This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2948863, IEEE Transactions on Knowledge and Data Engineering

Two-sided Online Micro-Task Assignment in Spatial Crowdsourcing

Yongxin Tong, *Member, IEEE*, Yuxiang Zeng, Bolin Ding, Libin Wang, and Lei Chen, *Member, IEEE*

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. Many existing studies focus on the offline scenario, where all the spatiotemporal information of micro-tasks and crowd workers is given. In this paper, we focus on the online scenario and 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-the-art algorithm for the online maximum weighted bipartite matching problem to the GOMA problem as the baseline algorithm. Although the baseline algorithm provides a 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 the real world. Thus, we consider the average performance of online algorithms, *a.k.a.* random order model. We propose a two-phase-based framework, based on which we present the TGOA algorithm with a $\frac{1}{4}$-competitive ratio under the random order model. To improve its efficiency, we further design the TGOA-Greedy and TGOA-OP algorithm following this framework, which runs faster than the TGOA algorithm with a competitive ratio of $\frac{1}{8}$ and $\frac{1}{4}$, respectively. We also revisit the average performance of Greedy, which has long been considered as the worst due to its unbounded competitive ratio in the worst case. Finally, we verify the effectiveness and efficiency of the proposed methods through extensive experiments on synthetic and real datasets.

Index Terms—Spatial Crowdsourcing; Task Assignment; Online Bipartite Matching

1 Introduction

In recent years, spatial crowdsourcing has attracted much attention from the industry and the research communities, where crowd workers (workers for short) are paid to perform micro-tasks (tasks for short) using their mobile phones [1]. For example, on Gigwalk [2] and TaskRabbit [3], consulting companies recruit crowd workers to check the prices of products in supermarkets, and Waze [4] uses crowd workers to collect real-time information of traffic or remaining parking lots.

A central issue in spatial crowdsourcing is task assignment (*a.k.a.* task allocation) [1], [5], [6], [7], which aims to assign tasks to suitable workers such that the total number of assigned tasks or the total weighted value of the assigned task-worker pairs is maximized. However, many studies make the *offline scenario* assumption, where the spatiotemporal information of all the tasks and workers is known before task assignment. Therefore, they are inapplicable in real-time dynamic environments, where tasks and workers may appear anywhere at anytime and require immediate responses from the spatial crowdsourcing platforms. Imagine the following scenario. At noon at weekends, Tony wants to know how crowded his favorite restaurants around his home are so that he can decide which restaurant to go to for lunch without waiting in queues. Thus, Tony posts a task on a spatial crowdsourcing platform (*e.g.*, Gigwalk), and asks the crowd to take photos of the waiting queues at the restaurants. And he wants to receive immediate responses. Tasks like this arrive dynamically and require real-time response, and so do crowd workers. 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?*

In the offline scenario [1], the task assignment problem in spatial crowdsourcing can be solved by being reduced to the problem of maximum weighted bipartite matching [8], 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 tasks and workers. However, the offline solutions become infeasible in the online scenario since the arrival orders of tasks and workers are unknown. We illustrate this situation via the following toy example.

Example 1. *Suppose we have six micro-tasks t_1-t_6 and three crowd workers w_1-w_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 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 example, w_1-w_3 have capacities of 1, 3, and 2, respectively (in brackets). TABLE 1 presents the utility values between each pair of tasks and workers, which depends on the payoff of the task and the success ratio of the worker that can be inferred from how well*

- *Y. Tong and L. Wang are with the State Key Laboratory of Software Development Environment, School of Computer Science and Engineering, Beihang University, PR China. E-mail: {yxtong, lbwang}@buaa.edu.cn.*
- *Y. Zeng and L. Chen are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China. E-mail: {yzengal, leichen}@cse.ust.hk.*
- *B. Ding is with Alibaba Group, Bellevue, WA, USA. E-mail: bolin.ding@alibaba-inc.com*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2948863, IEEE Transactions on Knowledge and Data Engineering

TABLE 1: 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

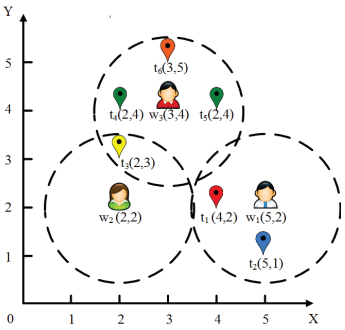


Fig. 1: Initial locations of micro-tasks and crowd workers.

this worker performed other tasks in history [1], [9]. In the offline scenario, the total utility of the optimal task assignment, which is marked in red in TABLE 1, is 17. *However, in the online scenario, 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 it is unknown which the next arrived task or worker is. For example, if the tasks and workers arrive following the "1st order" as shown in TABLE 2, w_1 can be randomly assigned to t_1 and w_2 is assigned to t_3 after their arrival. 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 a total utility of 17, which is exactly the optimal allocation in the offline scenario. It indicates that the effectiveness of an online task assignment significantly depends on the arrival orders of tasks and workers.*

In this paper, we propose a new task assignment problem in the online scenario, called the *Global Online Micro-task Allocation in spatial crowdsourcing* (GOMA) problem. A related branch of research is the online maximum weighted bipartite matching (OMWBM) problem [10], [11], [12], [13], [14], where the information of the left-hand vertices in a bipartite graph is known, while the right-hand vertices arrive dynamically. The GOMA problem mainly differs from the OMWBM problem in that both the tasks and the workers are dynamic. That is, the GOMA problem is *global online* or *two-sided online* and generalizes the OMWBM problem.

As the example above shows, the arrival order of tasks and workers significantly affects the performance of the solutions. Existing studies generally study the performance of the online algorithms on the worst-case arrival order (*i.e.*, adversarial order model). For example, Greedy-RT [14] achieves the best-known and nearly optimal guarantee under the adversarial order model, while the Greedy algorithm has been considered to be ineffective due to its unbounded guarantee. However, we find that the Extended Greedy-RT does not perform well in practice and Greedy performs much better in our GOMA problem. The reason is that the worst-case order rarely occurs in real-world applications. Hence we focus on the average performance of online algorithms, *i.e.*, random order model. We present four more effective algorithms with much better guarantees (*i.e.*, constant competitive ratio) under the random order model. The main contributions of this work are:

TABLE 2: Arrival time of micro-tasks and crowd workers.

Arrival Time	0	1	2	7	8	9	9	15	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

- We identify a new online task assignment problem in spatial crowdsourcing, *i.e.*, the GOMA problem.
- We extend the Greedy-RT algorithm as a baseline and propose three algorithms TGOA, TGOA-Greedy, and TGOA-OP. We analyze their competitive ratios under the *adversarial order model* and the *random order model*.
- We revisit the Greedy algorithm and our competitive analysis shows that the effectiveness of Greedy should not be arbitrarily bad on average, which is much more promising than the worst-case analysis.
- We verify the effectiveness and efficiency of the proposed methods through extensive experiments on real and synthetic datasets. In terms of effectiveness, our proposed TGOA, TGOA-Greedy, TGOA-OP and Greedy are up to $169.58\%, 93.21\%, 170.13\%, 167.07\%$ better than the baseline Extended Greedy-RT.

A preliminary version of this work can be found in [15], and we make the following new contributions: (1) We design a new algorithm TGOA-OP which is more effective and efficient by overcoming the drawbacks of TGOA (*i.e.*, unscalable) and TGOA-Greedy (*i.e.*, with 50% lower competitive ratio than TGOA) proposed in [15]. (2) We analyze the competitive ratios of the Greedy algorithm under the random order model with various widely-used distributions. (3) We conduct new evaluations on both synthetic and real datasets.

The rest of the paper is organized as follows. We formulate our problem in Sec. 2 and review related work in Sec. 3. We introduce our baseline in Sec. 4 and present three effective algorithms with constant competitive ratios under the random order model based on our two-phase-based framework in Sec. 5. Then we revisit the Greedy algorithm and analyze its competitive ratio in Sec. 6. Finally, we conduct experiments in Sec. 7 and conclude in Sec. 8.

2 Problem Statement

We first define the GOMA problem in Sec. 2.1 and then introduce two competitive analysis models in Sec. 2.2.

2.1 Problem Definitions

Definition 1 (Micro-Task). *A micro-task ("task" for short) is a tuple $t = \langle \boldsymbol{l}_t, a_t, d_t, p_t \rangle$, which is posted on the platform at the location $\boldsymbol{l}_t$ in the 2D space at time a_t. It can only be allocated to a crowd worker who arrives at the platform before the response deadline d_t with a payoff of p_t.*

Definition 2 (Crowd Worker). *A crowd worker ("worker" for short) is a tuple $w = \langle \boldsymbol{l}_w, a_w, d_w, r_w, c_w, \delta_w \rangle$, who arrives at the platform at the initial location $\boldsymbol{l}_w$ in the 2D space at time a_w. He/She can perform several tasks that arrive at the platform before his/her response deadline d_w with two constraints: (1) He/She can perform at most c_w tasks (i.e., his/her capacity); (2) He/She can only perform tasks posted within a circular range centered at $\boldsymbol{l}_w$ and with a radius of r_w. $\delta_w \in (0, 1]$ is the success ratio of w based on his/her historical task completion records.*

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2948863, IEEE Transactions on Knowledge and Data Engineering

Micro-tasks are usually simple, *e.g.*, taking a photo or checking prices in the supermarket. In practice, a spatial crowdsourcing platform usually uses a success ratio to represent the reliability of workers to complete these micro-tasks. Thus, the utility function not only maximizes the payoff of the workers but also guarantees the reliability of the assigned workers. Without loss of generality (*WLOG*), we assume that $U(t, w)$ is between 0 and U_{max}.

Finally, we define our *Global Online Micro-task Allocation in spatial crowdsourcing* (GOMA) problem.

Definition 4 (GOMA Problem)**.** *Given a set of tasks T, a set of workers W, and a utility function $U(.,.)$ on a spatial crowdsourcing platform, which is unaware of the spatiotemporal information of tasks and workers till their arrival, 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 has arrived at the platform before the response deadline d_t or is not assigned thereafter, and vice versa.*
- ***Invariable Constraint.*** *Once a task t is assigned to a worker w, the assignment of (t, w) cannot be changed.*
- ***Capacity Constraint.*** *The number of tasks assigned to a worker w cannot exceed his/her capacity c_w.*
- ***Range Constraint.*** *Any task assigned to a worker w must locate in the restricted range of w.*

In the GOMA problem, the total utility of the allocation can represent many practical meanings on real applications, including the total number of assigned tasks [16] ($\forall t, p_t = 1$ and $\forall w, \delta_w = 1$), the expected total number of assigned tasks [17], [18] ($\forall t, p_t = 1$), the total payoff of the workers [19], [20] ($\forall w, \delta_w = 1$), the expected total payoff of the workers [21]. Thus, the matching result with larger utility indicates better effectiveness on real applications.

2.2 Competitive Analysis Models

In the following, we first provide the offline version of the GOMA problem and then formally introduce the competitive analysis models, which is usually compared with the optimal result in the offline scenario.

The offline version of the GOMA problem ("offline GOMA" for short) is identical to the GOMA problem except that the spatiotemporal information of all the tasks and the workers is known at the beginning. The offline GOMA problem is unrealistic since both tasks and workers appear dynamically. The optimal solution to the offline GOMA problem (see our supplemental materials [22]) can be regarded as the upper bound of any online algorithm. *WLOG* we use OPT to denote the optimal assignment and MaxSum(OPT) to denote the optimal total utility.

Competitive ratio (CR) is a widely used metric to evaluate the performance of online algorithms [23]. In particular, the competitive ratio measures how good an online algorithm is compared with the optimal result of the offline version where all the information is provided. Based on different assumptions on the arrival order, we introduce two competitive analysis models for the GOMA problem, the *adversarial order 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.

Definition 5 (CR in the Adversarial Order Model)**.** *The competitive ratio in the adversarial order model of a specific online algorithm is the minimum ratio between the result of the online algorithm and the optimal result of the* ***worst-case arrival order*** *over all possible arrival orders of the tasks and the workers,*

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

where $G(T, W, U)$ is an arbitrary input of tasks, workers and their utilities, V is the set of all possible arrival 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 version.

Definition 6 (CR in the Random Order Model)**.** *The competitive ratio in the random order model of a specific online algorithm is the minimum ratio between the* ***expected*** *result of the online algorithm and the* ***expected*** *optimal result over all possible arrival orders of the tasks and the workers,*

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

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

3 Related Work

Our work is related to two categories of research, spatial crowdsourcing and online maximum bipartite matching.

3.1 Spatial Crowdsourcing

Recently, a wide spectrum of operations for crowdsourcing are studied and many crowd-powered database systems have been developed, such as similarity join [24], [25], SLADE [26], iCrowd [27], CDB [28], etc. Task assignment is one of the most important issues in crowdsourcing [29]. Existing studies usually study task assignment in the offline scenarios by learning the quality of crowd workers [27], [30]. All these studies focus on data management for traditional crowdsourcing rather than spatial crowdsourcing.

In spatial crowdsourcing, [1] firstly introduces the major taxonomy and [5], [7], [31] comprehensively describe the challenges of spatial crowdsourcing. [9] aims to maximize the expected total number of assigned tasks and [17], [18], [21] integrate the success ratio or reliability of workers into the task assignment problem. All these studies focus on the task assignment problem in the *offline scenario* and hence can not be applied in the real-time spatial crowdsourcing applications, where tasks and workers dynamically arrive.

Some recent works focus on the dynamic environment in spatial crowdsourcing. Specifically, [32] studies the task assignment problem of minimizing total latency in spatial crowdsourcing. [16], [33] aim to maximize the number of assigned tasks and [34] aims to minimize the total travel cost of the workers. [35] proposes a matching-based method to dynamically adjust the payoff of the tasks. [36], [37], [38] focus on maximizing the total utility in the online route planning problem, which are all different from our problem.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2948863, IEEE Transactions on Knowledge and Data Engineering

The closely related works are [19], [20], [39], [40], which also studies the online task assignment problems in spatial crowdsourcing. The main differences between these works and ours are summarized as follows:

(1) In our online scenario, all the tasks and workers can dynamically appear anywhere and anytime, but in [19], [20], [39] only tasks are dynamic. Besides, [19], [20], [40] assume the appearance of the tasks follows the independent and identical distribution (IID) and [40] also restricts that workers follow the IID distribution, which is impractical in real-world spatial data [41].
(2) In our problem setting, the workers may fail to complete the tasks, which is usually true in practice. However, [19], [20], [40] assume the workers will always complete the tasks.
(3) Our goal is to maximize the total utility of the assigned pairs of tasks and workers, while [39] maximizes the number of assigned tasks and [20], [40] maximize the total payoff the assigned tasks.

Therefore, the solutions in the above studies can not be extended to solve our problem.

3.2 Online Maximum Bipartite Matching

Our GOMA problem is related to the *Online Maximum Bipartite Matching (OMBM)* problem [23]. The input of the OMBM problem is a bipartite graph $G = (L, R, E)$, where L denotes the left-hand vertices, R denotes the right-hand vertices, and E denotes the set of edges. Specifically, the vertices in R always arrive one by one and the vertices in L either arrive at the beginning (*i.e.*, one-sided OMBM) or also dynamically appear (*i.e.*, two-sided OMBM). In the following, we classify the representative and recent works based on the objectives into two categories: *maximizing the matching size* and *maximizing the total weight*.

Maximizing the Matching Size. In practice, the size of the matching result equals to the number of the assigned tasks. Karp *et al.* [10] first study the one-sided version of this problem under the adversarial order model. They propose the Greedy algorithm with a competitive ratio of 0.5 and the Ranking algorithm with a better ratio of $1 - \frac{1}{e}$. The Greedy algorithm achieves the same competitive ratio in the two-sided version [42] and the competitive ratio of $1 - \frac{1}{e}$ under the random order model [43]. To beat the Greedy algorithm, [42] proposes a Water-filling algorithm with a ratio of 0.526 and [44] extends the Ranking algorithm to achieve the best-known ratio of 0.5541. However, extensive experiments on both synthetic and real datasets show that the Greedy algorithm always has a larger matching size in practice [41]. Thus, in this paper, we revisit the Greedy algorithm for the GOMA problem and analyze its competitive ratio.

Maximizing the Total Weight. The OMBM problem, whose objective is maximizing the total weight of the matching, is also known as *Online Maximum Weighted Bipartite Matching (OMWBM)* problem [23]. The weight of each edge (*i.e.*, utility) can be either determined by only the left-hand vertices [11], [12] or both left-hand and right-hand vertices [13], [14].

Since the weight of each edge in [11], [12] is determined by the vertices that are known at the beginning, both studies use different methods to define the priorities of allocating these known vertices L to the dynamically arriving vertices R. However, as all the vertices dynamically appear in GOMA, their solutions can not be applied in our problem.

[13], [14] study the *one-sided* OMBM problem with the same objective as our GOMA problem. [13] proposes a sample-and-price algorithm with a competitive ratio of 0.125. However, [13] does not support both the deadline constraint and the capacity constraint, and hence can not be applied in spatial crowdsourcing applications. Among these studies, only [14] can be extended to our GOMA problem since it does not rely on impractical assumptions and supports all the constraints. Moreover, [14] still achieves the *best-known* and *nearly optimal* competitive ratio under the adversarial order model. Hence, the extended solution of [14] is still the best baseline to assess the effectiveness and efficiency of our proposed algorithms. Besides, we also use the optimal results in the offline scenario to demonstrate the effectiveness of our algorithms in practice.

Algorithm 1: Extended Greedy-RT

```
   input : T, W, U(.,.)
   output: A feasible allocation M
1  θ ← ⌈ln(U_max + 1)⌉;
2  k ← randomly choose an integer from
     {0, ··· , θ − 1};
3  foreach newly arrived task or worker v do
4  |  Cand ← {∀u|u is an unmatched neighbor of v
   |    such that U(u, v) ≥ e^k and it satisfies all the
   |    constraints};
5  |  if Cand is not empty then
6  |  |  u* ← an arbitrary item is chosen from Cand;
7  |  |  M ← M ∪ {(u*, v)};
8  return M;
```

4 Baseline Algorithm

In this section, we extend the Greedy-RT algorithm [14] as our baseline. The Greedy-RT algorithm has the best-known and nearly optimal competitive ratio under the adversarial order model for the *one-sided* OMWBM problem, where only left-hand vertices in the bipartite graph arrive online.

Basic Idea. The basic idea of the Extended Greedy-RT algorithm is to first randomly choose a threshold on the weights of edges, and then randomly choose an edge incident to each newly arrived vertex among those edges whose weights are no less than the threshold.

Algorithm Details. Algo. 1 illustrates the procedure of the Extended Greedy-RT algorithm. In lines 1-2, we first randomly choose a threshold (e^k) on the weights of edges according to the estimated maximum weight U_{max}. When a new vertex (either a task or a worker) arrives, Extended Greedy-RT adds an edge among the ones whose weights are no less than a threshold and that satisfy all the constraints to the match result in lines 3-7. If a worker with capacity c_w arrives, Algo. 1 regards him/her as c_w duplicates of w that arrive at the same time and processes them one by one.

Example 2. *Back to our running example in Example 1. Algo. 1 sets* $\theta = \lceil \ln(9+1) \rceil = 3$, *so* $k \in \{0, \cdots, 2\}$. *If* k *is chosen as 0, the threshold is* $e^0 = 1$. *According to the 1st arrival order in TABLE 2, when* w_1 *arrives, the candidate set is* $\{t_1, t_2\}$, *and we*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2948863, IEEE Transactions on Knowledge and Data Engineering

would 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 is 10. Since Algo. 1 is a randomized algorithm on choosing k, the expectation of the total utility for all possible k is $\frac{10+16+16}{3} = 14$.

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

Competitive Analysis. Since Extended Greedy-RT (Algo. 1) is a randomized algorithm, we analyze the (expected) competitive ratio under the *adversarial order model* in Lemma 1.

Lemma 1. *The competitive ratio of Extended Greedy-RT algorithm is $\frac{1}{2e\lceil \ln(U_{max}+1)\rceil}$ under the adversarial order model.*

Proof. Please see our supplemental materials [22]. □

Although Extended Greedy-RT provides a theoretical guarantee for the worst case by randomization, the worst-case arrival order only appears with a low probability ($\frac{1}{n!}$) if the total number of tasks and workers is n. This is because the worst-case arrival order is only one among $n!$ different possible arrival orders. We argue that the average performance of online algorithms is more important in real-world applications. Below we design algorithms with effective competitive ratios under the random order model, which measures the average performance of online algorithms.

5 A Two-Phase-Based Framework

This section presents a two-phase-based framework for the GOMA problem. Based on the framework, we first present a *Two-phase-based Global Online Allocation (TGOA)* algorithm with a competitive ratio of $\frac{1}{4}$. To improve its time efficiency, we further present the TGOA-Greedy algorithm, which is faster than the TGOA algorithm but with a slightly lower competitive ratio of $\frac{1}{8}$. Finally, we propose TGOA-OP based on the same framework, which is also more efficient but keeps the competitive ratio $\frac{1}{4}$.

5.1 TGOA Algorithm

Basic Idea. Inspired by the solution to the secretary problem [13], our basic idea is to first divide all the vertices (both tasks and workers) into two equal groups based on their arrival orders and adopt different strategies on them.

- For the first half of tasks and workers, TGOA conducts a greedy strategy to assign each newly arrived task (worker) to the corresponding worker (task) with the highest utility and satisfying all the constraints.
- For the other half of tasks and workers (*i.e.*, the second half of vertices), we adopt a more optimal strategy. Specifically, among the second-half vertices that have arrived, including the newly arrived v, we hypothetically find a global optimal match M_v using the Hungarian algorithm [8]. 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 a vertex has not been assigned and satisfies all the constraints.

Algorithm Details. Algo. 2 illustrates the procedure of TGOA. In line 1, we calculate the number of the first half of vertices $k \leftarrow \lfloor \frac{m+n}{2} \rfloor$, where m is the total number of tasks and n is the total capacities of workers. That is, for each worker with capacity c_w, we treat him/her as c_w duplicated workers who arrive at the same time. The values of m and n can be estimated by historical records. In line 2, we initialize two sets to store the vertices that have arrived. Note that W^Δ is a multiset. Then in lines 3-21, we iteratively process each newly arrived task or worker. Particularly, we adopt a greedy strategy on the first half of the vertices in lines 4-10 and adopt a more optimal strategy on the second half of the vertices in lines 12-19. For each vertex in the first half, we either assign a task with the highest utility that satisfies all the constraints if it is a worker (lines 6-7) or assign a worker with the largest utility that satisfies all the constraints if it is a task (lines 9-10). Then for each vertex in the second half, we first run the Hungarian algorithm [8] on the vertices that have arrived to obtain a current global optimal match M_v in line 12. 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 (lines 15-16) 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 (lines 18-19). Finally, the sets T^Δ and W^Δ are updated in lines 20-21.

Algorithm 2: TGOA

```
   input : T, W, U(.,.)
   output: A feasible allocation M
1  m ← |T|, n ← Σ_{w∈W} c_w, k ← ⌊(m+n)/2⌋;
2  Multiset W^Δ ← ∅, T^Δ ← ∅;
3  for each newly arrived task or worker v do
4      if |T^Δ| + |W^Δ| < k then            // Phase I: greedy strategy
5          if v is a worker then
6              t ← the task with the highest utility that is unmatched and satisfies all constraints;
7              if t exists then M ← M ∪ {(t, v)} ;
8          else
9              w ← the worker with the highest utility that is unmatched and satisfies all constraints;
10             if w exists then M ← M ∪ {(v, w)} ;
11     else                                 // Phase II: optimal strategy
12         M_v ← Hungarian(T^Δ ∪ W^Δ ∪ {v});
13         if v is matched in M_v then
14             if v is a worker then
15                 t ← the task assigned to v in M_v;
16                 if t is unmatched in M and satisfies all constraints then M ← M ∪ {(t, v)} ;
17             else
18                 w ← the worker assigned v in M_v;
19                 if w is unmatched in M and satisfies all constraints then M ← M ∪ {(v, w)} ;
20     if v is a worker then
           W^Δ ← W^Δ ∪ {c_v duplicated workers v} ;
21     else T^Δ ← T^Δ ∪ {v} ;
22 return M;
```

Example 3. *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 2, for the first half of the vertices, i.e., the first 6 arrived tasks and duplicates of workers, TGOA assigns t_1 to w_1. For the second half of the*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2948863, IEEE Transactions on Knowledge and Data Engineering

vertices, t_3 and t_4 are assigned to w_3. Therefore, the total utility is 7+9+1=17, which is better than Extended Greedy-RT.

Complexity Analysis. For each task or worker in the first half, the time and space complexities of TGOA are both $O(\max(|T|,|W|))$. For each task or worker in the second half, the time and space complexities of TGOA are $O(\max(|T^{\Delta}|^3,|W^{\Delta}|^3))$ and $O(\max(|T^{\Delta}|^2,|W^{\Delta}|^2))$.

Competitive Analysis. For convenience, we use a variant of TGOA, TGOA-Filter, which ignores the first half of tasks and workers and only performs lines 12-21 of TGOA for the second half of tasks and workers. Hence the competitive ratio of TGOA must be larger than that of TGOA-Filter. The following analysis assumes that TGOA-Filter filters out the first $\lfloor\frac{m+n}{2}\rfloor$ arrived items in the first half.

The analysis about TGOA-Filter relies on the fact that, for each newly arrived item v, the probability of $v\in T$ (v being a task) equals that of $v\in W$ (v being a worker). For the first $\lfloor\frac{m+n}{2}\rfloor$ arrived items that are filtered out by TGOA-Filter, we can assume that $\lfloor\frac{m+n}{4}\rfloor$ tasks and $\lfloor\frac{m+n}{4}\rfloor$ workers are filtered out, respectively. When TGOA-Filter processes the i-th arrived item v ($i\in[\lfloor\frac{m+n}{2}\rfloor+1,m+n]$), $|T^{\Delta}|$ tasks and $|W^{\Delta}|$ workers have already arrived. Hence, we have the following lemmas.

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

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

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

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

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

Therefore,

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

□

Based on Lemma 2, we can derive the following lemma.

Lemma 3.

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

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 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

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

According to Lemma 2, we have

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

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

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

□

Lemma 2 and Lemma 3 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 iff 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 $(\lfloor\frac{m+n}{2}\rfloor+1)$-th to $(i-1)$-th items arrive.

Lemma 4.

$$\begin{cases}Pr(w\text{ is unmatched in steps }[\lfloor\frac{m+n}{2}\rfloor+1,|T^{\Delta}|-1])\geq\frac{\lfloor\frac{m+n}{4}\rfloor}{|T^{\Delta}|-1} & v\in T\\ Pr(t\text{ is unmatched in steps }[\lfloor\frac{m+n}{2}\rfloor+1,|W^{\Delta}|-1])\geq\frac{\lfloor\frac{m+n}{4}\rfloor}{|W^{\Delta}|-1} & v\in W\end{cases}$$

Proof. If $v\in T$, we construct $T^{'}=\{t_{\lfloor\frac{m+n}{4}\rfloor+1},\cdots,t_{|T^{\Delta}|-1}\}$, which is the set of the tasks that arrive after the $(\lfloor\frac{m+n}{4}\rfloor)$-th task but before v. Let $t_j\in T'$ be the j-th task which arrives, then worker w is assigned to t_j with probability at most $\frac{1}{j}$ since at least the first $j-1$ arriving tasks cannot be matched to w. That is, for the j-th arrived task, w is unmatched with probability at least $\frac{j-1}{j}$. Hence we have

$$\begin{aligned}&\Pr(w\text{ is unmatched in steps }[\lfloor\frac{m+n}{2}\rfloor+1,|T^{\Delta}|-1])\\ &\geq\prod_{j=\lfloor\frac{m+n}{4}\rfloor+1}^{|T^{\Delta}|-1}\frac{j-1}{j}=\frac{\lfloor\frac{m+n}{4}\rfloor}{|T^{\Delta}|-1}\end{aligned}$$

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

$$\Pr(t\text{ is unmatched in steps }[\lfloor\frac{m+n}{2}\rfloor+1,|W^{\Delta}|-1])\geq\frac{\lfloor\frac{m+n}{4}\rfloor}{|W^{\Delta}|-1}$$

□

Lemma 3 and Lemma 4 lead to the following lemma.

Lemma 5.

$$\begin{cases}\mathbb{E}[U(v,w)\in M]\geq\frac{\lfloor\frac{m+n}{4}\rfloor}{|T^{\Delta}|-1}\cdot\frac{|W^{\Delta}|}{mn}\mathbb{E}[MaxSum(OPT)] & v\in T\\ \mathbb{E}[U(t,v)\in M]\geq\frac{\lfloor\frac{m+n}{4}\rfloor}{|W^{\Delta}|-1}\cdot\frac{|T^{\Delta}|}{mn}\mathbb{E}[MaxSum(OPT)] & v\in W\end{cases}$$

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 $(\lfloor\frac{m+n}{2}\rfloor+1)$-th to $(i-1)$-th items arrive. According to Lemma 3 and Lemma 4, we can obtain the expectation of the utility $U(v,w)\in M$

$$\mathbb{E}[U(v,w)\in M]\geq\frac{\lfloor\frac{m+n}{4}\rfloor}{|T^{\Delta}|-1}\times\frac{|W^{\Delta}|}{mn}\mathbb{E}[\text{MaxSum}(OPT)]$$

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

$$\mathbb{E}[U(t,v)\in M]\geq\frac{\lfloor\frac{m+n}{4}\rfloor}{|W^{\Delta}|-1}\times\frac{|T^{\Delta}|}{mn}\mathbb{E}[\text{MaxSum}(OPT)]$$

□

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

Proof. Let U_v be the utility contributed by v in the final match M. According to Lemma 5, v can represent either a worker or a task. Thus, we have

$$\begin{aligned}\mathbb{E}[\sum_{v=1}^{m+n}U_v]&=\sum_{v=1}^{m+n}\mathbb{E}[U_v]\\ &=\frac{1}{2}\sum_{v=1}^{m+n}\big(\mathbb{E}[U(v,w)\in M]+\mathbb{E}[U(t,v)\in M]\big)\end{aligned}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2948863, IEEE Transactions on Knowledge and Data Engineering

Algorithm 3: Greedy-Match

input : A bipartite graph
output: An offline allocation M_v^{Gdy}
1 $M_v^{Gdy} \leftarrow \emptyset$;
2 **while** *True* **do**
3 $(t, w) \leftarrow$ a feasible pair of tasks and workers with the highest utility that is unmatched;
4 **if** (t, w) *exists* **then** $M_v^{Gdy} \leftarrow M_v^{Gdy} \cup \{(t, w)\}$;
5 **else** break ;
6 **return** M_v^{Gdy};

In addition, in the online random order model, an arbitrary order must have a corresponding symmetrical order. In other words, if a newly arrived item v is a task, there must be an item with the same arrival order that is a worker. Therefore, we have

$$\begin{aligned}
&\mathbb{E}[\text{MaxSum}(M)] = \frac{1}{2}\mathbb{E}[\sum_{v=1}^{m+n} U_v] \\
&\geq \frac{1}{2}\cdot\frac{1}{2}\sum_{v=\lceil\frac{m+n}{2}\rceil}^{m+n}(\frac{\lfloor\frac{m+n}{4}\rfloor}{|W^{\Delta}|-1}\cdot\frac{|T^{\Delta}|}{mn}\mathbb{E}[\text{MaxSum}(OPT)]+ \\
&\qquad\qquad \frac{\lfloor\frac{m+n}{4}\rfloor}{|T^{\Delta}|-1}\cdot\frac{|W^{\Delta}|}{mn}\mathbb{E}[\text{MaxSum}(OPT)]) \\
&\geq \frac{1}{2}\sum_{i=\lceil\frac{m+n}{2}\rceil}^{m+n}\frac{\frac{m+n}{4}}{mn}\mathbb{E}[\text{MaxSum}(OPT)] \\
&= \frac{\lfloor\frac{m+n}{4}\rfloor}{2mn}(m+n-\lceil\frac{m+n}{2}\rceil+1)\mathbb{E}[\text{MaxSum}(OPT)] \\
&\geq \frac{1}{2}\cdot(1-\frac{1}{2})\mathbb{E}[\text{MaxSum}(OPT)]
\end{aligned}$$

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

5.2 TGOA-Greedy Algorithm

Basic Idea. Although the TGOA algorithm adopts a more optimal strategy on the second half of vertices, it is inefficient because it takes cubic time complexity to find the current global optimal match for each vertex in the second-half. To improve the efficiency, we replace the optimal Hungarian algorithm with a greedy strategy, which leads to a slightly lower competitive ratio.

Algorithm Details. We replace the Hungarian algorithm with the greedy strategy as illustrated in Algo. 3. In lines 3-5, we iteratively add an unmatched edge with the highest utility into M_v if such an edge exits. If no such edge exists, M_v^{Gdy} is returned. In line 12 of Algo. 2, TGOA will find the hypothetical match M_v. Thus, TGOA-Greedy is similar to TGOA, except that line 12 is replaced by "$M_v \leftarrow$ Greedy-Match$(T^{\Delta} \cup W^{\Delta} \cup \{v\})$".

Example 4. *Based on the 1st arrival order in TABLE 2, TGOA-Greedy yields the same result as TGOA for the first half of vertices. For the second half of vertices, TGOA-Greedy also assigns t_3 and t_4 to w_3. Thus, the total utility of TGOA-Greedy is 7+9+1=17 as well.*

Complexity Analysis. For each task or worker in the first half, the time and space complexities of TGOA-Greedy are the same as TGOA. For each task or worker in the second half, we implement Algo. 3 by using a heap. Thus, its time and space complexities are $O(|T^{\Delta}||W^{\Delta}|\log(|T^{\Delta}||W^{\Delta}|))$ and $O(\max(|T^{\Delta}|^2, |W^{\Delta}|^2))$, respectively.

Competitive Analysis. The TGOA-Greedy algorithm follows the framework of Algo. 2, except that line 12 calls Algo. 3 instead of the Hungarian algorithm. The greedy strategy affects M_v^{Gdy} when v arrives at the platform. Hence we have the following lemma and theorem.

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

Proof. Please see our supplemental materials [22]. □

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

Proof. Please see our supplemental materials [22]. □

5.3 TGOA-OP Algorithm

Although TGOA-Greedy is faster than TGOA, it has a worse competitive ratio. Thus, we propose TGOA-OP to improve the efficiency while retaining the same competitive ratio.

Basic Idea. The basic idea of TGOA-OP is to keep removing the tasks and workers whose deadlines have passed, such that the number of available task and workers is much smaller than the total number of tasks and workers.

Algorithm Details. We simply remove workers/tasks whose deadlines have passed after line 21 of TGOA in Algo. 2. That is, we remove workers/tasks from W^{Δ} and T^{Δ} whose deadlines are earlier than the current time.

Complexity Analysis. *WLOG* denote $\mathcal{T}$ and $\mathcal{W}$ as a set of tasks and a set of workers, which have the maximum cardinality among all iterations. For each task/worker in the first half, the time and space complexities of TGOA-OP are both $O(\max(|\mathcal{T}|, |\mathcal{W}|))$. For each task/worker in the second half, the time and space complexities of TGOA-OP are $O(\max(|\mathcal{T}|^3, |\mathcal{W}|^3))$ and $O(\max(|\mathcal{T}|^2, |\mathcal{W}|^2))$, respectively. Since $|\mathcal{T}| \ll |T|$ and $|\mathcal{W}| \ll |W|$ in practice, the efficiency of TGOA-OP is better than TGOA and TGOA-Greedy.

Competitive Analysis. After removing unavailable tasks and workers, the input of bipartite for Hungarian algorithm will be changed, *i.e.*, line 12 of Algo. 2. Although the final allocation M may change, the competitive ratio is still $\frac{1}{4}$.

Theorem 3. *The competitive ratio of TGOA-OP under the random order model is $\frac{1}{4}$.*

Proof. Both Lemma 2 and Lemma 3 still hold since the optimal allocation should satisfy the deadline constraint.

We next prove that Lemma 4 still holds. If $v \in T$, the set $T' = \{t_{\lfloor\frac{m+n}{4}\rfloor+1}, \cdots, t_{|T^{\Delta}|-1}\}$ will be refined since we remove the tasks whose deadlines are passed. *WLOG* the set T' changes into $\{t_k, \cdots, t_{|T^{\Delta}|-1}\}$, where $k \geq \lfloor\frac{m+n}{4}\rfloor + 1$. With the same reason in Lemma 4, we have

$$\begin{aligned}
&\Pr\big(w \textit{ is unmatched in steps } [\lfloor\frac{m+n}{2}\rfloor+1, |T^{\Delta}|-1]\big) \\
&\geq \prod_{j=k}^{|T^{\Delta}|-1}\frac{j-1}{j} = \frac{k-1}{|T^{\Delta}|-1} \geq \frac{\lfloor\frac{m+n}{4}\rfloor}{|T^{\Delta}|-1}
\end{aligned}$$

Similarly, we obtain the same result in Lemma 4 if $w \in W$.

Since both Lemma 3 and Lemma 4 still hold, the Lemma 5 (based on Lemma 3 and Lemma 4) and Theorem 1 (based on Lemma 5) also hold. □

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2948863, IEEE Transactions on Knowledge and Data Engineering

Algorithm 4: Greedy

```
input : T, W, U(.,.)
output: A feasible allocation M
1 foreach newly arrived task or worker v do
2     Cand ← {∀u|u is an unmatched neighbor of v
        and satisfies all constraints};
3     if Cand is not empty then
4         u* ← the element with the largest U(u, v) is
            chosen from Cand;
5         M ← M ∪ {(u*, v)};
6     remove workers/tasks whose deadlines have
        passed;
7 return M;
```

6 Greedy Algorithm Revisited

Greedy has been considered as an ineffective algorithm in terms of the competitive ratio under the adversarial order model for the OMWBM problem [10], [12], [13], [14], [44], *i.e.*, a special case of our GOMA problem. However, recent experimental studies show that Greedy performs well in online task assignment with other objectives, *e.g.*, maximizing the total number of assigned tasks [41] or minimizing the total travel cost of the workers [34]. Thus, it remains open *how Greedy performs on average for the GOMA problem* and we are motivated to revisit the Greedy algorithm in the following. Specifically, we review Greedy and its worst-case performance under the adversarial order model in Sec. 6.1. We then analyze its competitive ratio under the random order model in Sec. 6.2, which shows its average-case performance.

6.1 Greedy Algorithm

Basic Idea. The basic idea of Greedy is to assign each new vertex to its unmatched neighbor with the highest utility such that all constraints are satisfied.

Algorithm Details. Algo. 4 presents the procedure of Greedy. In line 2, we select all unmatched neighbors of the new vertex v as a candidate set. In lines 3-5, we greedily assign the neighbor with the highest utility to v if feasible. In line 6, we safely remove tasks/workers whose deadlines have expired.

Example 5. *Back to our running example in Example 1. It is the same situation where the threshold of Extended Greedy-RT equals 0. According to the 1st arrival order in TABLE 2, when w_1 arrives, the candidate set $Cand = \{t_1, t_2\}$, and the algorithm chooses t_1. Similarly, t_3 and t_4 are assigned to w_2 and w_3, respectively. Greedy obtains a total utility of 10.*

Complexity Analysis. For each newly arrived task or worker, the time and space complexities of Greedy are both $O(\max(|\mathcal{T}|, |\mathcal{W}|))$.

Competitive Analysis under Adversarial Order Model.

Theorem 4. *The competitive ratio of Greedy is unbounded under the adversarial order model.*

Proof. Suppose there are two edges with utilities of ε and U_{max} in the bipartite graph, and they have a common endpoint. In the worst-case order, Greedy first meets the edge with utility of ε and then chooses it, while it fails to match the edge with utility of U_{max}. Meanwhile OPT achieves a final utility score of U_{max}. Thus, the competitive ratio of Greedy is at most $\frac{\varepsilon}{U_{max}}$ under adversarial order model, which is unbounded since ε can be arbitrarily small. □

As mentioned in Sec. 4, we focus on the average performance of an algorithm to the GOMA problem since the worst-case instance rarely occurs in practice. Despite its unbounded competitive ratio under the adversarial order model, Greedy can be still useful if it has reasonable competitive ratios under the random order model, which we will analyze in detail below.

6.2 Competitive Ratio Analysis

In this subsection we analyze the competitive ratio of Greedy under the random order model by assuming different distributions of utilities between tasks and workers. We focus on three widely used distributions, *i.e.*, uniform, exponential and normal, since they can approximate the utilities between tasks and workers [15], [45], [46].

6.2.1 Analysis Under Uniform Distribution

Assume that all the utilities of edges in the bipartite graph follow the uniform distribution, we analyze the competitive ratio of Greedy under the random order model. Let X_i be a random variable representing the utility between v and the i-th neighbor in the candidate set $Cand$ (line 2 of Greedy) and let p denote the size of $Cand$. Then, all the random variables $X_1, \cdots, X_p$ are independent and uniformly distributed in $[0, U_{max}]$. Let X^* denote the highest utility in line 4 of Greedy, *i.e.*, $X^* = \max\{X_1, X_2, \ldots, X_p\}$.

Theorem 5. *If the utilities follow the uniform distribution, the competitive ratio of Greedy is $\frac{1}{2}(1 - \frac{1}{e})$ under the random order model.*

Proof. With respect to each random variable X_i, the cumulative distribution function (CDF for short) is defined by

$$F(x) = \Pr(X_i \leq x) = \frac{x}{U_{max}}, \quad x \in [0, U_{max}] \tag{3}$$

Thus, the CDF with respect to X^* is defined by

$$F^*(x) = \Pr(X_1 \leq x) \cdot \Pr(X_2 \leq x) \cdots \Pr(X_p \leq x) = [F(x)]^p$$

Then we can obtain the probability density function by calculating the derivative of CDF with respect to x,

$$f^*(x) = p[F(x)]^{p-1} \cdot f(x) = \frac{px^{p-1}}{(U_{max})^p} \tag{4}$$

Hence the expected value of X^* is calculated by the integration with respect to the probability density function.

$$\mathbb{E}[X^*] = \int_0^{U_{max}} x f^*(x)\, dx = \frac{p}{p+1} U_{max} \geq \frac{1}{2} U_{max} \tag{5}$$

Since [43] proves that $\mathbb{E}[|M|] \geq (1 - \frac{1}{e})\mathbb{E}[|OPT|]$ under the random order model, we can bound the expected total utility of Greedy as follows.

$$\mathbb{E}[\text{MaxSum}(M)] \geq (1 - \frac{1}{e})\mathbb{E}[|OPT|] \cdot \frac{1}{2} U_{max} \tag{6}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2948863, IEEE Transactions on Knowledge and Data Engineering

Since the expected total is at most $\mathbb{E}[|OPT|] \cdot U_{max}$, we have

$$CR_{RO} = \frac{\mathbb{E}[\text{MaxSum}(M)]}{\mathbb{E}[\text{MaxSum}(OPT)]} \geq \frac{1}{2}(1 - \frac{1}{e}) \sim 0.32$$

□

6.2.2 Analysis Under Exponential Distribution

Similar to the analysis in Sec. 6.2.1, we assume that each random variable X_i follows an exponential distribution with parameter λ. Then we have the following conclusion on the competitive ratio of Greedy.

Theorem 6. *If the utilities follow the exponential distribution (with parameter λ), the competitive ratio of Greedy is $(1 - \frac{1}{e}) \cdot \frac{1}{\lambda U_{max}}$ under the random order model.*

For example, if we let $1/\lambda$ equals to $0.5U_{max}$, the ratio is asymptotic to $(1 - \frac{1}{e}) \cdot \frac{1}{2} \sim 0.32$.

Proof. Similarly, the CDF with respect to X_i is defined by

$$F(x) = \Pr(X_i \leq x) = \frac{1 - e^{-\lambda x}}{1 - e^{-\lambda U_{max}}}, \quad x \in [0, U_{max}] \quad (7)$$

Similar to Eq. (4), we have the probability density function:

$$\begin{aligned} f^*(x) &= p[F(x)]^{p-1} \cdot f(x) = p \cdot \left(\frac{1 - e^{-\lambda x}}{1 - e^{-\lambda U_{max}}}\right)^{p-1} \cdot \lambda e^{-\lambda x} \\ &= \frac{p\lambda e^{-\lambda x}}{(1 - e^{-\lambda U_{max}})^{p-1}} \sum_{k=0}^{p-1} \binom{p-1}{k} (-e^{-\lambda x})^k \\ &= \frac{1}{(1 - e^{-\lambda U_{max}})^{p-1}} \sum_{k=0}^{p-1} (-1)^k p \binom{p-1}{k} \lambda (e^{-\lambda x})^{k+1} \end{aligned} \quad (8)$$

By substituting $p\binom{p-1}{k} = (k+1)\binom{p}{k+1}$ into Eq. (8), we have

$$\begin{aligned} f^*(x) &= \frac{1}{(1 - e^{-\lambda U_{max}})^{p-1}} \sum_{k=0}^{p-1} (-1)^k (k+1) \binom{p}{k+1} \lambda (e^{-\lambda x})^{k+1} \\ &= -\frac{1}{(1 - e^{-\lambda U_{max}})^{p-1}} \sum_{k=1}^{p} (-1)^k \binom{p}{k} k\lambda (e^{-k\lambda x}) \end{aligned} \quad (9)$$

Similar to Eq. (5), the expectation of X^* (*i.e.*, $\mathbb{E}[X^*]$) is calculated as follows.

$$\begin{aligned} &- \frac{1}{(1 - e^{-\lambda U_{max}})^{p-1}} \sum_{k=1}^{p} (-1)^k \binom{p}{k} \int_0^{U_{max}} k\lambda x (e^{-k\lambda x})\, dx \\ &= -\frac{1}{(1 - e^{-\lambda U_{max}})^{p-1}} \sum_{k=1}^{p} (-1)^k \binom{p}{k} \frac{1}{k\lambda} \end{aligned} \quad (10)$$

Since the p-th Harmonic number [47] (denoted by H_p) can be calculated as $H_p = -\sum_{k=1}^{p} (-1)^k \binom{p}{k} \frac{1}{k}$, we have

$$\mathbb{E}[X^*] = \frac{H_p}{\lambda(1 - e^{-\lambda U_{max}})^{p-1}} \geq \frac{1}{\lambda} \quad (11)$$

Based on the same reason of Eq. (6), we can bound the expected total utility of Greedy as follows:

$$\mathbb{E}[\text{MaxSum}(M)] \geq (1 - \frac{1}{e})\mathbb{E}[|OPT|] \cdot \frac{1}{\lambda} \quad (12)$$

Since the expected total utility is at most $\mathbb{E}[|OPT|] \cdot U_{max}$, we have

$$CR_{RO} = \frac{\mathbb{E}[\text{MaxSum}(M)]}{\mathbb{E}[\text{MaxSum}(OPT)]} \geq (1 - \frac{1}{e}) \frac{1}{\lambda U_{max}}$$

□

TABLE 3: Real datasets.

Platform	$\|T\|$	$\|W\|$	c_w	δ_w	r_w	p_t	due
gMission	713	532	1,2,3,4,5	0.8	1	10.45	5
EverySender	4036	817	1,2,5,10,20	0.6	1	5.24	10

TABLE 4: Synthetic datasets.

Parameter	Setting
$\|T\|$	500, 1000, **2500**, 5000, 10000
$\|W\|$	100, 200, **500**, 1000, 5000
c_w	**1**, 2, 3, 4, 5
δ_w	0.1, 0.3, **0.5**, 0.7, 0.9
r_w	1.0, 1.5, **2.0**, 2.5, 3.0
p_t $(\mu, mean, 1/\lambda)$	2, 5, **10**, 15, 20
due	2, 4, **6**, 8, 10
scalability $(\|T\| \times \|W\|)$	10K × 1K, 20K × 2K, 30K × 3K, 40K × 4K, 50K × 5K, 60K × 6K, ⋯, 100K × 10K

6.2.3 Analysis Under Normal Distribution

Similar to the analysis in Sec. 6.2.1, we assume that each random variable X_i follows a normal distribution with parameter μ and σ. Then we have the following conclusion on the competitive ratio of Greedy.

Theorem 7. *If the utilities follow the normal distribution (with parameter μ and σ), the competitive ratio of Greedy is $(1 - \frac{1}{e})\frac{\mu}{U_{max}}$ under the random order model.*

For example, if we let μ equals to $0.5U_{max}$, the ratio is asymptotic to $(1 - \frac{1}{e}) \cdot \frac{1}{2} \sim 0.32$.

Proof. Under the normal distribution with parameter μ and σ, [48] gives the lower bound of $\mathbb{E}[X^*]$ as follows.

$$\mathbb{E}[x^*] \geq \mu + 0.23\sigma\sqrt{\log p} \geq \mu \quad (13)$$

Since [43] proves that $\mathbb{E}[|M|] \geq (1 - \frac{1}{e})\mathbb{E}[|OPT|]$ under random order model, we can bound the expected total utility of Greedy as follows

$$\mathbb{E}[\text{MaxSum}(M)] \geq (1 - \frac{1}{e})\mathbb{E}[|OPT|] \cdot \mu \quad (14)$$

Since the expected total utility is at most $\mathbb{E}[|OPT|] \cdot U_{max}$, we have

$$CR_{RO} = \frac{\mathbb{E}[\text{MaxSum}(M)]}{\mathbb{E}[\text{MaxSum}(OPT)]} \geq (1 - \frac{1}{e})\frac{\mu}{U_{max}}$$

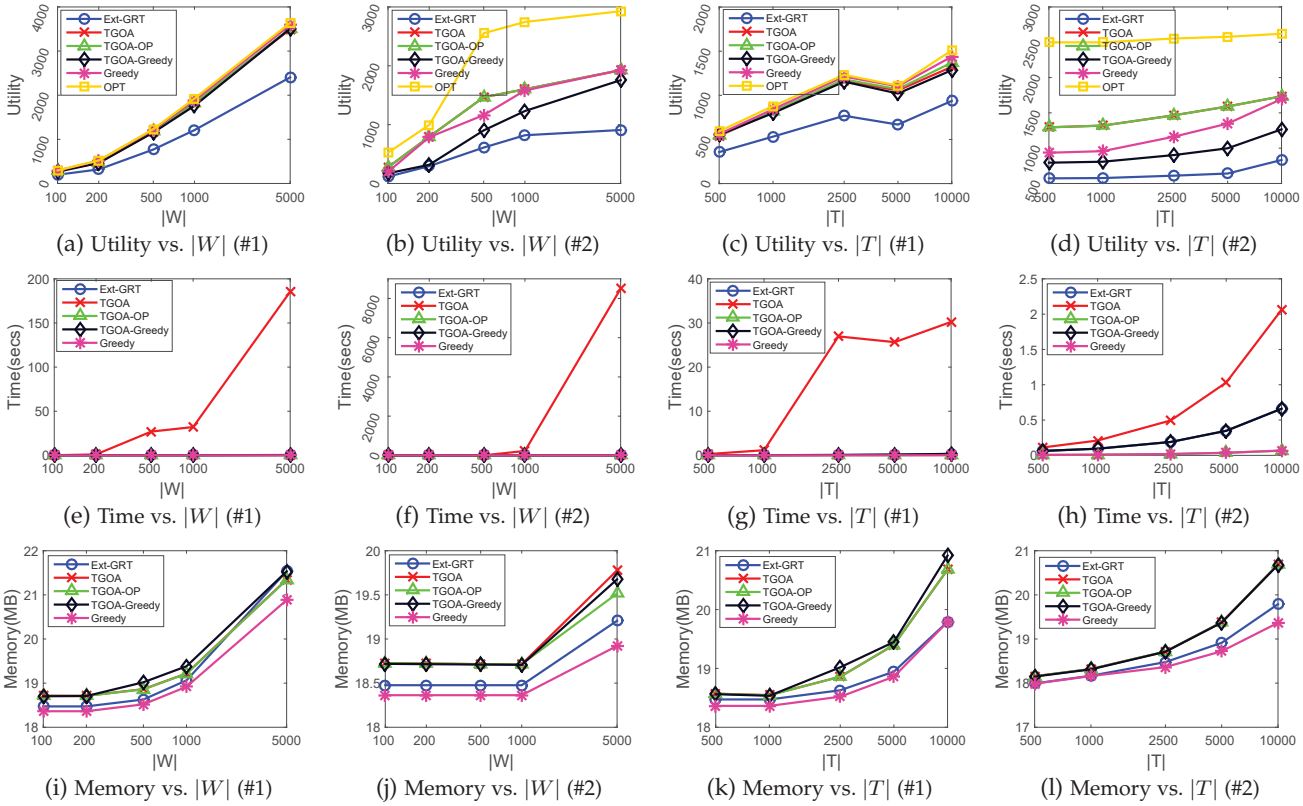
□

Note that the competitive ratio of Greedy under random order model is not arbitrarily bad. However, compared with TGOA-based algorithms (*i.e.*, TGOA, TGOA-Greedy and TGOA-OP), its competitive ratio is sensitive to the distribution of datasets. For example, under exponential and normal distributions, the competitive ratio of Greedy is affected by the mean of the utilities. Specifically, only when the mean is larger than $0.4U_{max}$, the ratio of Greedy is better than TGOA and TGOA-OP. Though the competitive ratio of Greedy is better than the TGOA-based algorithms under uniform distribution, the analysis of TGOA-based algorithms does not rely on the uniform distribution. Overall, *the performance of Greedy is not arbitrarily bad on average* and may be less stable than TGOA and TGOA-OP in experiments.

7 Experimental Study

This section presents the experimental evaluations of our proposed algorithms.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2948863, IEEE Transactions on Knowledge and Data Engineering



(a) Utility vs. $|W|$ (#1) (b) Utility vs. $|W|$ (#2) (c) Utility vs. $|T|$ (#1) (d) Utility vs. $|T|$ (#2)

(e) Time vs. $|W|$ (#1) (f) Time vs. $|W|$ (#2) (g) Time vs. $|T|$ (#1) (h) Time vs. $|T|$ (#2)

(i) Memory vs. $|W|$ (#1) (j) Memory vs. $|W|$ (#2) (k) Memory vs. $|T|$ (#1) (l) Memory vs. $|T|$ (#2)

Fig. 2: Results on varying $|W|$ and $|T|$ on synthetic datasets.

7.1 Experimental Setup

Datasets. We use two real datasets, the gMission dataset [49] and the EverySender dataset [50]. 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 due time (minute) and payoff. Each worker is also associated with a location, an available time, a due time (minute), the maximum activity range (km), 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.*, collecting packages, or conduct tasks as a worker. Similar to the gMission dataset, each task and worker in the EverySender dataset also includes its corresponding information. Since the capacity of workers (*i.e.*, c_w) is not given in the datasets, we generate the capacity of workers. TABLE 3 presents the statistics of the real datasets.

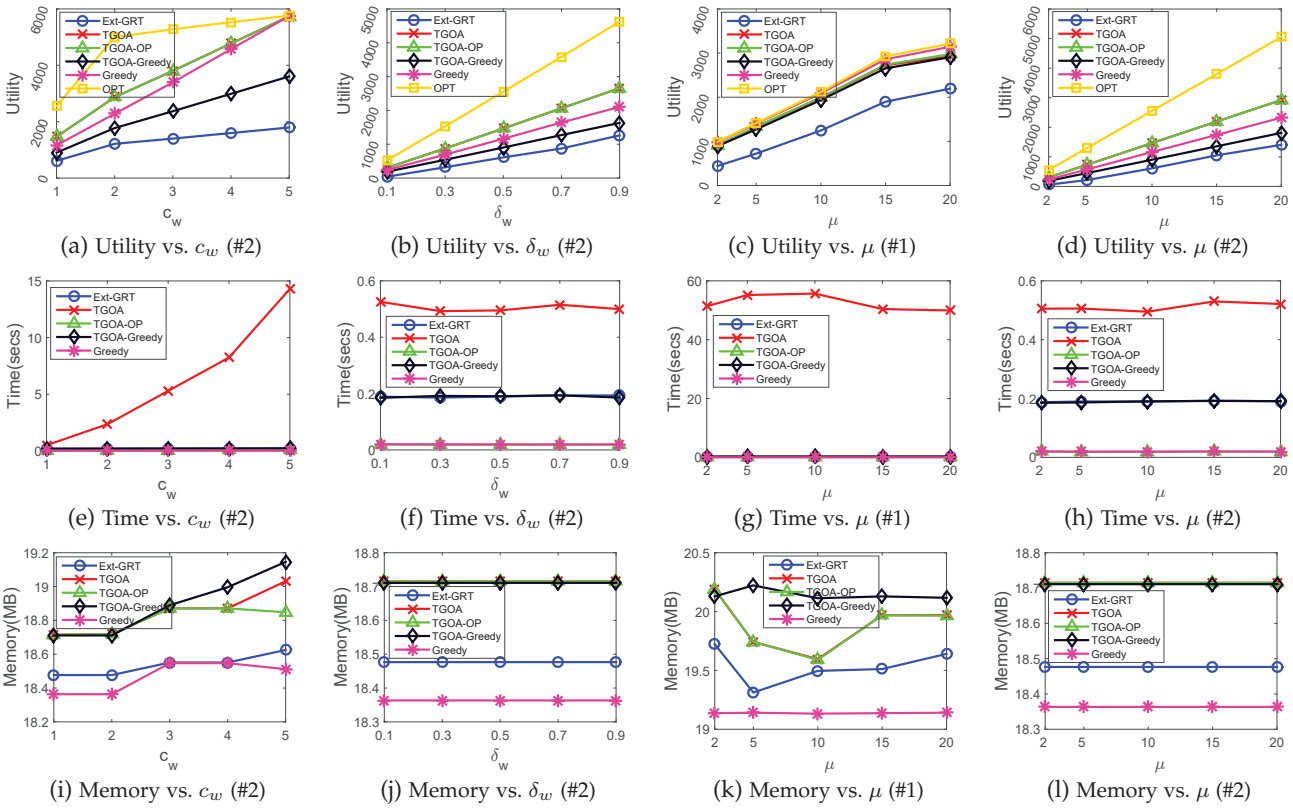
We also use synthetic datasets for evaluation. The parameter settings of the experiments are shown in TABLE 4, where the default parameters are marked in bold. To simulate the cases when utilities follow certain distribution, we generate the values of p_t following normal, uniform and exponential distributions. Specifically, we vary the parameter $\mu, mean, 1/\lambda$ in the aforementioned distributions, respectively. For normal distribution, we set parameter σ as 3.75 and only vary the value of μ, because μ tends to have more influence on the performance (based on the competitive analysis in Theorem 7). The deadline is calculated as the arrival time of a task/worker added by the parameter "due" in the table. We use two different ways to generate the locations of tasks and workers within a square of 100×100 in the 2D coordinates. In one way, we totally generate the locations in a random way as in our preliminary version [15]. In the other way, we first randomly generate the locations of workers and then dependently generate the locations of tasks, *i.e.*, we randomly sample the locations within the circular regions of workers. For simplicity, we use *Syn#1* and *Syn#2* to denote the synthetic datasets generated by these two ways. The main difference is that a worker can perform fewer tasks in *Syn#1* than *Syn#2* due to the range constraint. Furthermore, the arrival order of all tasks and workers follows uniform distribution following the random order model.

Metrics and Implementation. We evaluate the Extended Greedy-RT (denoted by "Ext-GRT"), TGOA, TGOA-Greedy, TGOA-OP and Greedy algorithms in terms of total utility score (utility for short), total running time (time for short) and memory cost (memory for short). We also present the results of offline optimal solution (denoted by "OPT"). In each experiment, we repeatedly test 100 different online arrival orders of tasks and workers and report the average results. All the algorithms are implemented in GNU C++, and the experiments were performed on a server with Intel(R) Xeon(R) X5675 3.07GHz CPU and 128GB main memory.

7.2 Experiment Results

We first present the results on two synthetic datasets with various parameters, and then show the performance on two real datasets. Due to the space limit, some of the experimental results are shown in our supplemental materials [22], including the results of varying capacity c_w and success ratio δ_w on *Syn#1*, varying radius r_w and response deadline due on *Syn#1* and *Syn#2*.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2948863, IEEE Transactions on Knowledge and Data Engineering



(a) Utility vs. c_w (#2) (b) Utility vs. δ_w (#2) (c) Utility vs. μ (#1) (d) Utility vs. μ (#2)

(e) Time vs. c_w (#2) (f) Time vs. δ_w (#2) (g) Time vs. μ (#1) (h) Time vs. μ (#2)

(i) Memory vs. c_w (#2) (j) Memory vs. δ_w (#2) (k) Memory vs. μ (#1) (l) Memory vs. μ (#2)

Fig. 3: Results on varying capacity c_w, success ratio δ_w and normal distribution parameter μ on synthetic datasets.

Impact of cardinality of W. The results of varying $|W|$ are shown in the first two columns of Fig. 2. The utility increases as $|W|$ increases, since there are more matched edges as $|W|$ increases. TGOA, TGOA-Greedy and TGOA-OP return better matching results than the baseline Extended Greedy-RT does, which yield up to 169.58%, 93.21% and 170.13% more utility than the baseline. The utility of Greedy is up to 167.07% larger than the baseline Extended Greedy-RT. Greedy outperforms TGOA and TGOA-Greedy on *Syn#1* while Greedy is less effective than them on *Syn*#2. Since there are more tasks around a worker on *Syn*#2 with different payoff, Greedy may match the task which arrives first yet with less payoff. However, the allocations by TGOA and TGOA-OP are relatively globally optimal. TGOA-Greedy returns slightly worse utility than the other proposed algorithms. The time and memory costs of all the algorithms increase when $|W|$ increases. Greedy is the most efficient and TGOA is the least efficient, which is aligned with their complexity analysis. TGOA-Greedy and TGOA-OP are more efficient than TGOA in both time and memory costs.

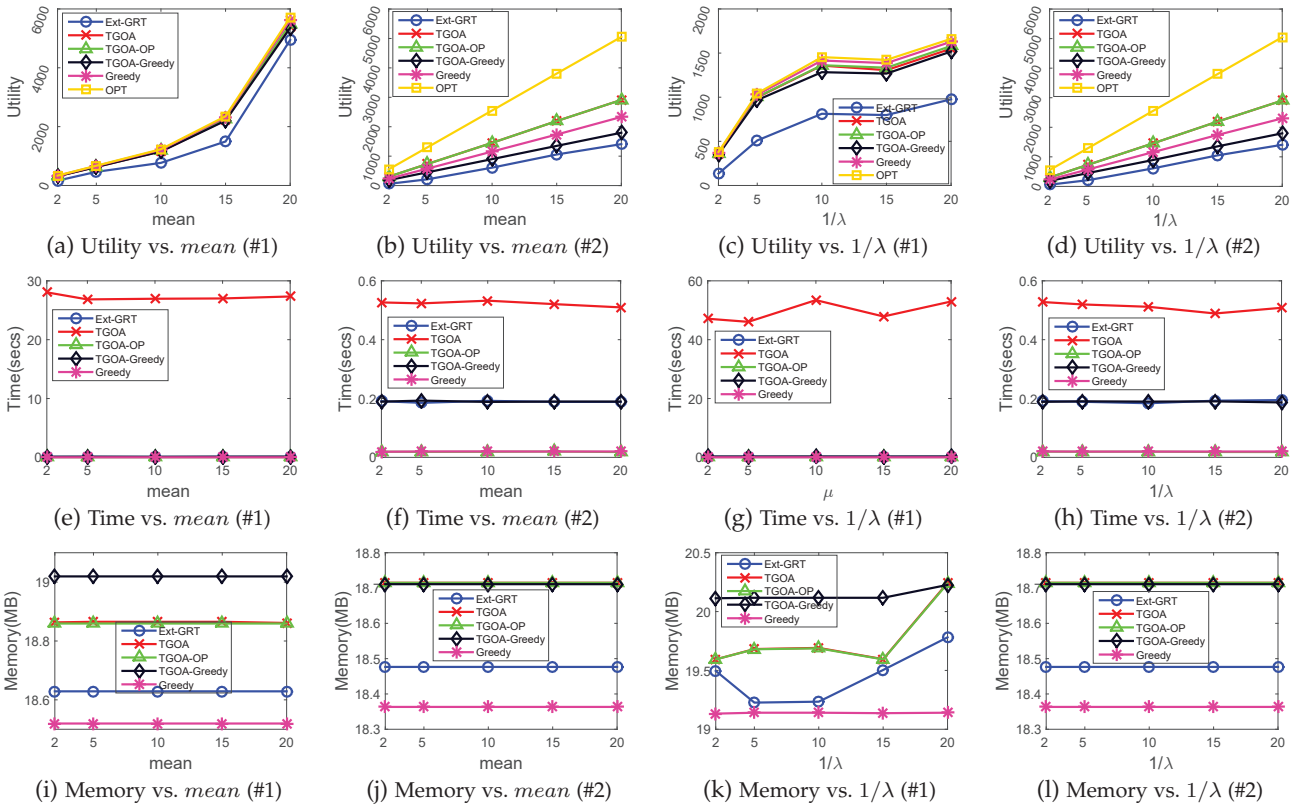
Impact of cardinality of T. The results of varying $|T|$ are shown in the last two columns of Fig. 2. The results are similar to those when varying the cardinality of W. Again, all the proposed algorithms are better than Extended Greedy-RT. On *Syn#1*, Greedy performs the best, with up to 58.31% more utility than the baseline, followed by TGOA-OP, TGOA and TGOA-Greedy. On *Syn*#2, TGOA-OP and TGOA perform the best, with 148.20% higher utility in average than the baseline, followed by Greedy and TGOA-Greedy. TGOA-Greedy is still better than the baseline, with 49.01% and 46.9% higher utility in average on *Syn#1* and *Syn*#2. In terms of time and memory costs, Greedy is still the most efficient and the TGOA is the least efficient. TGOA-Greedy and TGOA-OP are as efficient as Extended Greedy-RT, *i.e.*, the differences in time and space consumptions are less than 40ms and 1.2MB in average.

Impact of capacity c_w on *Syn*#2. The results of varying c_w are presented in the first column of Fig. 3. The total utility of all algorithms increases when c_w increases, since a worker can gradually perform more tasks. When the capacity c_w is 5, TGOA, TGOA-OP and Greedy tend to obtain near optimal utility. Because there are $2500/500 = 5$ tasks around each worker in average, the workers with capacity 5 tend to perform the most of tasks. Compared with the baseline, our proposed algorithms are much better, by at least 50.55% higher utility in average. As for the time and memory costs, we can observe similar patterns to other experiment results.

Impact of success ratio δ_w on *Syn*#2. The results of varying the success ratio of workers are presented in the second column of Fig. 3. The utility of all the algorithms increases with the increase of success ratio of workers. TGOA and TGOA-OP are the most effective algorithms, followed by Greedy, TGOA-Greedy and Extended Greedy-RT. As for running time, TGOA-OP and Greedy are the most efficient, followed by TGOA-Greedy and Extended Greedy-RT. TGOA is the least efficient. As for memory cost, all the algorithms consume only a little space (*i.e.*, less than 19MB).

Impact of parameter μ of normal distribution. The results of varying the utilities under normal distribution are shown in the last two columns of Fig. 3. In terms of utility, Greedy is the most effective in *Syn#1* and is beaten by TGOA and TGOA-OP on *Syn*#2. All the proposed algorithms are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2948863, IEEE Transactions on Knowledge and Data Engineering



(a) Utility vs. $mean$ (#1) (b) Utility vs. $mean$ (#2) (c) Utility vs. $1/\lambda$ (#1) (d) Utility vs. $1/\lambda$ (#2)

(e) Time vs. $mean$ (#1) (f) Time vs. $mean$ (#2) (g) Time vs. $1/\lambda$ (#1) (h) Time vs. $1/\lambda$ (#2)

(i) Memory vs. $mean$ (#1) (j) Memory vs. $mean$ (#2) (k) Memory vs. $1/\lambda$ (#1) (l) Memory vs. $1/\lambda$ (#2)

Fig. 4: Results on varying uniform distribution parameter $mean$ and exponential distribution parameter $1/\lambda$ on synthetic datasets.

still more effective than Extended Greedy-RT, with at least 61.14% and 61.38% higher utility in average on the two datasets. In terms of time and memory costs, Greedy is still the most efficient and TGOA is the least efficient.

Impact of parameter $mean$ of uniform distribution. The results of varying the utilities under the uniform distribution are in the first two columns of Fig. 4. The trend of the utility when varying $mean$ of uniform distribution is similar to that in the normal distribution. On *Syn#1*, TGOA-based algorithms and Greedy are close to the optimal results. On *Syn*#2, TGOA and TGOA-OP have a notable improvement over the other algorithms. Overall, all the proposed algorithms are more effective than the baseline. As for the time and memory costs, the patterns are similar to other experiment results.

Impact of parameter $1/\lambda$ of exponential distribution. The results of varying the utilities under exponential distribution are shown in the last two columns of Fig. 4. In terms of utility, the TGOA-based algorithms and Greedy are more effective than the baseline. The ranks of all the algorithms are similar to previous experiment results (*e.g.*, Fig. 3c and Fig. 3d). As for the time and memory costs, all the proposed algorithms are efficient except that TGOA takes more running time than others.

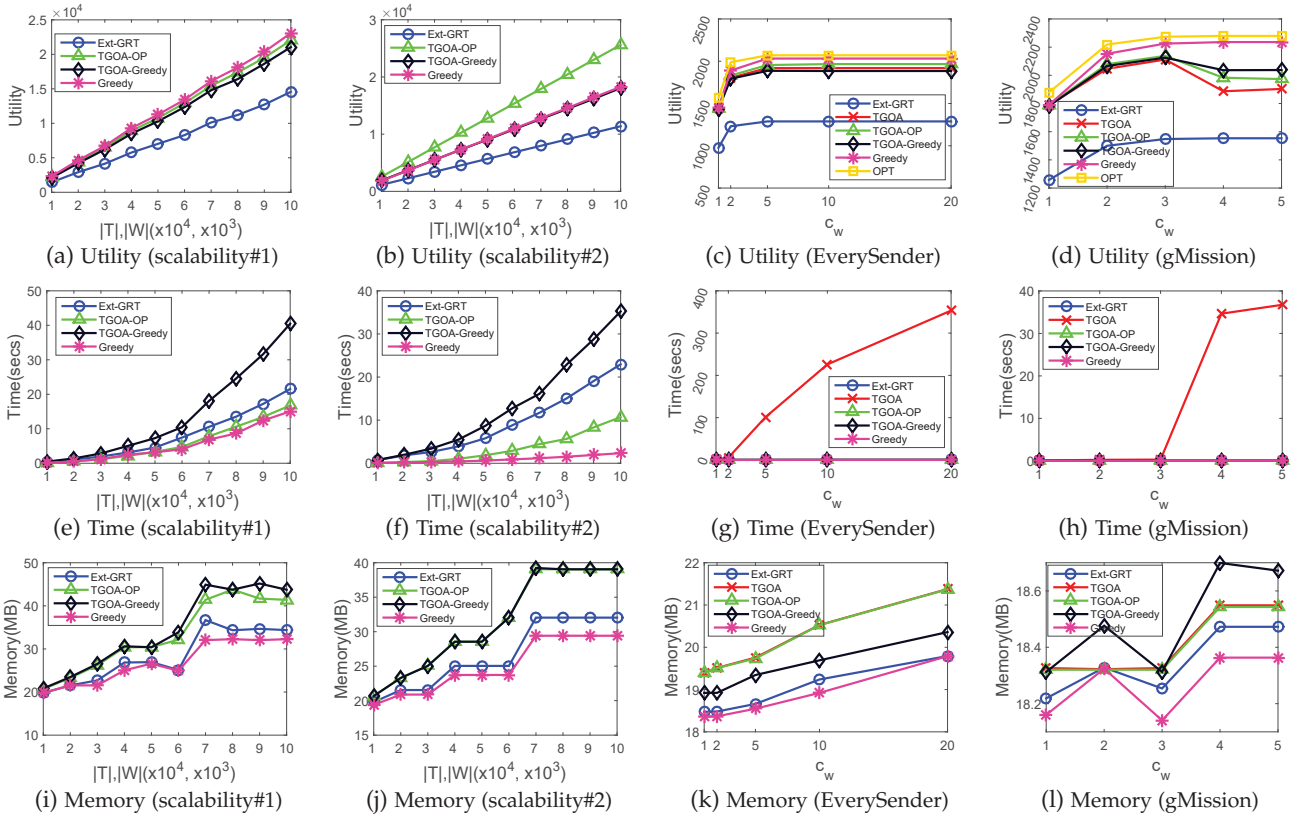
Scalability test. Since TGOA is not efficient enough according to our previous experiment results, we omit its experimental results. The results of scalability test are shown in the first two columns in Fig. 5. In terms of utility, all the proposed algorithms outperform the baseline Extended Greedy-RT. For example, on *Syn*#2, TGOA-OP, TGOA-Greedy and Greedy are up to 2.25x, 1.59x and 1.61x more effective than the baseline Extended Greedy-RT. As for the running time on *Syn#1* and *Syn*#2, Greedy is the fastest, by up to 10.06x faster than baseline. TGOA-OP is the second fastest, by up to 9.35x faster than baseline. Even though TGOA-Greedy is slower than the baseline, it is still efficient because each requests can be responded within 10ms in average. As for the memory cost, all the algorithms are efficient (*i.e.*, less than 50MB).

Performance on real datasets. The last two columns of Fig. 5 show the results on real datasets. On EverySender dataset, the increase of utility tends to be stable when c_w becomes greater than 5, because the total number of tasks is closed to five times of total number of workers. On gMission dataset, the utility still increases for Greedy and Extended Greedy-RT but drops a little bit for the TGOA-based algorithms when c_w increases from 3 to 4. Since the total capacity of workers first becomes much larger than the total number of tasks (when c_w is 4), some of the workers may not perform as many tasks as possible due to globally optimal allocation of second phase in TGOA-based algorithms. The total utility becomes stable when c_w is larger than 4. Our proposed algorithms obtain at least 31.82% higher utility than Extended Greedy-RT. In terms of time and memory costs, Greedy is the most efficient, and TGOA-Greedy and TGOA-OP are as efficient as the baseline.

Summary of Results. We summarize our experimental findings as follows.

- TGOA is up to 169.58% more effective than the baseline Extended Greedy-RT at the cost of more time and space.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2948863, IEEE Transactions on Knowledge and Data Engineering



(a) Utility (scalability#1) (b) Utility (scalability#2) (c) Utility (EverySender) (d) Utility (gMission)

(e) Time (scalability#1) (f) Time (scalability#2) (g) Time (EverySender) (h) Time (gMission)

(i) Memory (scalability#1) (j) Memory (scalability#2) (k) Memory (EverySender) (l) Memory (gMission)

Fig. 5: Results on scalability tests on synthetic datasets and the performances on real datasets.

- TGOA-Greedy is less effective than TGOA while still up to 93.21% more effective than the baseline. It significantly outperforms TGOA in terms of running time and memory cost.
- TGOA-OP inherits the advantages of both TGOA and TGOA-Greedy, which is up to 170.13% more effective than baseline with low time and memory costs (*e.g.*, up to 9.35x faster than baseline in scalability test).
- Greedy is the most efficient algorithm with up to 167.07% higher utility than Extended Greedy-RT. However, its effectiveness is sensitive to the distributions of datasets, *e.g.*, it is most effective on *Syn#1* but is less effective than TGOA and TGOA-OP on *Syn#2*.
- The results on real datasets show that though state-of-the-art algorithm (*i.e.*, Extended Greedy-RT) has better competitive ratio under adversarial order model, it does not perform well in practice.

8 Conclusion

In this paper, we first identify a new online micro-task allocation problem, called *Global Online Micro-task Allocation in spatial crowdsourcing* (GOMA), which is more general than the Online Maximum Weighted Bipartite Matching (OMWBM) problem. Then, we extend the state-of-the-art Greedy-RT algorithm to the OMWBM problem as our baseline. Although the baseline has nearly optimal guarantee under the adversarial order model, it does not perform well enough in practice. Thus, we propose a two-phase-based framework under the random order model, which is more suitable to reflect the average performance. Based on this framework, we present the TGOA algorithm with competitive ratio of $\frac{1}{4}$, but is not scalable to large datasets. To improve the scalability, we further design TGOA-Greedy and TGOA-OP, which runs faster with the competitive ratio of $\frac{1}{8}$ and $\frac{1}{4}$. Finally, we analyze the average performance of Greedy algorithm, which is considered as the ineffective algorithm due to its unbounded competitive ratio under the adversarial order model. We conduct extensive experiments which verify the effectiveness, efficiency and scalability of the proposed approaches. Based on our experimental results, the Greedy algorithm is more effective than the state-of-the-art but is sensitive to the distribution of the datasets. TGOA-OP is as effective as TGOA but is more efficient. TGOA-OP is also more stable than Greedy.

References

[1] L. Kazemi and C. Shahabi, "Geocrowd: enabling query answering with spatial crowdsourcing," in *GIS*, 2012.
[2] "Gigwalk," *https://gigwalk.com/*.
[3] "TaskRabbit," *https://taskrabbit.com/*.
[4] "Waze," *https://www.waze.com/*.
[5] Y. Tong, L. Chen, and C. Shahabi, "Spatial crowdsourcing: Challenges, techniques, and applications," *PVLDB*, 2017.
[6] Y. Tong and Z. Zhou, "Dynamic task assignment in spatial crowdsourcing," *SIGSPATIAL Special*, 2018.
[7] L. Chen and C. Shahabi, "Spatial crowdsourcing: Challenges and opportunities," *IEEE Data Eng. Bull.*, 2016.
[8] R. E. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems, Revised Reprint*, 2009.
[9] H. To, C. Shahabi, and L. Kazemi, "A server-assigned spatial crowdsourcing framework," *ACM Trans. Spatial Algorithms and Systems*, 2015.
[10] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, "An optimal algorithm for on-line bipartite matching," in *STOC*, 1990.
[11] G. Aggarwal, G. Goel, C. Karande, and A. Mehta, "Online vertex-weighted bipartite matching and single-bid budgeted allocations," in *SODA*, 2011.

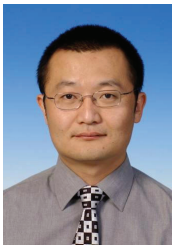
[12] Z. Huang, Z. G. Tang, X. Wu, and Y. Zhang, "Online vertex-weighted bipartite matching: Beating 1-1/e with random arrivals," in *ICALP*, 2018.
[13] N. Korula and M. Pál, "Algorithms for secretary problems on graphs and hypergraphs," in *ICALP*, 2009.
[14] H. Ting and X. Xiang, "Near optimal algorithms for online maximum edge-weighted b-matching and two-sided vertex-weighted b-matching," *Theor. Comput. Sci.*, 2015.
[15] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen, "Online mobile micro-task allocation in spatial crowdsourcing," in *ICDE*, 2016.
[16] Y. Tong, L. Wang, Z. Zhou, B. Ding, L. Chen, J. Ye, and K. Xu, "Flexible online task assignment in real-time spatial data," *PVLDB*, 2017.
[17] D. Deng, C. Shahabi, and U. Demiryurek, "Maximizing the number of worker's self-selected tasks in spatial crowdsourcing," in *GIS*, 2013.
[18] L. Zhang, T. Hu, Y. Min, G. Wu, J. Zhang, P. Feng, P. Gong, and J. Ye, "A taxi order dispatch model based on combinatorial optimization," in *KDD*, 2017.
[19] J. P. Dickerson, K. A. Sankararaman, A. Srinivasan, and P. Xu, "Allocation problems in ride-sharing platforms: Online matching with offline reusable resources," in *AAAI*, 2018.
[20] B. Zhao, P. Xu, Y. Shi, Y. Tong, Z. Zhou, and Y. Zeng, "Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach," in *AAAI*, 2019.
[21] D. Gao, Y. Tong, J. She, T. Song, L. Chen, and K. Xu, "Top-k team recommendation and its variants in spatial crowdsourcing," *Data Science and Engineering*, 2017.
[22] "Full paper," *http://home.cse.ust.hk/~yzengal/goma.pdf*.
[23] A. Mehta, "Online matching and ad allocation," *Foundations and Trends in Theoretical Computer Science*, 2013.
[24] H. Hu, G. Li, Z. Bao, J. Feng, Y. Wu, Z. Gong, and Y. Xu, "Top-k spatio-textual similarity join," *IEEE Trans. Knowl. Data Eng.*, 2016.
[25] N. Ta, G. Li, Y. Xie, C. Li, S. Hao, and J. Feng, "Signature-based trajectory similarity join," *IEEE Trans. Knowl. Data Eng.*, 2017.
[26] Y. Tong, L. Chen, Z. Zhou, H. V. Jagadish, L. Shou, and W. Lv, "SLADE: A smart large-scale task decomposer in crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, 2018.
[27] J. Fan, G. Li, B. C. Ooi, K.-l. Tan, and J. Feng, "iCrowd: An adaptive crowdsourcing framework," in *SIGMOD*, 2015.
[28] G. Li, C. Chai, J. Fan, X. Weng, J. Li, Y. Zheng, Y. Li, X. Yu, X. Zhang, and H. Yuan, "CDB: A crowd-powered database system," *PVLDB*, 2018.
[29] G. Li, J. Wang, Y. Zheng, and M. J. Franklin, "Crowdsourced data management: A survey," *IEEE Trans. Knowl. Data Eng.*, 2016.
[30] H. Hu, Y. Zheng, Z. Bao, G. Li, J. Feng, and R. Cheng, "Crowdsourced POI labelling: Location-aware result inference and task assignment," in *ICDE*, 2016.
[31] S. R. B. Gummidi, X. Xie, and T. B. Pedersen, "A survey of spatial crowdsourcing," *ACM Trans. Database Syst.*, 2019.
[32] Y. Zeng, Y. Tong, L. Chen, and Z. Zhou, "Latency-oriented task completion via spatial crowdsourcing," in *ICDE*, 2018.
[33] H. To, C. Shahabi, and L. Xiong, "Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server," in *ICDE*, 2018.
[34] Y. Tong, J. She, B. Ding, L. Chen, T. Wo, and K. Xu, "Online minimum matching in real-time spatial data: Experiments and analysis," *PVLDB*, 2016.
[35] Y. Tong, L. Wang, Z. Zhou, L. Chen, B. Du, and J. Ye, "Dynamic pricing in spatial crowdsourcing: A matching-based approach," in *SIGMOD*, 2018.
[36] Q. Tao, Y. Zeng, Z. Zhou, Y. Tong, L. Chen, and K. Xu, "Multi-worker-aware task planning in real-time spatial crowdsourcing," in *DASFAA*, 2018.
[37] Y. Tong, Y. Zeng, Z. Zhou, L. Chen, J. Ye, and K. Xu, "A unified approach to route planning for shared mobility," *PVLDB*, 2018.
[38] N. Ta, G. Li, T. Zhao, J. Feng, H. Ma, and Z. Gong, "An efficient ride-sharing framework for maximizing shared route," *IEEE Trans. Knowl. Data Eng.*, 2018.
[39] U. ul Hassan and E. Curry, "A multi-armed bandit approach to online spatial task assignment," in *UIC*, 2014.
[40] J. P. Dickerson, K. A. Sankararaman, A. Srinivasan, and P. Xu, "Assigning tasks to workers based on historical data: Online task assignment with two-sided arrivals," in *AAMAS*, 2018.
[41] Y. Li, J. Fang, Y. Zeng, B. Maag, Y. Tong, and L. Zhang, "Two-sided online bipartite matching in spatial data: Experiments and analysis," *GeoInformatica*, 2019.
[42] Y. Wang and S. C.-w. Wong, "Two-sided online bipartite matching and vertex cover: Beating the greedy algorithm," in *ICALP*, 2015.
[43] G. Goel and A. Mehta, "Online budgeted matching in random input models with applications to adwords," in *SODA*, 2008.
[44] Z. Huang, N. Kang, Z. G. Tang, X. Wu, Y. Zhang, and X. Zhu, "How to match when all vertices arrive online," in *STOC*, 2018.
[45] L. Kazemi, C. Shahabi, and L. Chen, "Geotrucrowd: trustworthy query answering with spatial crowdsourcing," in *GIS*, 2013.
[46] T. Song, Y. Tong, L. Wang, J. She, B. Yao, L. Chen, and K. Xu, "Trichromatic online matching in real-time spatial crowdsourcing," in *ICDE*, 2017.
[47] "Harmonic number," *https://en.wikipedia.org/wiki/Harmonic_number*.
[48] G. Kamath, "Bounds on the expectation of the maximum of samples from a gaussian," *http://www.gautamkamath.com/writings/gaussian_max.pdf*, 2015.
[49] "gMission," *http://gmission.github.io/*.
[50] "EverySender," *http://www.dajiasong.com/*.

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 an associate professor in the School of Computer Science and Engineering, Beihang University. His research interests include crowdsourcing, big spatio-temporal data processing, uncertain data mining and management and social network analysis. He is a member of the IEEE.

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.

Bolin Ding received the Ph.D. degree in Computer Science at University of Illinois at Urbana-Champaign, USA, in 2012. He is currently a researcher in the Data Analytics and Intelligence Lab (DAIL) at Alibaba DAMO Academy. His research interests center on large-scale data management and analytics, including interactively querying and mining "big" data, privacy-preserving data analytics, and optimizations in databases and machine learning.

Libin Wang is currently a Master candidate in the School of Computer Science and Engineering, Beihang University. His major research interests are crowdsourcing and spatio-temporal data management.



Lei Chen received the B.S. degree in computer science and engineering from Tianjin University, China, in 1994, the M.A. degree from the Asian Institute of Technology, Thailand, in 1997, and the Ph.D. degree in computer science from the University of Waterloo, Canada, in 2005. He is now a professor at the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His current research interests include crowdsourcing-based data processing and data-driven machine learning. He is a member of the IEEE.