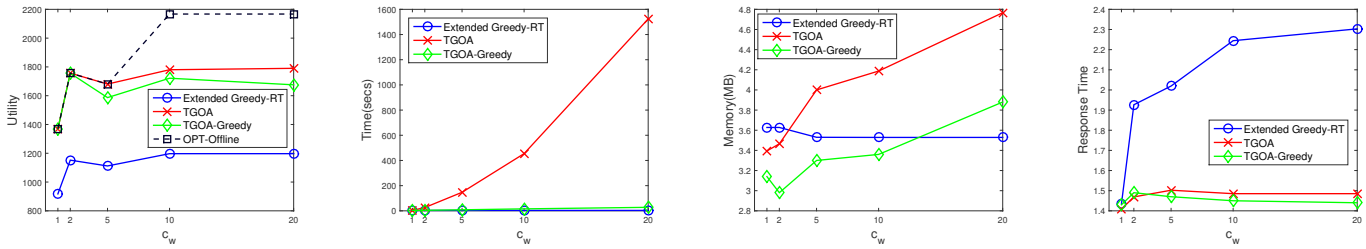
Scalability. The TGOA algorithm is not efficient enough according to our previous experiment results. Thus, we study the scalability of TGOA-Greedy in this part. The results are shown in the last column in Fig. 3. Specifically, we set $|W| = 500, 1000, 2500$ respectively, and vary the size of $|T|$. Since $|T|$ is relatively large, we set $\max c_v$ to 20. The other parameters are set to default. We observe that the memory cost of TGOA-Greedy grows linearly with the size of data. Also, the time cost of TGOA-Greedy grows nearly linearly with the size of data. The results show that TGOA-Greedy is scalable in both time and space.

Real dataset. Fig. 4 shows the results on real dataset (EverySender). Notice that the results on real dataset have similar patterns to those of the synthetic data. In particular, we test the average response time, $Time_{rs} - a_t$ ($Time_{rs} - a_w$), of each arrival task (worker) where $Time_{rs}$ and a_t (a_w) are the time the task/worker is allocated and the arrival time of each task (worker), respectively. Note that if a task (worker) is unmatched, its response time is $d_t - a_t$ ($d_w - a_w$). We can observe that the response time of the TGOA-based algorithms is much smaller than that of the baseline algorithm, and TGOA-Greedy has the least response time. This is because the TGOA-based algorithms avoid being trapped in local optimum and thus have better average response time. Similar patterns are observed on the gMission dataset and when the capacity values are generated following Normal distribution, and we omit the results due to limited space.

Conclusion. The algorithms using the two-phase-based framework always achieve the larger total utility value compared with the Extended Greedy-RT (baseline) algorithm. The TGOA-Greedy algorithm, though with a theoretically lower competitive ratio than that of the TGOA algorithm, significantly outperforms the TGOA algorithm in terms of running time and memory cost and is nearly as good as the TGOA algorithm in total utility. Finally, TGOA-Greedy is effective and also scalable in both terms of time and space in practice.

VII. Conclusion

In this paper, we identify a new online micro-task allocation problem, called *Global Online Micro-task Allocation in spatial crowdsourcing* ($GOMA$). We first analyze our differences with existing spatial crowdsourcing problems that assume offline scenario and traditional online maximum weighted bipartite matching (OMWBM) problems. Then, we extend the state-of-art algorithm under the online adversarial model for OMWBM problem as the baseline algorithm. Although the baseline algorithm guarantees the worst case, it does not perform well enough in practice. To find better solutions, we first clarify that the worse performance of the baseline algorithm is because the worst case happens with a



(a) Uility of EverySender (b) Run time of EverySender (c) Memory of EverySender (d) Response of EverySender

Fig. 4: Performance on the real dataset (EverySender).

very low probability in real world, and then propose a two-phase-based framework. Based on this framework, We present the TGOA algorithm, which has $\frac{1}{4}$-competitive ratio under the online random order model, which describes the average performance of an online algorithm, but is not scalable to large dataset due to its cubic time complexity. In order to improve the scalability, we further design the TGOA-Greedy algorithm following the two-phase-based framework, which runs faster than the TGOA algorithm but returns a result with a lower competitive ratio, $\frac{1}{8}$. We conduct extensive experiments which verify the efficiency, effectiveness and scalability of the proposed approaches.

ACKNOWLEDGMENT

This work is supported in part by the National Science Foundation of China (NSFC) under Grant No. 61502021, 61328202, 61532004, and 91118008, National Grand Fundamental Research 973 Program of China under Grant 2012CB316200, the Hong Kong RGC Project N_HKUST637/13, NSFC Guang Dong Grant No. U1301253, Microsoft Research Asia Gift Grant, Google Faculty Award 2013, and Microsoft Research Asia Fellowship 2012.

REFERENCES

[1] M. Musthag and D. Ganesan, "Labor dynamics in a mobile micro-task market," in *CHI 2013*.

[2] L. Kazemi and C. Shahabi, "Geocrowd: enabling query answering with spatial crowdsourcing," in *GIS 2012*.

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

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

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

[6] P. Cheng, X. Lian, Z. Chen, R. Fu, L. Chen, J. Han, and J. Zhao, "Reliable diversity-based spatial crowdsourcing by moving workers," *PVLDB 2015*.

[7] R. E. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems, Revised Reprint*, 2009.

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

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

[10] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani, "Adwords and generalized online matching," *Journal of the ACM*.

[11] H. Ting and X. Xiang, "Near optimal algorithms for online maximum edge-weighted b-matching and two-sided vertex-weighted b-matching," *Theoretical Computer Science*, 2015.

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

[13] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom, "Crowdscreen: algorithms for filtering data with humans," in *SIGMOD 2012*.

[14] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, "Crowder: Crowdsourcing entity resolution," *PVLDB 2012*.

[15] Y. Tong, C. C. Cao, C. J. Zhang, Y. Li, and L. Chen, "Crowdcleaner: Data cleaning for multi-version data on the web via crowdsourcing," in *ICDE 2014*.

[16] Y. Tong, C. C. Cao, and L. Chen, "Tcs: Efficient topic discovery over crowd-oriented service data," in *SIGKDD 2014*.

[17] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, "Crowddb: answering queries with crowdsourcing," in *SIGMOD 2011*.

[18] H. Park, R. Pang, A. G. Parameswaran, H. Garcia-Molina, N. Polyzotis, and J. Widom, "Deco: A system for declarative crowdsourcing," *PVLDB 2012*.

[19] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller, "Demonstration of qurk: a query processor for humanoperators," in *SIGMOD 2011*.

[20] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng, "Qasca: A quality-aware task assignment system for crowdsourcing applications," in *SIGMOD 2015*.

[21] J. Fan, G. Li, B. C. Ooi, K.-l. Tan, and J. Feng, "icrowd: An adaptive crowdsourcing framework," in *SIGMOD 2015*.

[22] C.-J. Ho and J. W. Vaughan, "Online task assignment in crowdsourcing markets," in *AAAI 2012*.

[23] L. Pournajaf, L. Xiong, V. Sunderam, and S. Goryczka, "Spatial task assignment for crowd sensing with cloaked locations," in *MDM*.

[24] Y. Li, M. Yiu, and W. Xu, "Oriented online route recommendation for spatial crowdsourcing task workers," in *SSTD 2015*.

[25] Z. Chen, R. Fu, Z. Zhao, Z. Liu, L. Xia, L. Chen, P. Cheng, C. C. Cao, Y. Tong, and C. J. Zhang, "gmission: A general spatial crowdsourcing platform," *PVLDB 2014*.

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

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

[28] C. Long, R. C.-W. Wong, P. S. Yu, and M. Jiang, "On optimal worst-case matching," in *SIGMOD 2013*.

[29] U. ul Hassan and E. Curry, "A multi-armed bandit approach to online spatial task assignment," in *UIC 2014*.

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

[31] Y. Wang and S. C.-w. Wong, "Two-sided online bipartite matching and vertex cover: Beating the greedy algorithm," in *ICALP 2015*.

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

[33] N. Korula and M. Pál, "Algorithms for secretary problems on graphs and hypergraphs," in *ICALP 2009*.

[34] T. Kesselheim, K. Radke, A. Tönnis, and B. Vöcking, "An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions," in *ESA 2013*.

[35] T. S. Ferguson, "Who solved the secretary problem?" *Statistical science*, 1989.