Geoinformatica
https://doi.org/10.1007/s10707-019-00351-4

Multi-skill aware task assignment in real-time spatial crowdsourcing

Tianshu Song[1] · Ke Xu[1] · Jiangneng Li[1] · Yiming Li[1] · Yongxin Tong[1]

Received: 23 November 2018 / Revised: 7 March 2019 / Accepted: 17 March 2019 /
Published online: 04 April 2019

Abstract
With the development of mobile Internet and the prevalence of sharing economy, spatial crowdsourcing (SC) is becoming more and more popular and attracts attention from both academia and industry. A fundamental issue in SC is assigning tasks to suitable workers to obtain different global objectives. Existing works often assume that the tasks in SC are micro and can be completed by any single worker. However, there also exist macro tasks which need a group of workers with different kinds of skills to complete collaboratively. Although there have been a few works on macro task assignment, they neglect the dynamics of SC and assume that the information of the tasks and workers can be known in advance. This is not practical as in reality tasks and workers appear dynamically and task assignment should be performed in real time according to partial information. In this paper, we study the multi-skill aware task assignment problem in real-time SC, whose offline version is proven to be NP-hard. To solve the problem effectively, we first propose the Online-Exact algorithm, which always computes the optimal assignment for the newly appearing tasks or workers. Because of Online-Exact's high time complexity which may limit its feasibility in real time, we propose the Online-Greedy algorithm, which iteratively tries to assign workers who can cover more skills with less cost to a task until the task can be completed. We finally demonstrate the effectiveness and efficiency of our solutions via experiments conducted on both synthetic and real datasets.

Keywords Spatial crowdsourcing · Real-time · Task assignment · Multi-skill

1 Introduction

The development of mobile Internet breeds rich location-based applications and related research [5, 8, 11, 14, 15, 18]. In recent years, the combination of sharing economy

Grants or other notes about the article that should go on the front page should be placed here.
General acknowledgments should be placed at the end of the article.

✉ Tianshu Song
songts@buaa.edu.cn

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

and mobile Internet is initiating many kinds of spatial crowdsourcing platforms such as Gigwalk (http://www.gigwalk.com), TaskRabbit (http://www.taskrabbit.com) and gMission [30], where requesters of tasks and workers log in and off dynamically at different time and places. A fundamental research topic on spatial crowdsourcing is task assignment [21, 24], *i.e.*, assigning suitable workers to nearby tasks with different optimization objectives, such as maximizing the total utility [27], maximizing the number of completed tasks [26] and minimizing the total moving distance of the workers [29].

Existing works on task assignment mainly assume that all the tasks are *atomic and homogeneous*, which can be completed by*any single worker*, such as picking up and delivering things [31].

However, like the research in the web-based crowdsourcing [22], there are also some tasks in spatial crowdsourcing which are *complex and structural* and require *a group of workers* with different kinds of skills to work collaboratively. For example, a requester on gMission can publish a task whose content is to organize a party. Then, workers who are good at preparing the food, the drinks and the lights may be needed. Or, if the task is yard work, workers who can mow lawns, repair the fence and take care of the flowers are necessary.

Although there are some studies on assigning workers for complex tasks such as [23, 28], they only consider the static scenario. In other words, they assume that the information on the tasks and workers is known in advance, based on which task assignment is performed. However, in practice, the tasks and workers log in and out of the spatial crowdsourcing platform dynamically and we always perform task assignment based on the information released currently. In this paper, we study the Online Multi-skill-Aware Task Assignment problem (Fig. 1).

We next use an example to demonstrate the above motivation.

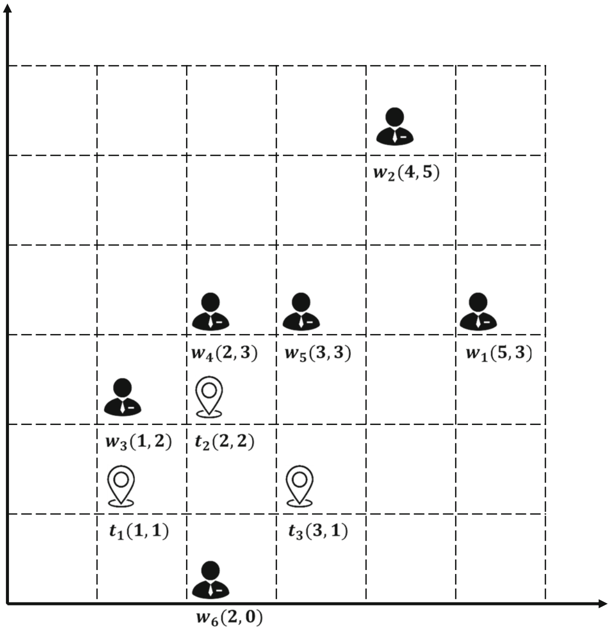


Fig. 1 Locations of Tasks and Workers

Table 1 Information of tasks

Tasks	Lists of required skills	Location	Budget
t_1	s_1 (Mix drinks), s_2 (Prepare food), s_3 (Prepare barbecue), s_4 (Decorate venues) , s_5 (Take photos)	(1, 1)	50
t_2	s_1 (Mix drinks), s_2 (Prepare food), s_3 (Prepare barbecue), s_4 (Decorate venues) , s_5 (Take photos)	(2, 2)	50
t_3	s_1 (Mix drinks), s_2 (Prepare food), s_3 (Prepare barbecue), s_4 (Decorate venues) , s_5 (Take photos)	(3, 1)	50

Example 1 Suppose we have three tasks t_1-t_3 whose information is shown in Table 1. Specifically, each task has a required skill list, a location and a budget. In this example, all the three tasks require workers with the skills of mixing drinks (s_1), preparing food (s_2), preparing barbecue (s_3), decorating venues (s_4) and taking photos (s_5). They have different locations. For example, t_1 is located at the position (1, 1). They all have budget of 50.

We also have six workers whose information is shown in Table 2. Specifically, each worker has a mastered skill list and the corresponding fee of each skill. They also have locations. For example, the worker w_1 masters the skill of s_1 (mixing drinks), s_2 (preparing food) and s_3 (preparing barbecue). If s/he is assigned to a task and the content of the work is to mix drinks (s_1) and prepare barbecue (s_3), s/he will be paid $3 + 10 = 13$.

The arriving and leaving time of the above tasks and workers is shown in Table 3. For example, w_1 arrives at 17:05 and leaves at 17:45. Note that we cannot wait until all the tasks and workers arrive and perform the task assignment, as some tasks or workers have left by then. Thus, we always need to perform task assignment based on partial information (namely the tasks and workers that have arrived) and cannot change the decisions that have been made.

Motivated by the above example, we formalize the Online Multi-skill-Aware Task Assignment (OMATA in short) problem. In our problem, the tasks with requirement on different skills and the workers mastering different skills appear and leave dynamically. We need to perform task assignment before the tasks and workers leave to maximize the total utility, which is defined as the remaining budget of all tasks after paying the assigned workers.

Briefly, we make the following contributions.

- We formally define the OMATA problem and prove that even its offline version is NP-hard.

Table 2 Information of workers

Workers	Skills and fees	Location
w_1	$(s_1, 3), (s_2, 10), (s_3, 10)$	(5, 3)
w_2	$(s_2, 10), (s_3, 3), (s_5, 10)$	(4, 5)
w_3	$(s_1, 10), (s_2, 10), (s_5, 3)$	(1, 2)
w_4	$(s_2, 3), (s_3, 10), (s_4, 10)$	(2, 3)
w_5	$(s_1, 11), (s_2, 10), (s_4, 10)$	(3, 3)
w_6	$(s_2, 10), (s_4, 11), (s_5, 10)$	(2, 0)

Table 3 Timeline

Item	t_1	w_1	w_2	w_3	w_4	t_2	w_5	w_6	t_3
Arriving time	17:00	17:05	17:10	17:15	17:20	17:25	17:30	17:35	17:40
Leaving time	18:00	17:45	17:50	17:55	18:00	18:25	18:10	18:15	18:40

- Inspired by the research of online micro task assignment, we propose the Online-Exact algorithm, which always calculates the optimal assignment through brute force search for each task.
- To release the high time complexity of Online-Exact, we propose the Online-Greedy algorithm which assigns workers for each task via an effective heuristics method.
- We conduct experiments on both synthetic and real dataset to evaluate the proposed methods.

The following of the paper is organized as follows. We review the related work in Section 2, formalize our problem and analyze its hardness in Section 3. Then we propose our solutions in Section 4 and conduct experiments in Section 5. We finally conclude in Section 6.

2 Related work

We review related works from two categories: task assignment in SC and team formation in social networks.

2.1 Task assignment in spatial crowdsourcing

2.1.1 Micro-task assignment

The early works on task assignment in SC mainly focus on micro-tasks, which can be completed by any single worker. Kazemi and Shahabi [3] propose task assignment problem in spatial crowdsourcing under the offline scenario and the goal is to maximize the total number of the assigned tasks. Tong et al. [27] consider online scenario of task assignment and proposes algorithms with theoretical guarantees to maximize the total utility score of the assignment. Song et al. [12] also focus on the online scenario and takes the influence of work space into consideration. Tong et al. [29] study online task assignment to minimize the total moving distance of the workers. Tong et al. [26] propose a prediction-based method to solve the online task assignment problem. Liu et al. [2] present a privacy-preserving task assignment protocol which can protect the privacy for both workers and tasks with acceptable overheads.

Tao et al. [13] recommend routes for workers to maximize the total utility. Zeng et al. [16] assign tasks to workers while trading off quality and latency of task completion. Tong et al. [17] propose a match-based approach to solve the dynamic pricing problem in spatial crowdsourcing. Tran et al. [20] propose a real-time framework for task assignment.

The differences between the above works and this paper is that we focus on the scenario where tasks require specific skills and cannot be completed by single workers generally. Thus, the methods of above works cannot apply to our problem.

2.1.2 Macro-task assignment

We use macro-task to denote the tasks that have requirement on different skills. Gao et al. [4, 7] recommend top-k teams with or without leaders to a macro-task, and the goal is to minimize the total cost of the recommended teams. Song and Cheng et al. [23, 28] study assigning workers for specialty-aware tasks to maximize the total utility score.

The biggest difference between our work and the above studies is that in our paper we consider the dynamics of the tasks and workers. Specifically, the tasks and workers appear dynamically and their information cannot be known in advance. Thus, we must make decisions (assignments) based on current information. Besides, the assignments that have been made cannot be canceled. Otherwise, the user experience of both the workers and the requesters of the tasks may be hurt. Because of the above challenges, existing algorithms such as those in [23, 28] cannot be applied or extended trivially to solve it, as they have to be performed on all of the tasks and workers. But in our problem, this information is inaccessible.

2.2 Team formation problem

Another related topic is the team formation problem, where teams of experts with different skills are formed to complete tasks requiring multiple skills and the goal is often to find a qualified team with the minimal communication cost, which is defined based on the social graph of the experts [6, 9, 10, 19], such as the diameter and the sum of the weights of the minimum spanning tree of the team members' social graph in [9], the communication cost as the sum of the shortest distances between the members and the leader in [19]. Anagnostopoulos and Majumder et al. [6, 10] also consider the workload balance among the members of a team.

The difference between our problem and the above works on the team formation problem and its variants is that (i) the core focus in our paper is the total utility of the assignment and thus we do not consider the social relationships between users; (ii) we aim at assigning workers with multiple tasks while the above works mainly focus on assigning a team of experts with a single task.

3 Problem definition

In this section, we first define our problem and then analyze its hardness via the offline version.

Definition 1 (Worker) A worker w is defined as $< o_w, b_w, e_w, S_w, P_w >$, which means a worker appears at the location o_w at time b_w and will leave the platform at e_w if no task is assigned to her/him. $S_w = \{s_1^w, s_2^w, \cdots, s_{|S_w|}^w\}$ is the set of skills that w masters and $P_w = \{p_1^w, p_2^w, \cdots, p_{|S_w|}^w\}$ is the multi-set of cost for each of the skills in S_w. In other words, if w is assigned to use her/his skill s_i^w to perform a task, the responsible payment is p_i^w.

Definition 2 (Task) A task t is defined as $< o_t, b_t, e_t, S_t, B_t >$, which means that a task t appears at the location o_t at time b_t and will leave if it is not assigned to workers before e_t. $S_t = \{s_1^t, s_2^t, \cdots, s_{|S_t|}^t\}$ is the set of required skills of t. All skills in S_t should be mastered by an assigned worker or t cannot be completed. B_t is the total monetary budget.

We next define how to calculate a worker's reward, which includes two parts: (1) reward for the skills that the worker is required to be responsible for and (2) reward for letting the worker move from her/his current location to the location of the assigned task. The reasons are as follows. First, the reward of a worker roots in the labour s/he contribute. Thus, the reward should include the fees for the specified skills that the workers are assigned. Besides, to complete a task, a worker should move from her/his location to the location of the task, which incurs extra cost. We think this cost should be paid to the assigned worker and the scale of the cost should be calculated according to the distance between the locations of the task and worker.

Definition 3 (Reward of Worker) If worker w is assigned to perform task t and required to be responsible for the skills of $S'_w \subseteq S_w$, the reward of w is $R(w, t, S'_w) = \gamma \cdot dis(o_w, o_t) + \sum_{s \in S'_w} p^w_s$, where $dis(o_w, o_t)$ is the distance between L_w and L_t, which can be Euclidean distance or road network distance, γ is a global parameter representing the unit transportation fee.

We next define the utility of a task as follows.

Definition 4 (Utility of Task) If a set of workers W_t is assigned to task t, the utility of task t is defined as $U(t, W_t) = B_t - \sum_{t \in W_t} R(w, t, S'_w)$, where B_t is the budget of the task and $\sum_{t \in W_t} R(w, t, S'_w)$ is the summation of rewards of workers assigned to t. Note that if an assigned set of workers cannot complete the task, the reward of the task is zero.

We finally define our problem as follows.

Definition 5 (Online Multi-skill-Aware Task Assignment (OMATA) Problem) Given a set of tasks T, a set of workers W and a global unit transportation fee γ, the problem is to assign workers to tasks to maximize the total utility of the completed tasks and the following constraints should be satisfied:

- **Invariable Constraint**: once a set of workers is assigned to a task, it cannot be changed.
- **Skill Constraint**: the workers assigned to a task should be able to cover the required skills;
- **Budget Constraint**: the total rewards of workers assigned to a task cannot exceed the task's total budget;

Hardness Analysis We next prove that even the offline version of the OMATA problem is NP-hard. Thus, the OMATA problem is very difficult to address.

Theorem 1 *The offline version of the OMATA problem is NP-hard.*

Proof We prove through a reduction from the weighted set cover (WSC) problem [25].

In an instance of the WSC problem, we are given a set $U = \{a_1, a_2, \cdots, a_m\}$ and its n subsets $A_1, A_2, \cdots, A_n \subseteq U$. Each A_i has a weight w_i. The goal is to find $\mathcal{A}^* \subseteq \mathcal{A} = \{A_1, A_2, \cdots, A_n\}$ such that $\cup \mathcal{A}^* = U$ and $\sum_{A_j \in \mathcal{A}^*} w_j$ is minimized.

We next show how to transform the WSC problem to an instance of the offline version of OMATA problem. Suppose we only have one task t which requires skills $S_t = U$ and has much enough budget B_t. For n workers $\{w_1, w_2, \cdots, w_n\}$, their required fees for skills are all zero, and we adjust their locations and γ to make their transportation fee to perform t be c_i. We aim to find a set of workers K to maximize the utility of t, which in this case equals $B_t - \sum_{i \in K} c_i$. Thus we only need to minimize $\sum_{i \in K} c_i$. In this way, we reduce an instance of WSC problem to one of the OMATA problem. As the WSC problem is known to be NP-hard [25], the offline version of OMATA problem is also NP-hard. □

4 Method

We introduce the methods to solve the OMATA problem in this section.

4.1 Online-exact algorithm

Basic Idea When a new object obj appears, which could be a task t or a worker w, we calculate the optimal assignment among the current set of tasks T_c, the current set of workers W_c through obj. In other words, obj must be involved in the assignment if such assignment exists.

Algorithm 1: Online-Exact (OE).

```
input  : $T$ (set of tasks), $W$ (set of workers)
output: An assignment $M$ among $T$ and $W$
$M \leftarrow \emptyset$, $T_c \leftarrow \emptyset$, $W_c \leftarrow \emptyset$;
foreach new arrival object $obj$ do
    Call $ELA(T_c, W_c, obj)$ and update $M$;
    Update $T_c$, $W_c$;
return $M$
```

Algorithm 2: Exact local assignment (ELA).

```
input  : $T_c$ (set of current tasks), $W_c$ (set of current workers) and
         $obj$ (the new arriving task or worker)
output: An assignment $M_l$ among $T_c$, $W_c$ and $obj$
if $obj$ is a task then
    $t \leftarrow obj$;
    foreach $s \in S_t$ do
        $W_s \leftarrow \{w|w \in W_c$ and$s \in S_w\}$;
    $\mathbb{M} \leftarrow$ Cartesian Product of All the$W_s$;
    $W^* \leftarrow \arg\max_{m \in \mathbb{M}}$ Reward of Assignment$m$ and update $M_l$;
else
    /* $obj$ is a worker */
    $w \leftarrow obj$;
    foreach $t \in T_c$ do
        foreach $s \in S_t$ do
            $W_s^t \leftarrow \{w|w \in W_c \cup \{w\}$ and $s \in S_w\}$;
        $\mathbb{M} \leftarrow$ Cartesian Product of All the$W_s^t$;
        $W_t^* \leftarrow \arg\max_{m \in \mathbb{M}}$ Reward of Assignment$m$ and update $M_l^t$;
    $M_l \leftarrow M_l^t$ with the maximum utility;
return $M_l$
```

Algorithm Alg. (1) shows the pseudo-code of the Online-Exact algorithm. We first initialize the current assignment and the sets of tasks and workers which are not assigned currently in line 1. In Lines 2-3, for each new arrival object, which could be a task or a worker, we call the Exact Local Assignment (ELA in short) algorithm (see Alg. 2) which performs an optimal assignment for obj and update the assignment M. We update T_c, W_c in line 4 and return the final assignment in line 5. In Alg. 2, if obj is a task t (line 1), for each skill s required by t, we collect the workers who master this skill as set W_s (lines 3-4). We calculate the Cartesian product of all the W_s to enumerate all cases to assign workers for t in line 5 and return the case with the maximal utility in line 6. If obj is a worker w in lines 7-8, for each task $t \in T_c$, we use the method in lines 1-6 to find the assignment M_l^t for t in lines 10-14. We finally update M_l using the assignment with the maximum utility among all $\{M_l^t\}$ in line 15. We return the assignment in line 16.

Complexity We assume the number of skills that a task requires or a worker masters is a constant C. For Alg. (2), the worst case happens when a new worker w arrives. For each task t that have arrived and have not left (line 10), we have to calculate all possible ways to assign a group of workers including w with t (in lines 11-14), which consumes $O(|W_c|^C)$ time. Thus the total complexity is $|T_c||W_c|^C$. As a result, the time complexity of Alg. 1 is $(|T| + |W|)|T||W|^C$.

We next give an running example of Alg. (1).

Example 2 After w_4 arrives, we assign w_1, w_2, w_3 and w_4 to t_1. Specifically, we let w_1 perform the skill of s_1, w_2 perform the skill of s_3, w_3 perform the skill of s_5 and w_4 perform the skills of s_2 and s_4. Thus, the utility gained from t_1 is $50 - 0.1 \times (4.47 + 5 + 1 + 2.23) - (3 + 3 + 3 + 3 + 10) = 26.73$. After that, Online-Exact does not assign workers for t_2 and t_3 as they cannot be completed by the remaining workers.

4.2 Online-Greedy algorithm

Basic Idea Although Online-Exact can optimize the assignment for each task, it has two drawbacks. First, the time complexity is too high. Second, it may consume too many workers to pursue the optimal assignment for a single task and thus works bad when the workers are insufficient. In this subsection, we propose the Online-Greedy (OG for short) algorithm, which considers both the reward gained from a single task and the number of workers the task consumes.

Algorithm 3: Online-Greedy (OG).

```
input  : $T$ (set of tasks), $W$ (set of workers)
output: An assignment $M$ among $T$ and $W$
$M \leftarrow \emptyset$, $T_c \leftarrow \emptyset$, $W_c \leftarrow \emptyset$;
foreach new arrival object obj do
    Call $GLA(T_c, W_c, obj)$ and update $M$;
    Update $T_c$, $W_c$;
return $M$
```

```
Algorithm 4: Greedy local assignment (GLA).
   input : $T_c$ (set of current tasks), $W_c$ (set of current workers) and
           $obj$ (the new arriving task or worker)
   output: An assignment $M_l$ among $T_c$, $W_c$ and $obj$
1  if $obj$ is a task then
2      $t \leftarrow obj$;
3      $W_t \leftarrow \emptyset$;
4      while $t$ cannot be complete by $W_t$ do
5          $w^* \leftarrow \arg\min_{w \in W_c} R(w, t, S'_w)/|S'_w \cap S_t|$;
6          if $w^*$ is NULL then
7              Break;
8          $W_t \leftarrow W_t \cup \{w^*\}$;
9          Remove the covered skill from $S_t$;
10         $B_t \leftarrow B_t - R(w, t, S'_{w*})$;
11     Update $M_l$;
12 else
13     /* $obj$ is a worker */
14     $w \leftarrow obj$;
15     $t^* \leftarrow \arg\min_{t \in T} R(w, t, S'_w)$;
16     Call $GLA(T_c \backslash \{t^*\}, W_c, t^*)$ and update $M_l$;
17 return $M_l$
```

Algorithm Alg. (3) shows the pseudo-code of the Online-Greedy, which is similar to Alg. (1). The only difference lies in the line 3 where we use the Greedy Local Assignment (GLA) algorithm to perform the assignment for the new arriving object obj. Alg. (4) shows the pseudo-code of the GLA algorithm. When a task t appears in line 1, we iteratively assign workers with the minimal $R(w, t, S'_w)/|S'_w \cap S_t|$ to it in lines 4-8. The underlying idea is that we alway assign the worker with low reward and high skill coverage at the same time. Then we update the state of t by removing the covered skills in line 9 and calculating the remaining budget in line 10. If obj is a worker w in lines 12-14, we first find the task t^* with the minimum $R(w, t, S'_w)$ in line 15. Then we assign workers to cover the remaining skills of t^* through recursion in line 16. We finally return the assignment in line 17.

Complexity We still assume the number of skills that a task requires or a worker masters is a constant C. For Alg. (4), the worst case happens when a new worker w arrives. We first find a task t^* in line 15 using $O(T_c)$ time. Then, we use $O(C|W_c|)$ time to assign workers to t^* (in lines 4-10). Thus the total time complexity is $O(|T_c| + |W_c|)$. As a result, the time complexity of Alg. 3 is $(|T| + |W|)^2$.

We next give an running example of Alg. (3).

Example 3 When w_4 arrives, we find t_1 can be completed. We first assign w_4 to t_1 and let w_4 to perform the skills of s_1 and s_5. Then, we assign w_3 to t_1 and let w_3 to perform the skills of s_2, s_3 and s_4 because w_3 has the minimal value of $R(w, t, S'_w)/|S'_w \cap S_t|$. After that, we find t_1 is completed and the utility is 10.77. Similarly, t_2 cannot be completed until w_5 comes. We assign w_5 to t_2 and let w_5 perform the skills of s_1, s_2 and s_4. Afterwards, we assign w_2 who has the minimal value of $R(w, t, S'_w)/|S'_w \cap S_t|$ to t_2. Thus, t_2 can be completed and the utility is 5.5. When t_3 comes, we first assign w_6 to t_3 and let w_6 perform the skills of s_2, s_4 and s_5 because this can get the minimal value of $R(w, t, S'_w)/|S'_w \cap S_t|$. We assign w_1 and let w_1 perform the skills of s_1 and s_3. Finally, the total utility is 24.76.

5 Evaluation

5.1 Experiment setup

Datasets We use both real and synthetic datasets to evaluate our algorithms.

The real dataset is from [1], which is crawled from Meetup, an event-based social network. We use the data from the area of Hong Kong with latitude from 22.209 to 22.609 and longitude from 113.843 to 114.283. We use the events as the tasks and users as the workers. The tags are viewed as the skills. We remove the tasks which cannot be completed as they have the skills that are not mastered by any workers, the workers whose mastered skills do not have any overlap with all the tasks and the skills of workers which are not required by the tasks. Finally, the statistics of the remaining tasks and workers is shown in Table 4. Besides, we generate the budgets of tasks and costs of skills mastered by workers following the Gaussian distribution. Especially, because of the uncertain size of skills that both tasks require and workers master, we generate the mean of budget depending on the size of skills required by tasks. The variance of B_t related to one skill is fixed as 10 and the variance of p^w is fixed as 5. The setting of these generated parameters can also be found in Table 4.

We also use a synthetic dataset for experiments. Specifically, we generate data varying the following parameters.

- $|T|$: the number of tasks;
- $|W|$: the number of workers;
- $|S_t|$: the number of skills that each task requires;
- $|S_w|$: the number of skills that each worker masters;
- Mean of B_t: the mean of budgets of different skills required by the tasks following Gaussian distribution;
- Variance of B_t: the variance of budgets of different skills required by the tasks following Gaussian distribution;
- Mean of p^w: the mean of costs of different skills mastered by the workers following Gaussian distribution;

Table 4 Real dataset

Factor	Value
$\|T\|$	1234
$\|W\|$	3275
Mean of B_t (Generated)	20 40 **60** 80 100
Average $\|P_t\|$	8.62
Average $\|P_w\|$	7.33
Mean of p^w (Generated)	10 20 **30** 40 50
Waiting Time	2h
Skill Range	554
γ (Generated)	**0.1** 0.3 0.5 0.7 0.9

- Variance of p^w: the variance of costs of different skills mastered by the workers following Gaussian distribution;
- Skill Range: the total number of skills;
- γ: the global transaction fee for unit distance.

Statistics of the above parameters are shown in Table 5, where we mark our default settings in bold font. The appearing and leaving time of the tasks and workers are generated following uniform distribution.

Compared Algorithms The following algorithms are compared in the experimental study.

- Baseline, which assigns workers (tasks) to new arriving tasks (workers) according to the appearing order of the workers (tasks);
- Online-Exact (Alg. (1)), which performs assignment for the new arriving tasks or workers via ELA algorithm (Alg. (2));
- Online-Greedy (Alg. (3)), which performs assignment for the new arriving tasks or workers via GLA algorithm (Alg. (4));

Metrics We compare the total utility, runtime and memory cost of the compared algorithms.

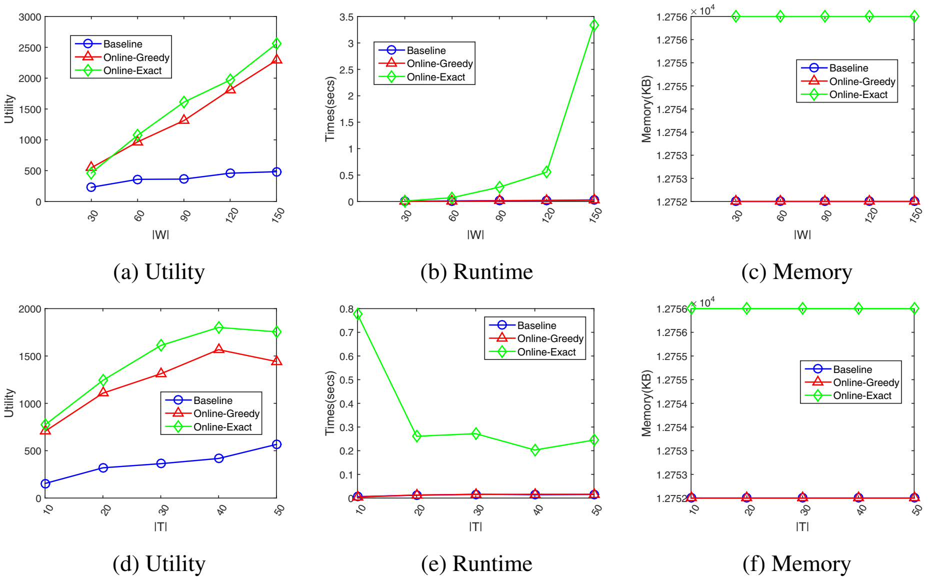
5.2 Experimental results on synthetic datasets

Because of the inefficiency of Online-Exact, we first study the performance of all the three algorithms in a small dataset where the number of tasks and workers is 100x times less than that of Table 5. We then only compare Baseline and Online-Greedy following the setting in Table 5.

Results on the small synthetic datasets The results are shown in Fig. 2. It can be seen that the utility of Online-Greedy is a little less than Online-Exact and outperforms Baseline a lot. Reasonably, the time and memory cost of Online-Exact is extremely more than that of Online-Greedy and Baseline, making it useless in real applications. In Fig. 2e, the runtime of Online-Exact decreases overall with the increment of the number of tasks. The reason is that when there is only a small number of tasks, each task can be assigned to more workers. Due to the high time complexity of Online-Exact, more total time is needed. However, when

Table 5 Synthetic dataset

Factor	Setting
$\|T\|$	1k 2k **3k** 4k 5k
$\|W\|$	3k 6k **9k** 12k 15k
$\|S_t\|$	3 4 **5** 6 7
$\|S_w\|$	3 4 **5** 6 7
Mean of B_t	100 200 **300** 400 500
Variance of B_t	10 20 **30** 40 50
Mean of p^w	10 20 **30** 40 50
Variance of p^w	5 10 **15** 20 25
Skill Range	10 15 **20** 25 30
γ	**0.1** 0.3 0.5 0.7 0.9



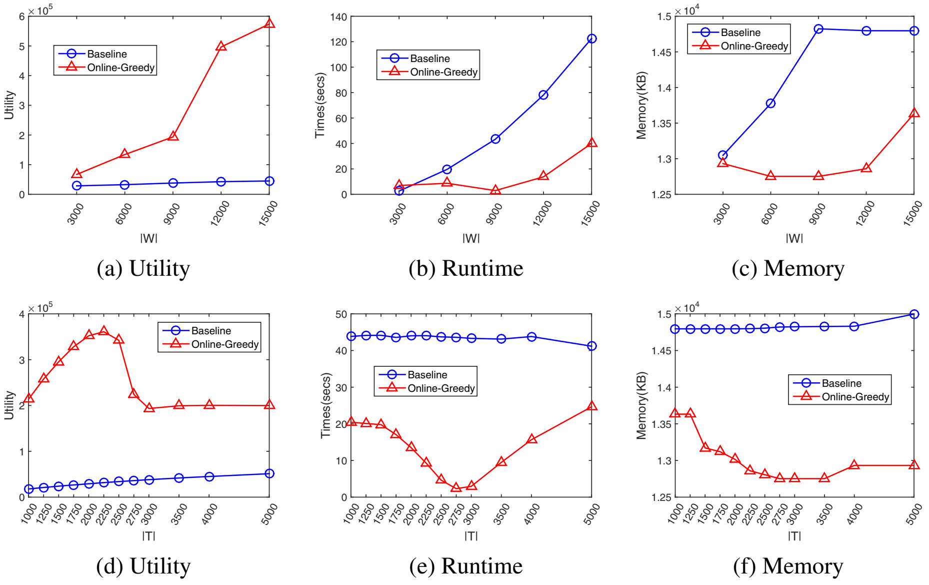
(a) Utility (b) Runtime (c) Memory

(d) Utility (e) Runtime (f) Memory

Fig. 2 Results on varying the number of the workers and tasks on a small dataset

there are more tasks, each task can only be assigned to a small number of workers. As a result, although more tasks should be assigned, the runtime is less.

Effects of the number of workers $|W|$ The results are presented in Fig. 3a-c. We first observe that the total utility increases reasonably with larger $|W|$ for all two algorithms. The



(a) Utility (b) Runtime (c) Memory

(d) Utility (e) Runtime (f) Memory

Fig. 3 Results on varying the number of the workers and tasks

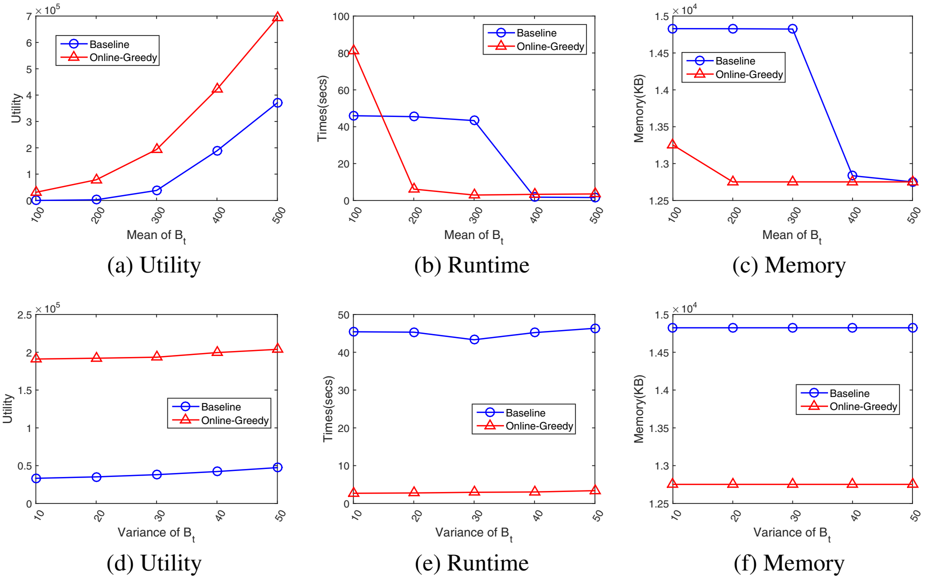
reason is that more workers are available and thus the utility gained from all the completed tasks increases. The increment of Baseline is not obvious. The reason is that Baseline does not try to select suitable workers who can bring more utility. As a result, Online-Greedy outperform Baseline a lot. We can find that Baseline is even slower than Online-Greedy when $|W|$ is more than 6,000. The reason is that Baseline is very ineffective and as a result, there are always many tasks and workers waiting for assigning. Thus Baseline needs to spend much time to process each new arriving task and worker. While in Online-Greedy, the number of waiting tasks and workers is small, making it more efficient than Baseline. Finally, for the memory consumption, the observation is that generally the memory uses increase when $|W|$ increases. Baseline consumes more memory as the heap of unassigned tasks and workers.

Effects of the number of tasks $|T|$ The results are presented in Fig. 3d-f. For the total utility score, we can first observe that Online-Greedy still outperforms Baseline. Besides, the tendency of Online-Greedy is complex. After analyzing the intermediate results of Online-Greedy, the phenomenon can be explained as follows. The total utility is effected by two factors, namely the number of finished tasks and the utility gained from each tasks. When $|T|$ increases from 1,000 to 2,250, more tasks can be finished and the assignment for each task can be optimized as there are adequate available workers. However, when $|T|$ increases from 2,250 to 3,000, although more tasks can be finished, the utility of each finished tasks decreases rapidly. As a result, the total utility deceases, too. When $|T|$ increases more than 3,000, the number of finished workers is stable, as the workers are lacking. Thus, the utility is also steady. As for running time, we can find Online-Greedy is still faster than baseline. For the complex tendency of Online-Greedy, the reason is that first with the increase of $|T|$, nearly all the workers are assigned and thus few workers are accumulated, making the assignment for each new arriving task or worker faster. Then when $|T|$ increases further, more tasks are accumulated, making the running time increase. For memory consumptions, still Baseline consumes more memory for the heap of tasks and workers.

Effects of mean of B_t The results are presented in Fig. 4a-c. Firstly we can observe that the total utility increases reasonably when B_t increases as more utility scores can be obtained from each accomplished task. Online-Greedy still outperforms Baseline obliviously. As for running time, when the mean is 100, Online-Greedy is slower than Baseline. The reason is that the budget is small, thus many tasks and workers cannot be assigned and heap. In this case, Baseline is faster for its low time complexity. When the mean is bigger, for the effectiveness of Online-Greedy, less workers and tasks heap up, making Online-Greedy even faster than Baseline. When the mean is bigger than 400, Baseline can also consume the tasks and workers efficiently as Online-Greedy and runs faster than Online-Greedy. Finally, as for memory, when the mean is small, the heap of tasks and workers in Baseline makes it consume more memory than Online-Greedy. After the mean is large enough, the memory cost of the two algorithms is similar.

Effects of variance of B_t The results are presented in Fig. 4d-f. Firstly, we can see that the utility scores keep stable on varying the variance of B_t and Online-Greedy outperforms Baseline a lot. Second, as Baseline is ineffective, many tasks and workers heap up, making it consume more time and memory than Online-Greedy.

Effects of mean of the cost of workers' each skill The results are presented in Fig. 5a-c. As for the total utility, we can observe that the utility scores decrease as $|p^w|$ increases.

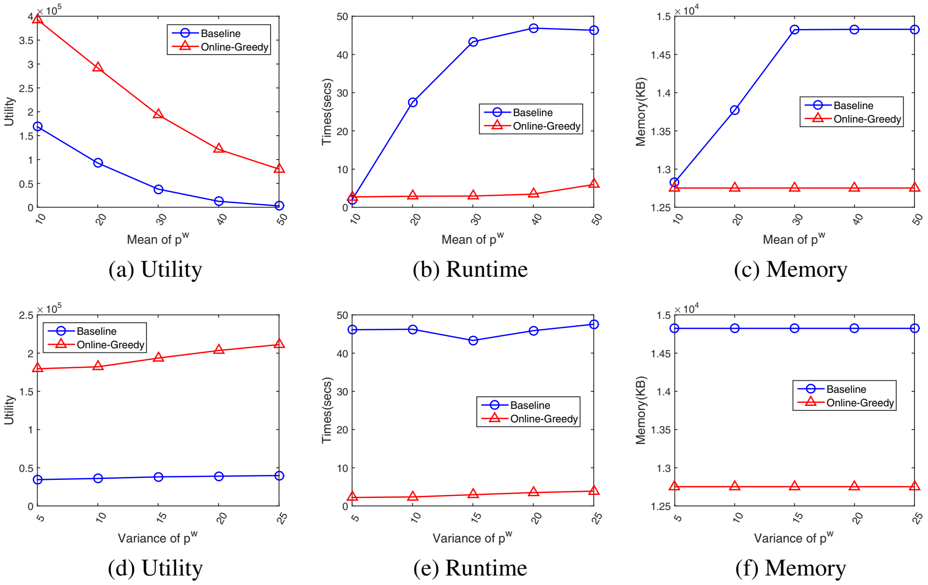


(a) Utility (b) Runtime (c) Memory

(d) Utility (e) Runtime (f) Memory

Fig. 4 Results on varying the mean and variance of the tasks' budget

This is reasonable as utility reduces when workers get more amount of payment. Similarly, with the increment of the mean, more workers and tasks heap up in Baselines, thus Baseline consumes more time and memory while the time and memory cost of Online-Greedy is stable.



(a) Utility (b) Runtime (c) Memory

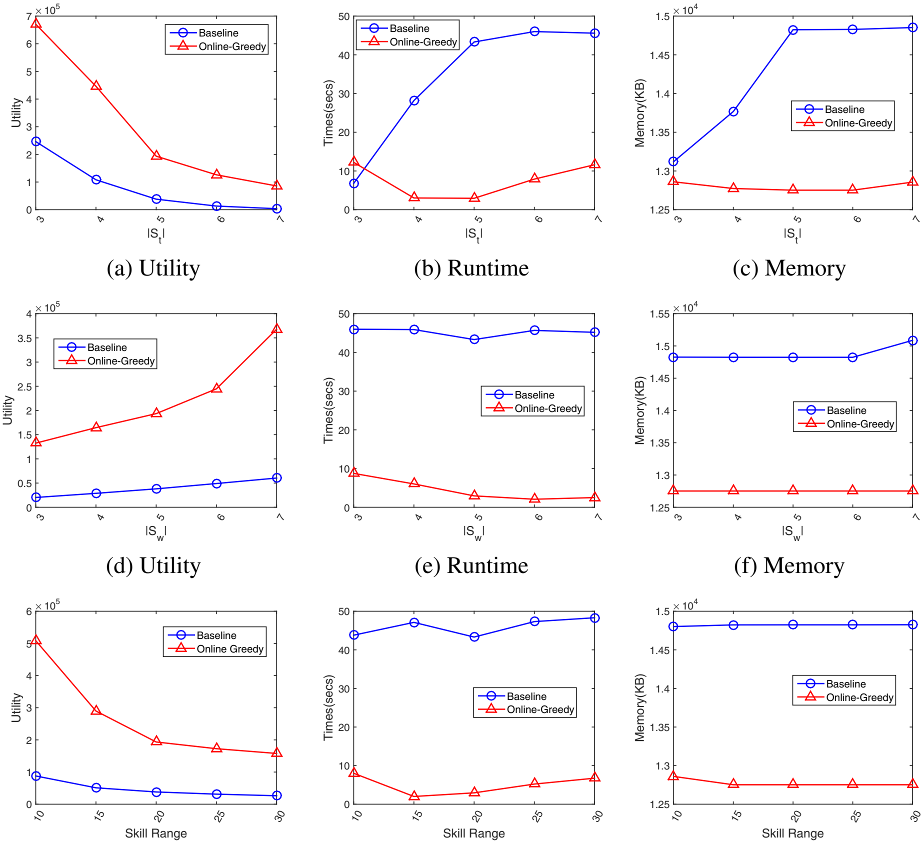
(d) Utility (e) Runtime (f) Memory

Fig. 5 Results on varying mean and variance of the cost of workers' each skill

Effects of variance of the cost of workers' each skill The results are presented in Fig. 5d-f. Online-Greedy has more utility and less time and memory cost. The reason is similar. Baseline is ineffective and many tasks and workers heap up. Thus, although the time complexity of Baseline is lower than Online-Greedy, it still consumes more time and memory.

Effects of the number of required skills of tasks The results are presented in Fig. 6a-c. Online-Greedy still outperforms Baseline in term of utility. As for time, initially, Baseline is more efficient. The reason is that the number of required skills is small. Thus all the tasks can be completed easily and as a result there are not many tasks and workers heaping up in Baseline. However, when the number is bigger, Baseline cannot find workers to complete enough tasks which results in the heap of tasks and workers and makes Baseline slower than Online-Greedy. In terms of memory, with the increment of the number of required skills, Baseline is influenced for the above reason and consumes more memory.

Effects of the number of mastered skills of workers The results are presented in Fig. 6d-f. Online-Greedy has more utility than Baseline. Still for the heap of the tasks and workers, Baseline consumes more time and memory.



(a) Utility (b) Runtime (c) Memory

(d) Utility (e) Runtime (f) Memory

(g) Utility (h) Runtime (i) Memory

Fig. 6 Results on varying the number of required skills of tasks, the number of mastered skills of workers and the number of total skills

Effects of the number of total skills The results are presented in Fig. 6g-i. We find that both algorithms have less utility with the increment of the number of total skills. This is reasonable as generally the tasks become harder to complete. Besides, Online-Greedy outperforms Baseline again in terms of utility, time and memory.

Effects of γ The results are presented in Fig. 7a-c. First, with the increment of γ both of the algorithms has less utility. This is reasonable as the the payment for the workers is increased. Again Online-Greedy outperforms Baseline. For the heap of the tasks and workers in Baseline, it consumes more time and memory. The time cost of Online-Greedy increases quickly as with a larger γ, the degree of accumulate of Online-Greedy is heavier, to which Online-Greedy is sensitive for its higher time complexity. In Fig. 7b, the runtime of Online-Greedy increases with the increment of γ. The reason is that the increment of γ is equivalent to the decrement of the budget. As a result, Online-Greedy has to use more time to find a group of workers through more loops in the algorithm. On the other hand, the runtime of Baseline is stable. The reason is that Baseline does not consider much on budget. It only cares about the order of the workers when assigning a task to them.

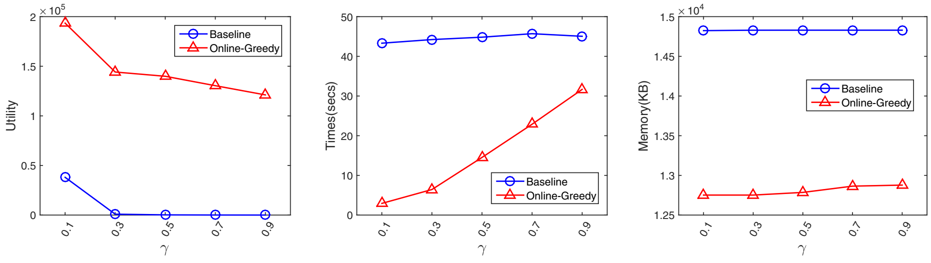
5.3 Experimental results on real datasets

In this subsection, we demonstrate the experimental results on real datasets. As in the real dataset, the total number of skills is 554 (see Table 4), Online-Exact cannot finish in 24 hours. Thus we do not show the results of Online-Exact.

The experimental results are similar with that of the synthetic dataset. Specifically, with the increment of the mean of the tasks' budget on each skill (see Fig. 8a-c), both algorithms have more utility as they can get more profit from each completed tasks. With the increment of the mean of the cost of the workers' each skill (see Fig. 8d-f) and γ (see Fig. 8g-i), both algorithms have less utility as they have to pay more to the workers. Baseline still consumes more time and memory for the heap of tasks and workers.

5.4 Experimental summary

Online-Exact and Online-Greedy always perform better than Baseline in terms of total utility. Online-Exact is only a little better than Online-Greedy but consumes unbearable time and memory cost. In most cases, Online-Greedy even has less time and memory cost, as it can consumes the arriving workers and tasks effectively. While in Baseline, the heap of tasks and workers makes it even more inefficient.

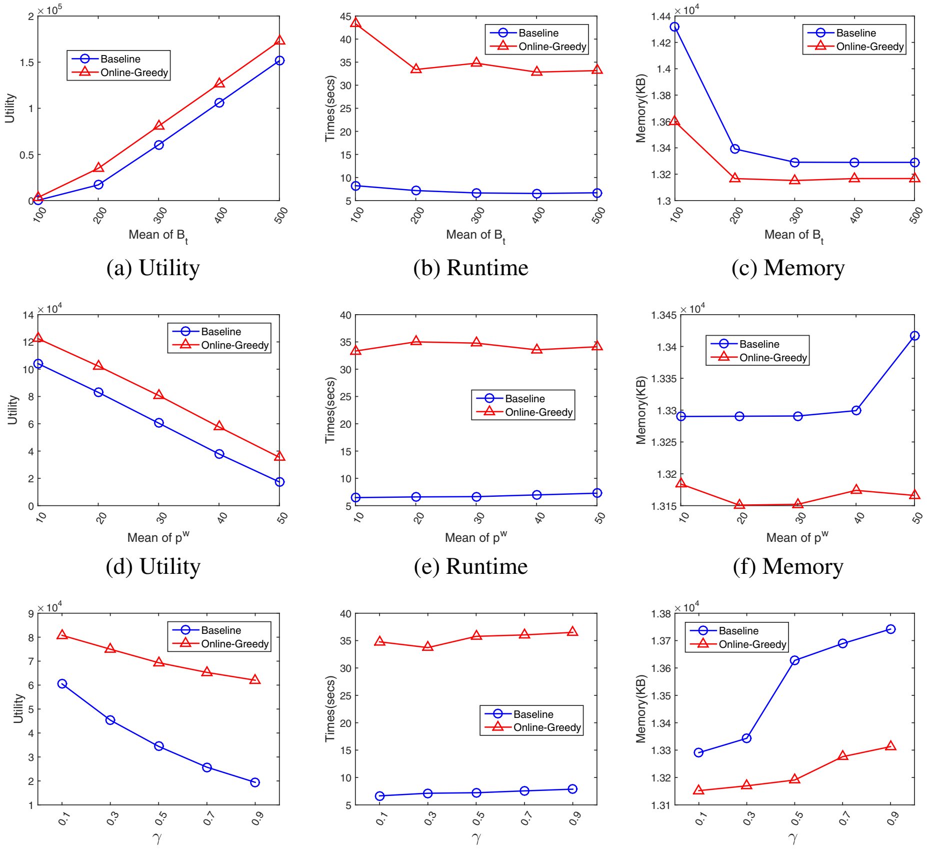


(a) Utility

(b) Runtime

(c) Memory

Fig. 7 Results on varying γ



(a) Utility (b) Runtime (c) Memory

(d) Utility (e) Runtime (f) Memory

(g) Utility (h) Runtime (i) Memory

Fig. 8 Results on varying the mean of the tasks' budget for each skill, the mean of the cost of workers' each skill and γ

6 Conclusion

In this paper we formalize a novel Online Multi-skill-Aware Task Assignment problem. We first analyze the difference of our problem and existing works. Then we prove that even the offline version of the problem is NP-hard. To solve the problem, we first propose Online-Exact algorithm, which always find an optimal assignment for the new coming task or worker. However, because of the high time complexity of Online-Exact, it is not practical in real applications. We next propose the Online-Greedy algorithm, which considers both the utility gained from a task and the occupation on the workers. We finally verify the effectiveness and efficiency of our problems via experiments on both synthetic and real datasets.

References

1. Liu X, He Q, Tian Y, Lee W, McPherson J, Han J (2012) Event-based social networks: linking the online and offline social worlds. KDD:1032–1040

2. Liu A, Wang W, Shang S, Li Q, Zhang X (2018) Efficient task assignment in spatial crowdsourcing with worker and task privacy protection. GeoInformatica 22(2):335–362
3. Kazemi L, Shahabi C (2012) Geocrowd: enabling query answering with spatial crowdsourcing. GIS:189–198
4. Gao D, Tong Y, She J, Song T, Chen L, Xu K (2017) Top-k Team Recommendation and Its Variants in Spatial Crowdsourcing. Data Sci Eng 2(2):136–150
5. Xu Y, Chen L, Yao B, Shang S, Zhu S, Zheng K, Li F (2017) Location-based Top-k Term Querying over Sliding Window. WISE:299–314
6. Anagnostopoulos A, Becchetti L, Castillo C, Gionis A, Leonardi S (2012) Online team formation in social networks. WWW:839–848
7. Gao D, Tong Y, She J, Song T, Chen L, Xu K (2016) Top-k Team Recommendation in Spatial Crowdsourcing WAIM:191–204
8. Chen L, Shang S, Yao B, Zheng K (2018) Spatio-temporal top-k term search over sliding window. World Wide Web:1–18
9. Lappas T, Liu K, Terzi E (2009) Finding a team of experts in social networks. KDD:467–476
10. Majumder A, Datta S, Naidu KVM (2012) Capacitated team formation problem on social networks. KDD:1005–1013
11. Zhao K, Liu Y, Yuan Q, Chen L, Chen Z, Cong G (2016) Towards Personalized Maps: Mining User Preferences from Geo-textual Data. PVLDB 9(13):1545–1548
12. Song T, Tong Y, Wang L, She J, Yao B, Chen L, Xu K (2017) Trichromatic online matching in Real-Time spatial crowdsourcing. ICDE:1009–1020
13. Tao Q, Zeng Y, Zhou Z, Tong Y, Chen L, Xu K (2018) Multi-Worker-Aware Task planning in Real-Time spatial crowdsourcing. DASFAA:301–317
14. Li M, Chen L, Cong G, Gu Y, Yu G (2016) Efficient processing of Location-Aware group preference queries. CIKM:559–568
15. Zhao K, Chen L, Cong G (2016) Topic exploration in Spatio-Temporal document collections. SIGMOD:985–998
16. Zeng Y, Tong Y, Chen L, Zhou Z (2018) Latency-Oriented Task completion via spatial crowdsourcing. ICDE:317–328
17. Tong Y, Wang L, Zhou Z, Chen L, Du B, Ye J (2018) Dynamic pricing in spatial crowdsourcing: a Matching-Based approach. SIGMOD:773–788
18. Chen L, Cong G, Jensen CS, Wu D (2013) Spatial Keyword Query Processing: An Experimental Evaluation. PVLDB 6(3):217–228
19. Kargar M, An A (2011) Discovering top-k teams of experts with/without a leader in social networks. CIKM:985–994
20. Tran L, To H, Fan L, Shahabi C (2018) A Real-Time Framework for Task Assignment in Hyperlocal Spatial Crowdsourcing. TIST 9(3):37:1-37:26
21. Tong Y, Zhou Z (2018) Dynamic task assignment in spatial crowdsourcing. SIGSPATIAL Special 10(2):18–25
22. Tong Y, Chen L, Zhou Z, Jagadish HV, Shou L, Lv W (2018) SLADE: A smart Large-Scale task decomposer in crowdsourcing. TKDE 30(8):1588–1601
23. Song T, Zhu F, Xu K (2108) Specialty-Aware Task assignment in spatial crowdsourcing. AISC:243–254
24. Tong Y, Chen L, Shahabi C (2017) Spatial crowdsourcing: challenges, Techniques, and Applications. PVLDB 10(12):1988–1991
25. Vazirani VV (2013) Approximation algorithms. Springer Science & Business Media, Berlin
26. Tong Y, Wang L, Zhou Z, Ding B, Chen L, Ye J, Xu K (2017) Flexible online task assignment in real-time spatial data. PVLDB 10(11):1334–1345
27. Tong Y, She J, Ding B, Wang L, Chen L (2016) Online mobile micro-task allocation in spatial crowdsourcing. ICDE:49–60
28. Cheng P, Lian X, Chen L, Han J, Zhao J (2016) Task assignment on Multi-Skill oriented spatial crowdsourcing. TKDE 28(8):2201–2215
29. Tong Y, She J, Ding B, Chen L, Wo T, Xu K (2016) Online minimum matching in real-time spatial data: experiments and analysis. PVLDB 9(12):1053–1064
30. Chen Z, Fu R, Zhao Z, Liu Z, Xia L, Chen L, Cheng P, Cao C, Tong Y, Zhang C (2014) gMission: A General Spatial Crowdsourcing Platform. PVLDB 7(13):1629–1632. http://gmission.github.io
31. Tong Y, Zeng Y, Zhou Z, Chen L, Ye J, Xu K (2018) A unified approach to route planning for shared mobility. PVLDB 11(11):1633–1646

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Tianshu Song received the B.E degree from Beihang University, Beijing, China in 2016. He is currently working toward the doctoral degree in the School of Computer Science and Engineering, Beihang University. His research interests include spatial crowdsourcing and spatio-temporal data management.

Ke Xu received the B.E, M.E, and Ph.D degrees from Beihang University, in 1993, 1996, and 2000, respectively. He is a professor with Beihang University, China. His research interests include algorithm and complexity, data mining, and complex networks.

Jiangneng Li is currently an undergraduate student at Beihang University. His current research interests include spatial crowdsourcing and spatial data analysis.

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

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

Affiliations

Tianshu Song[1] · **Ke Xu**[1] · **Jiangneng Li**[1] · **Yiming Li**[1] · **Yongxin Tong**[1]

Ke Xu
kexu@buaa.edu.cn

Jiangneng Li
lijiangneng@buaa.edu.cn

Yiming Li
ymli@buaa.edu.cn

Yongxin Tong
yxtong@buaa.edu.cn

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