Data Sci. Eng.
DOI 10.1007/s41019-017-0037-1

Top-*k* Team Recommendation and Its Variants in Spatial Crowdsourcing

Dawei Gao[1] · Yongxin Tong[1] · Jieying She[2] · Tianshu Song[1] · Lei Chen[2] · Ke Xu[1]

Received: 2 December 2016 / Revised: 2 March 2017 / Accepted: 5 March 2017

Abstract With the rapid development of mobile internet and online to offline marketing model, various spatial crowdsourcing platforms, such as Gigwalk and Gmission, are getting popular. Most existing studies assume that spatial crowdsourced tasks are simple and trivial. However, many real crowdsourced tasks are complex and need to be collaboratively finished by a team of crowd workers with different skills. Therefore, an important issue of spatial crowdsourcing platforms is to recommend some suitable teams of crowd workers to satisfy the requirements of skills in a task. In this paper, to address the issue, we first propose a more practical problem, called *Top-k team recommendation in spatial crowdsourcing* (Top*k*TR) problem. We prove that the Top*k*TR problem is NP-hard and designs a two-level-based framework, which includes an approximation algorithm with provable approximation ratio and an exact algorithm with pruning techniques to address it. In addition, we study a variant of the Top*k*TR problem, called Top*k*TRL, where a team leader is appointed among each recommended team of crowd workers in order to coordinate different crowd workers conveniently, and the aforementioned framework can be extended to address this variant. Finally, we verify the effectiveness and efficiency of the proposed methods through extensive experiments on real and synthetic datasets.

Keywords Spatial crowdsourcing · Top-k · Teams · Leader

1 Introduction

Recently, thanks to the development and wide use of smartphones and mobile Internet, the studies of crowdsourcing are switching from traditional crowdsourcing problems [1–5] to the issues in spatial crowdsourcing markets, such as Gigwalk, Waze and Gmission, where crowd workers (workers for short in this paper) are paid to perform spatial crowsourced tasks (tasks for short in this paper) that are requested on a mobile crowdsourcing platform [6, 7].

Most existing studies on spatial crowdsourcing mainly focus on the problems of task assignment [6–12], which are to assign tasks to suitable workers, and assume that tasks are all simple and trivial. However, in real applications, there are many complex spatial crowdsourced tasks, which often need to be collaboratively completed by a team of crowd workers with different skills. Imagine the following scenario. David is a social enthusiast and usually organizes different types of parties on weekends. On the coming

✉ Yongxin Tong
yxtong@buaa.edu.cn

Dawei Gao
david_gao@buaa.edu.cn

Jieying She
jshe@cse.ust.hk

Tianshu Song
songts@buaa.edu.cn

Lei Chen
leichen@cse.ust.hk

Ke Xu
kexu@buaa.edu.cn

[1] State Key Laboratory of Software Development Environment, School of Computer Science and Engineering, Beihang University, Beijing, China

[2] Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong SAR, China

Published online: 25 March 2017

Saturday, he intends to hold a dance party and needs to recruit some sound engineers, guitarists, cooks and dancers. However, David faces a dilemma that his limited budget cannot afford to recruit all the aforementioned workers. He has to recruit fewer cheap crowd workers who have multiple skills and can take up several responsibilities, e.g., a worker can play the guitar and also manage the sound systems. Therefore, David posts his tasks on a spatial crowdsourcing platform, Gigwalk,[1] and wants to find cheap crowd workers to satisfy his requirements. In fact, many task requestors have the same appeal: *can spatial crowdsourcing platforms recommend several cheaper candidate teams of crowd workers who can satisfy the multiple skills requirement of the tasks?*

Besides satisfying the multiple skills requirement of the tasks, an ideal team of crowd workers is still expected to have a team leader to coordinate different crowd workers and the task requestor in real applications. Back to the aforementioned scenario about David, if a candidate team has a conversable team leader, Bob, who can coordinate with David and other crowd workers, David will prefer the team leaded by Bob. As the approaches which only consider the cost of teams and the multiple skills requirement of the task cannot be suitable for the requirement of team leaders since the friendship of crowd workers is not considered, *another challenge for an intelligent spatial crowdsourcing platform is to discover who is suitable to be a team leader in the recommended team satisfying the multiple skills requirement of the task.*

To further illustrate this motivation, we go through a toy example as follows.

Example 1 Suppose we have five crowd workers $w_1 - w_5$ on a spatial crowdsourcing platform, whose locations are shown in a 2D space (X, Y) in Fig. 1a. Each worker owns different skills, which are shown in the second row in Table 1. Furthermore, each worker has a price for each task and a capacity, which is the maximum number of skills that can be used in a task that he/she performs, which are presented in the third and forth rows in Table 1. Moreover, a team-oriented spatial crowdsourced task and its locality range (the dotted circle) are shown in Fig. 1a. Particularly, the task requires that the recruited crowd workers must cover three skills, $\{e_1, e_2, e_3\}$. To help the task requestor save cost, the spatial crowdsourcing platform usually recommends top-k cheapest teams of crowd workers, who can satisfy the requirement of skills. Furthermore, the recommended teams should not have free riders. In other words, each recommended team cannot satisfy the required skills if any worker in the team leaves. Therefore, in this example, the top-2 cheapest teams without free riders are $\{w_2, w_3\}$ and $\{w_1, w_3\}$, respectively, if the parameter $k = 2$.

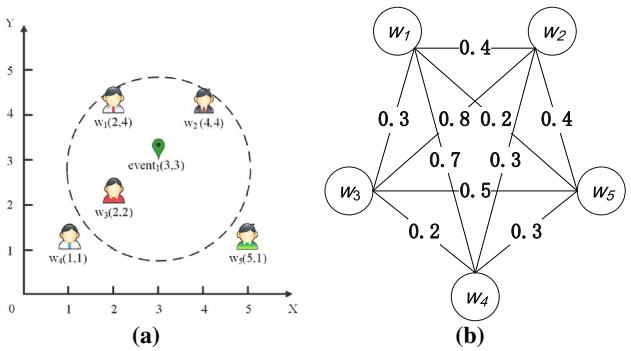


Fig. 1 Location information and friendship. **a** Locations of the task/workers. **b** Friendship among the workers

Table 1 Skill, payoff and capacity information of crowd workers

	w_1	w_2	w_3	w_4	w_5
Skills	$\{e_1, e_2\}$	$\{e_1\}$	$\{e_2, e_3\}$	$\{e_2\}$	$\{e_1, e_2, e_3\}$
Price	2	1	3	1	2
Capacity	1	1	2	1	1

Besides, Fig. 1b shows the relationship among the above five crowd workers. In Fig. 1b, each vertex corresponds to a crowd worker, and the weight of each edge represents the collaborative cost or the friendship between the corresponding two crowd workers. Especially, all the weights in Fig. 1b are normalized into the range [0, 1], and the smaller weight of two arbitrary vertices in Fig. 1b indicates the lower collaborative cost and the better friendship between the corresponding crowd workers. In this case, we already know that the top-2 cheapest teams without considering friendship are $\{w_2, w_3\}$ and $\{w_1, w_3\}$. We observe that the collaborative costs of $\{w_2, w_3\}$ and $\{w_1, w_3\}$ are 0.8 and 0.3, respectively. If the platform hopes that the recommended teams have lower collaborative cost and sets the budget of the collaborative cost to 0.6, the team $\{w_2, w_3\}$ cannot be returned as the result even if it is the cheapest team.

As discussed above, we propose a novel team recommendation problem in spatial crowdsourcing, called the *Top-k team recommendation in spatial crowdsourcing* (Top$k$TR) problem. As the example above indicates, the Top$k$TR problem not only recommends k cheapest teams but also satisfies the constraints of spatial range and skill requirement of tasks, capacity of workers and no free rider in teams. Notice that the Top-1TR problem can be reduced to the classical team formation problem if the constraints on the capacity of workers and free riders are removed. More importantly, the Top$k$TR problem needs to return

[1] http://www.gigwalk.com.

k teams instead of the cheapest team, which is its main challenge. In addition, we study a variant of the Top$k$TR problem, called Top-k team recommendation with leaders in spatial crowdsourcing (Top$k$TRL), which adds a new requirement that the crowd workers in the recommended teams have low collaborative costs or good friendship according to their historical collaborative records.

In summary, we make the following contributions. Note that different to our preliminary work [13], we make new contributions by proposing the Top$k$TRL problem and developing a new approximation algorithm extended from the framework solving the Top$k$TR problem.

- We identify a new type of team-oriented spatial crowdsourcing applications and formally define it as the Top-k team recommendation in spatial crowdsourcing (Top$k$TR) problem and its variant, called the Top$k$TRL problem. Then, we prove that both the Top$k$TR and Top$k$TRL problems are NP-hard.
- For the Top$k$TR problem, we design a two-level-based framework, which not only includes an exact algorithm to provide the exact solution but also can seamlessly integrate an approximation algorithm to guarantee $\ln|E_t|$ theoretical approximation ratio, where $|E_t|$ is the number of required skills of the task.
- For the Top$k$TRL problem, we extend the aforementioned two-level-based framework to address this variant, which has the same approximation ratio.
- We verify the effectiveness and efficiency of the proposed methods through extensive experiments on real and synthetic datasets.

The rest of the paper is organized as follows. In Sect. 2, we formally define the Top$k$TR problem and its variant, called the Top$k$TRL problem, and prove their NP-hardness. In Sect. 3, we present a two-level-based framework and its exact and approximation solutions. Moreover, the new algorithm which is extended from the aforementioned framework is proposed to address the Top$k$TRL problem in Sect. 4. Extensive experiments on both synthetic and real datasets are presented in Sect. 5. We review related works and conclude this paper in Sects. 6 and 7, respectively.

2 Problem Statement

In this section, we first formally define the Top$k$TR problem and, then, formulate the variant of the Top$k$TR problem, called the Top$k$TRL problem.

2.1 Top$k$TR Problem

In this subsection, we formally define the *Top-k team recommendation in spatial crowdsourcing* (Top$k$TR) problem and prove that this problem is NP-hard. For convenience of discussion, we assume $E = \langle e_1, \ldots, e_m \rangle$ to be the universe of m skills.

Definition 1 (*Team-oriented spatial crowdsourced task*) A team-oriented spatial crowdsourced task ("task" for short), denoted by $t = \langle \boldsymbol{l}_t, E_t, r_t \rangle$, at location $\boldsymbol{l}_t$ in a 2D space is posted to the crowd workers, who are located in the circular range with the radius r_t around $\boldsymbol{l}_t$, on the platform. Furthermore, $E_t \subseteq E$ is the set of the required skills of the task t for the recruited team of crowd workers.

Definition 2 (*Crowd worker*) A crowd worker ("worker" for short) is denoted by $w = \langle \boldsymbol{l}_w, E_w, p_w, c_w \rangle$, where $\boldsymbol{l}_w$ is the location of the worker in a 2D space, $E_w \subseteq E$ is the set of skills that the worker is good at, p_w is the payoff for the worker to complete a crowdsourced task, and c_w is the capacity of the worker, namely the maximum number of skills used by the worker to complete a crowdsourced task.

Note that the team-oriented spatial crowdsourced tasks studied in this paper, e.g., organizing a party, renovating a room, usually need to be completed in teams. Though a worker may be good at multiple required skills, he/she cannot finish all the works by himself/herself. Therefore, we limit the capacity of each worker to balance the workload of the whole team. To simplify the problem, we assume that each worker receives the same payoff for different tasks since the capacity of the used skills of each user can be restricted. On the one hand, these workers often have similar workloads and do not need a team leader to do a task. On the other hand, our model can be also easily extended to address the scenario where workers ask for different rewards for his/her different skills. Finally, we define our problem as follows.

Definition 3 (*TopkTR problem*) Given a team-oriented spatial crowdsourced task t, a set of crowd workers W, and the number of recommended crowdsourced teams k, the Top$k$TR problem is to find k crowdsourced teams, $\{g_1, \ldots, g_k\}$ $(\forall g_i \subseteq W, 1 \le i \le k)$ with k minimum $\text{Cost}(g_i) = \sum_{w \in g_i} p_w$ such that the following constraints are satisfied:

- Skill constraint: each required skill is covered by the skills of at least one worker.
- Range constraint: each worker $w \in g_i$ must locate in the restricted range of the task t.
- Capacity constraint: the number of skills used by each worker $w \in g_i$ cannot exceed w's capacity c_w.
- Free-rider constraint: no team still satisfies the skill constraint if any worker in the team leaves.

Theorem 1 *The TopkTR problem is NP-hard.*

Proof When $k = 1$ and the capacity constraint is ignored, such special case of the Top$k$TR problem is equivalent to the team formation problem [14], which has been proven to be NP-hard. Therefore, the Top$k$TR problem is also an NP-hard problem. □

2.2 Top$k$TRL Problem

In this subsection, we further formally define the Top$k$TRL problem, which is a variant of the Top$k$TR problem that considers a team leader for each recommended team. Notice that the team leader in a specific crowdsourced team is also a crowd worker in the team. We first define the concepts of the relationship network of all the crowd workers and the collaborative cost for a team leader in a crowdsourcd team and then describe the definition of the Top$k$TRL problem.

Definition 4 (*Relationship network of crowd workers*) Given a set of crowd workers W, the relationship network of crowd workers in W is represented as a graph $G = (W, F)$, where each vertex in G is a crowd worker, and an edge of any two crowd workers (vertices) w_i and w_j $e_{w_i,w_j} \in F$ measures the friendship between w_i and w_j. Notice that the weight of an edge is evaluated by an arbitrary function, which is denoted by the $d(.,.)$ and is normalized to the range [0, 1]. In particular, the smaller weight of two crowd workers represents the better friendship of the two crowd workers.

According to the relationship network of crowd workers, we define the collaborative cost of a given team leader in a crowdsourced team.

Definition 5 (*Collaborative cost of a team leader*) Given a relationship network of crowd workers $G(W, F)$, a crowdsourced team g of crowd workers and a team leader l, the collaborative cost of the team leader l is defined as

$$CC(g, l) = \sum_{w \in g} d(w, l)$$

where $d(.,.) \in [0, 1]$ is the weight function between two crowd workers.

Definition 6 (*TopkTRL problem*) Given a team-oriented spatial crowdsourced task t, a set of crowd workers W, the number of recommended crowdsourced teams k, and a budget B of collaborative cost, the Top$k$TRL problem is to find k crowdsourced teams $\{g_1, \ldots, g_k\}(\forall g_i \subset W, 1 \leqslant i \leqslant k)$, each of which g_i has a team leader $l_i \in g_i$, with k minimum $\text{Cost}(g_i) = \sum_{w \in g_i} p_w$ such that the following constraints are satisfied:

- Collaborative cost constraint: the collaborative cost of each team leader l_i is lower than and equal to the given budget B, namely $CC(g_i, l_i) \leq B, \forall i \in \{1, \ldots, k\}$.
- The four constraints of Top$k$TR are satisfied.

Notice that the most important constraint of the Top$k$TRL problem is the collaborative cost constraint, which measures the friendship in the team consisting of crowd workers.

Similar to the Top$k$TR problem, the Top$k$TRL problem is also proven as to be NP-hard.

Theorem 2 *The TopkTRL is NP-hard.*

Proof The Top$k$TR problem is a special case of the Top$k$TRL problem when the budget of the collaborative cost is equal to the cardinality of the given set of crowd workers. Based on Theorem 1, we know the Top$k$TR problem is NP-hard, so the Top$k$TRL problem is also NP-hard. □

Algorithm 1: Two-Level-based Framework

input : $W = \{w_1, \cdots, w_{|W|}\}, t, k$, and top-1 function top-1(.,.)
output: Top-k teams $G = \{g_1, \cdots, g_k\}$.

```
1  Queue ← ∅; G ← ∅;
2  Insert the team generated by the function top-1(W,t) into Queue;
3  while Queue ≠ ∅ do
4      res ← top of Queue;
5      G ← G ⋃ {res};
6      if |G| = k then
7          return G;
8      Remove top of Queue;
9      foreach w ∈ res do
10         Insert the team generated by the function top-1(W_res − {w},t) into Queue;
```

3 A Two-Level-Based Framework

To solve the problem effectively, we present a two-level-based algorithm framework. The first level aims to find the current top-1 feasible team with the minimum price, and the second level utilizes the function in the first level to iteratively maintain the top-k best teams. Particularly, the two-level-based framework has a nice property that the whole algorithm can keep the same approximation guarantee of the algorithm as in the first level.

3.1 Overview of the Framework

The main idea of the two-level framework is that the top-2 best team can be discovered if and only if the top-1 best team is found first. In other words, after excluding the top-1 best team from the solution space, not only the size of the solution space is shrunken, but also the global top-2 best team must be the local top-1 best team in the shrunken solution space. The function of finding the local top-1 best team is denoted as the top-1 function in the first level, which will be described in details as the approximation algorithm and the exact algorithm in Sects. 3.2 and 3.3, respectively.

The framework is shown in Algorithm 1. We first initialize an empty priority queue of teams *Queue*, which sorts the elements in non-decreasing prices of the teams, and the top-k teams G in lines 1–2. In line 2, we use a given algorithm, which can be exact or approximate, to get the exact or approximate top-1 team and insert it into *Queue*. In lines 4–10, if *Queue* is not empty, we get the top element *res* of *Queue* and insert *res* into G. For each w in res, we reduce the solution space of *res* to $W_r es - \{w\}$, find a solution in it, and insert the solution into *Queue*. We repeat this procedure until we get k teams.

As introduced above, the framework has a nice property that the whole algorithm can keep the same approximation guarantee of the algorithm (top-1 function) in the first level.

Theorem 3 *If the top-1 function top-1(.,.) in the framework is an approximation algorithm with approximate ratio of r, the approximate cost of the i-th team in the approximation top-k teams by the framework keeps the same approximate ratio compared to the cost of the corresponding i-th exact team.*

Proof We represent the approximation top-k teams generated by the framework as $\{g_1^a, \ldots, g_k^a\}$, and the exact top-k teams is denoted as $\{g_1^{\text{ex}}, \ldots, g_k^{\text{ex}}\}$. Because the top-1 function top-1(.,.) has approximate ratio of r, $\text{Cost}(g_1^a) \leq r \times \text{Cost}(g_1^{\text{ex}})$. When the framework excludes g_1^a from the solution space and utilizes the top-1 function to obtain the other local top-1 team, it has the following two cases: (1) if $g_1^a = g_1^{\text{ex}}$, we have $g_2^a \leq r \times g_2^{\text{ex}}$; (2) $g_1^a \neq g_1^{\text{ex}}$, $g_2^a \leq r \times g_1^{\text{ex}}$. □

3.2 Top-1 Approximation Algorithm

The main idea of the top-1 approximation algorithm utilizes the greedy strategy to choose the best worker w, who can bring the maximum gain to the current partial team g. Algorithm 2 illustrates the top-1 approximation algorithm. We first initialize a empty team g in line 1. In lines 2–4, when g cannot satisfy the requirement of skills of the task t, denoted by E_t, the algorithm selects a worker w with the maximum ratio of the gain and price for the current team. The function *MAXITEM*(.) is used to calculate the number of skills in E_t that can be covered by a specific team. In line 5, since g may contain free-rider workers, we have to refine the team to eliminate redundant workers. Notice that we only need to scan all workers in the team g by the selected order when the these redundant workers are deleted.

Example 2 Back to our running example in Example 1. The running process of the top-1 approximation algorithm is shown in Table 2, where we mark the largest benefit of each round in bold font. In the first round, we choose w_2 with the biggest benefit 1. Since $\{w_2\}$ cannot handle the task, we proceed to choose w_3 with the biggest benefit of $\frac{2}{3}$. Now, we can handle the task with $\{w_2, w_3\}$ and the price is 4.

Approximation Ratio The approximation ratio of the algorithm is $O(\ln |E_t|)$. Inspired by Majumder et al. [15], it is easy to get the approximation ratio of Algorithm 2. Due to the limited space, the details of the approximation ratio proof are omitted in this paper.

Algorithm 2: Top-1 Greedy Approximation Algorithm

input : $W = \{w_1, \cdots, w_{|W|}\}, t$
output: Team g.

1 $g \leftarrow \varnothing$;
2 **while** *the team g cannot satisfy the requirement of* E_t **do**
3 $\quad w \leftarrow \text{argmax}_{w \in W}\left(\frac{MAXITEM(g \bigcup \{w\}) - MAXITEM(g)}{p_w}\right)$;
4 $\quad g \leftarrow g \bigcup \{w\}$;
5 **return** $Refine(g)$

Table 2 Running process of Top-1 approximation algorithm

Round	w_1	w_2	w_3
1	1/2	**1**	2/3
2	1/2		**2/3**

The largest benefit of each round are shown in bold

Complexity Analysis The time consumed by *MAXITEM* is $O(|E_t|^2 \log(|E_t|))$. Line 3 will be executed at most $|E_t|$ times. The refine step takes $O(|W||E_t|)$ time. Thus, the total time complexity is $O(|W||E_t|^3 \log(|E_t|) + |W||E_t|)$. Since $|E_t|$ is usually very small in real applications, the algorithm is still efficient.

Finally, the following example illustrates the whole process of the complete approximation algorithm based on the two-level-based framework.

Example 3 Back to our running example in Example 1. Suppose $k = 2$ and the required skills of the task $t = \{e_1, e_2, e_3\}$. We first use the Top-1 greedy approximation algorithm to get a team of $\{w_1, w_3\}$ in the first level of the framework. Then, we continue to adopt the Top-1 greedy approximation algorithm to find the local top-1 teams from $W - \{w_1\}$ and $W - \{w_3\}$. The returned teams are $\{w_1, w_3\}$ and $\varnothing$, respectively. Thus, the final top-2 teams generated by the whole framework are $\{w_2, w_3\}$ and $\{w_1, w_3\}$.

3.3 Top-1 Exact Algorithm

Since the number of skills required by a task is often not large, the main idea of the Top-1 exact algorithm is to enumerate the cover state of every proper subset of the intersection of the skills between a worker and a task. We give the definition of cover state as follows.

Definition 7 (*Cover state*) Each cover state $s = \langle E', W', p \rangle$ consists of the covered skill set E', the worker set W' in which each worker participates in covering E'. p is the total price of the workers in W'.

For each proper subset, we maintain a cover state of the covered skills and the total price of workers. We update the global cover state when processing each worker. When we have processed all the workers, the cover states of all the required skills of the task are the exact solution.

The exact algorithm is shown in Algorithm 3. We first get a approximate solution using a greedy algorithm and store the price of the solution in C_g in line 1. We then initialize *state* to store the currently best cover states for different skill sets. In lines 4–10, we successively process each worker in W. For worker w, if w_p is not larger than C_g, we enumerate all the cover states of w_p. For each cover state c, we combine it with cover state *state*. If the combined price is not larger than C_g, we store the current cover state in *temp_state*. We finally store c in *temp_state* and use it to update *state*. After we have processed all the workers in W, we check the cover state of the required skills of task t and its associated team is the best team. In line 4 and line 7, we adopt two pruning strategies. In line 4, we use C_g to prune a single worker whose price is too high. In line 7, we use C_g to prune a new cover state whose price is too high.

Example 4 Back to our running example in Example 1. We first use the top-1 approximation algorithm shown in Algorithm 2 to get an approximate solution $T = \{w_2, w_3\}$ with total price of 4, which is used as the current lower bound. Then, we maintain the cover state using a triple structure, which contains the covered skills, the workers and the total price of the current optimal team for each possible combination of the required skills. w_1 can cover skill 1 or 2 with price 2, which is less than the lower bound of 4, so the cover state of w_1 can be $\{\langle\{e_1\}, \{w_1\}, 2\rangle, \langle\{e_2\}, \{w_1\}, 2\rangle\}$. As w_1 is the first worker we process, we just update the current best cover state as

Algorithm 3: Top-1 Exact Algorithm

```
   input : W = {w_1, ..., w_|W|}, t
   output: Team g.
1  C_g ← Price of Top-1 Greedy Approximation Algorithm(W, t);
2  state ← ∅;
3  foreach w ∈ W do
4      if p_w ⩽ C_g then
5          foreach cover condition of w as c do
6              foreach s ∈ state do
7                  if s.P + c.P ⩽ C_g then
8                      Insert new cover condition of s + c into temp_state;
9              Insert c into temp_state;
10         update state using temp_state and clear temp_state;
11 T ← cover condition of skills E_t;
12 return T;
```

$\{\langle\{e_1\},\{w_1\},2\rangle,\langle\{e_2\},\{w_1\},2\rangle\}$. We then proceed to process w_2. We combine the only cover state, $\langle\{e_1\},\{w_2\},1\rangle$ with the cover states in *state*, and then, we get a new cover state of $\langle\{e_1,e_2\},\{w_1,w_2\},2\rangle$. After processing w_2, the current best cover state is $\{\langle\{e_1\},\{w_2\},1\rangle,\langle\{e_2\},\{w1\},2\rangle,\langle\{e_1,e_2\},\{w_1,w_2\},3\rangle$. We can process w_3 similarly and the final cover state is $\{\langle\{e_1\},\{w_2\},1\rangle,\langle\{e_2\},\{w_1\},2\rangle,\ \langle\{e_1,e_2\},\{w_1,w_2\},3\rangle,\langle\{e_2,e_3\},\{w_3\},3\rangle,\langle\{e_1,e_2,e_3\},\{w_2,w_3\},4\rangle\}$ and the best team is $\{w_2,w_3\}$.

Complexity Analysis Line 3 runs $|W|$ times, line 5 runs $C(|E_t|,|E_t|/2)$ times, and line 8 runs $2^{|E_t|}$ times. Therefore, the total time complexity is $O(|W|(2^{|E_t|}))$. When $|E_t|$ is not too large, the exact algorithm can be used.

4 Solutions of Top$k$TRL

In this section, we extend the above two-level-based framework to address the Top$k$TRL problem, which not only recommends k crowdsourced teams satisfying all the constraints of the Top$k$TR problem but also assigns a team leader for each recommended team.

The basic idea of the extended framework is to check whether a certain crowd worker of the first team in *Queue* can be assigned as the leader. If such crowd worker exists, we assign her/him as the leader and return the team as one of the top-k teams. Otherwise, we skip the team and continue to search other teams from the solution space which excludes the team skipped.

The extended framework is shown in Algorithm 4. We first initialize an empty priority queue of teams *Queue*, which sorts the elements in non-decreasing prices of the teams, and the top-k teams G in lines 1–2. In line 3, we use a given algorithm, which can be exact or approximate, to get the exact or approximate top-1 team and insert it into *Queue*. In line 4, we get the top element *res* of *Queue*. Different from Algorithm 1 which inserts *res* into G directly, in lines 5–8 we check whether the collaborative cost of a certain crowd worker w_l in *res* satisfies the collaborative cost constraint, which means s/he can be assigned as the leader. If a leader can be found, we insert *res* into G. In lines 11–13, for each w in res, we reduce the solution space of *res* to $W_r es - \{w\}$, find a solution in it and insert the solution into *Queue*. We repeat this procedure until we get k teams.

Example 5 Back to the running example in Example 1. We still suppose $k = 2$, the required skills of the task $t = \{e_1,e_2,e_3\}$ and the given budget is 0.6. The social network of workers is shown in Fig. 1b. We first use the Top-1 greedy approximation function to get a team of $\{w_1,w_3\}$. Then, we traverse the crowd workers in the team and assign w_1 as the leader. When continuing to run the top-1 function to find the local top-1 team from $W-\{w_1\}$ and $W-\{w_3\}$, we find $\{w_2,w_3\}$ as the local optimal team. However, the collaborative cost between w_2 and w_3 is larger than the given budget and thus there is no team in the solution space that can cover the required skills. Finally, there is only one team $\{w_1,w_3\}$ with leader w_1 satisfying all the constraints. We next show that the extended framework also has the nice property shown in Theorem 4.

Theorem 4 *If the top-1 function top-1(.,.) in the extended framework is an approximation algorithm with approximate ratio of r, the approximate cost of the i-th team in the approximation top-k teams recommended by the extended*

Algorithm 4: Two-Level-based Framework With Leaders

```
   input  : W = {w_1, ..., w_|W|}, t, G(v, e), B.
   output: Top-k teams G = {g_1, ..., g_k}.
1  Queue ← ∅; G ← ∅;
2  Insert the team generated by the function top-1(W, t) into Queue;
3  while Queue ≠ ∅ do
4      res ← top of Queue;
5      foreach w_l ∈ res as the leader do
6          if d(w_l, res) ⩽ B then
7              G ← G ⋃ {res};
8              break;
9      if |G| = k then
10         return G;
11     remove top of Queue;
12     foreach w ∈ res do
13         insert the team generated by top-1(W_res − {w}, t) into Queue;
```

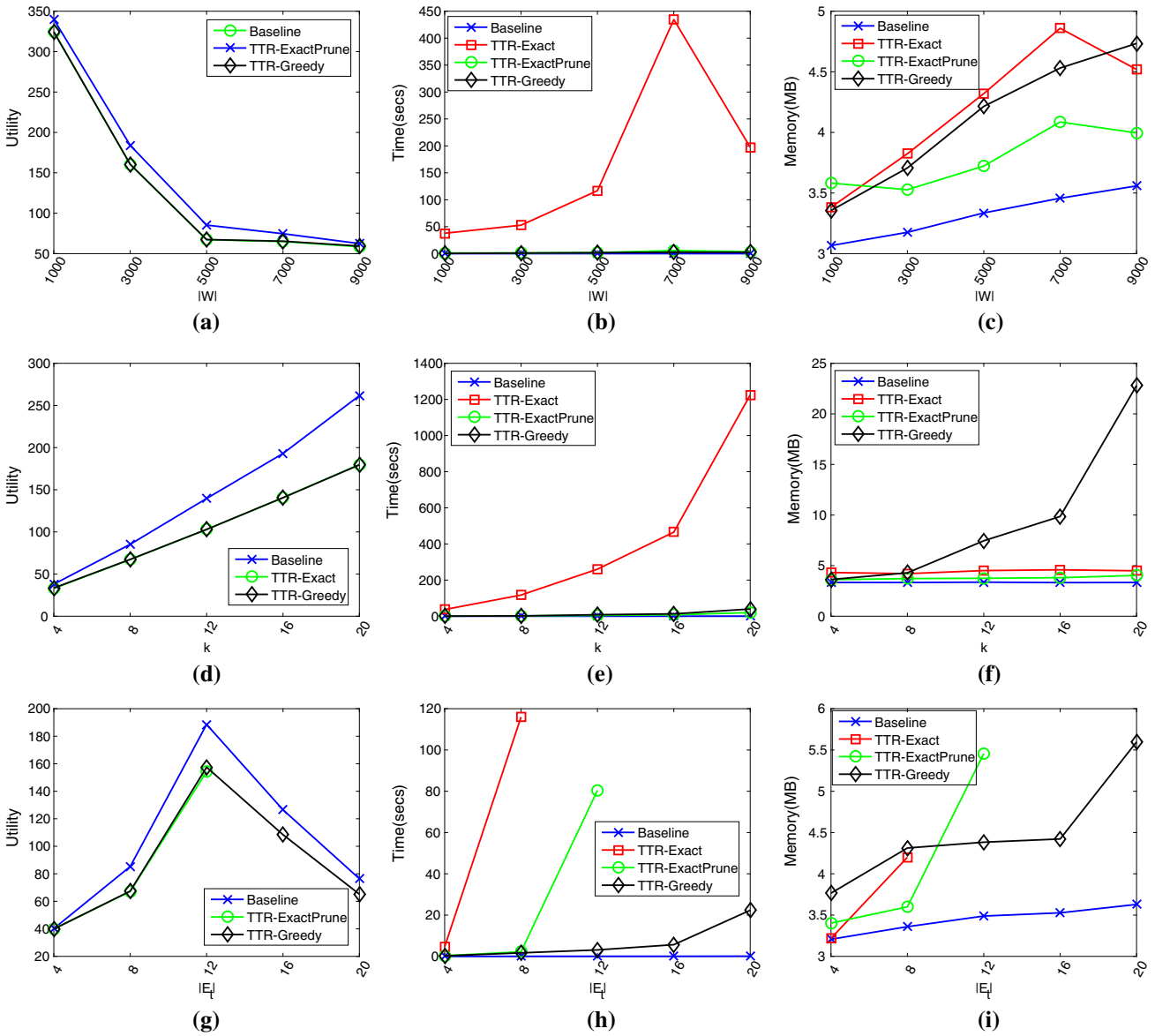


Fig. 2 Results on varying $|W|$, k, and $|E_t|$. **a** Utility of varying $|W|$, **b** time of varying $|W|$, **c** memory of varying $|W|$, **d** utility of varying k, **e** time of varying k, **f** memory of varying k, **g** utility of varying $|E_t|$, **h** time of varying $|E_t|$ and **i** memory of varying $|E_t|$

framework keeps the same approximate ratio compared to the cost of the corresponding i-th exact team.

Proof We use Algorithm 2 as the top-1 function. For the Top$k$TR problem, we represent the cheapest team generated by Algorithm 2 as t^a, the exact cheapest team is denoted by t^{ex}, and the approximation ratio is r. When considering the problem with leader, the exact team is represented as $t^{\text{ex}}_{\text{leader}}$. If $t^a \leqslant B$, t^a is also a solution of the problem considering the leaders. Then we have $t^a \leqslant t^{\text{ex}} \times r$, $t^{\text{ex}} \leqslant t^{\text{ex}}_{\text{leader}}$, and $t^{\text{ex}}_{\text{leader}} \leqslant t^a$, so there is the inequation:

$$t^{\text{ex}}_{\text{leader}} \leqslant t^a \leqslant t^{\text{ex}} \times r \leqslant t^{\text{ex}}_{\text{leader}} \times r$$

Table 3 Synthetic dataset

Factor	Setting
$\|W\|$	1000, 3000, **5000**, 7000, 9000
k	4, **8**, 12, 16, 20
$\|E_t\|$	4, **8**, 12, 16, 20
$\mu_{\|E_w\|}$	2, 4, **6**, 8, 10
$\sigma_{\|E_w\|}$	8, 10, **12**, 14, 16
Scalability ($\|W\|$)	10K, 30K, 50K, 70K, 90K

Default settings are shown in bold

Then for Algorithm 2, the extended framework has the approximation ratio $r = \ln |E_t|$. □

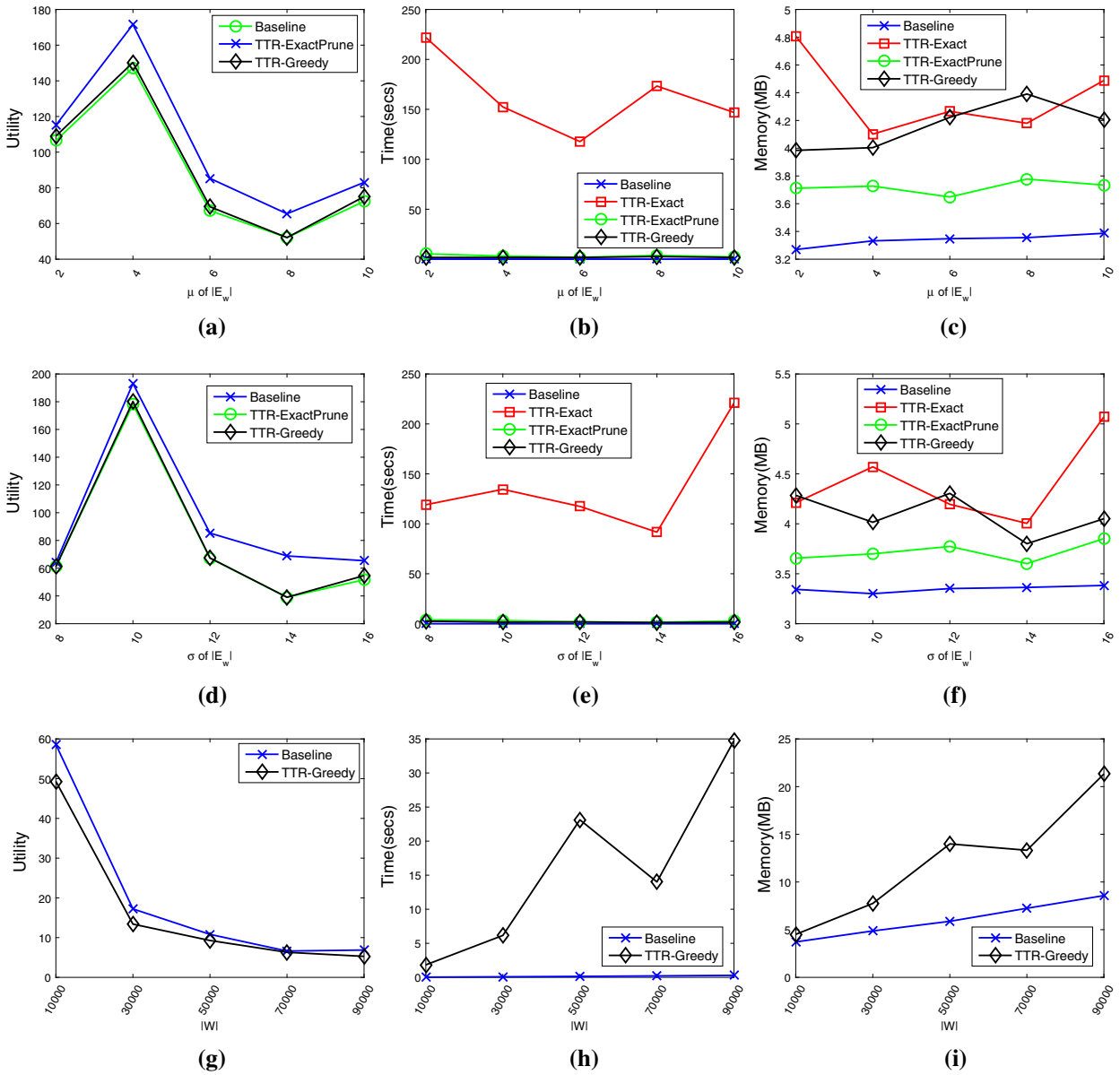


Fig. 3 Results on varying $\mu_{|E_w|}$, $\sigma_{|E_w|}$, and scalability test. **a** Utility of varying $\mu_{|E_w|}$, **b** time of varying $\mu_{|E_w|}$, **c** memory of varying $\mu_{|E_w|}$, **d** utility of varying $\sigma_{|E_w|}$, **e** time of varying $\sigma_{|E_w|}$, **f** memory of varying $\sigma_{|E_w|}$, **g** utility of scalability test, **h** time of scalability test and **i** memory of scalability test

5 Experimental Study

5.1 Experimental Setup

We use a real dataset collected from gMission [16], which is a research-based general spatial crowdsourcing platform. In the gMission dataset, every task has a task description, a location, a radius of the restricted range, and the required skills. Each worker is also associated with a location, a set of his/her owning skills, a price, and a capacity of skills that s/he completes a task. Currently, users often recruit crowd workers to organize all kinds of activities on the gMission platform. In this paper, our real dataset includes the information of 11,205 crowd workers, where the average number of skills and the average capacity owned by the workers are 5.46 and 4.18, respectively. We also use synthetic dataset for evaluation. In the synthetic dataset, the capacity and the number of skills owned by a worker follow uniform distribution in the range of 1–20, respectively. Statistics of the synthetic dataset are shown in Table 3, where we mark our default settings in bold font. In Table 3, $|W|$ is the number of workers. The parameter k represents the number of the result teams. $|E_t|$ is the average number of the required skills of the task t. $\mu_{|E_w|}$ and $\delta_{|E_w|}$ represent

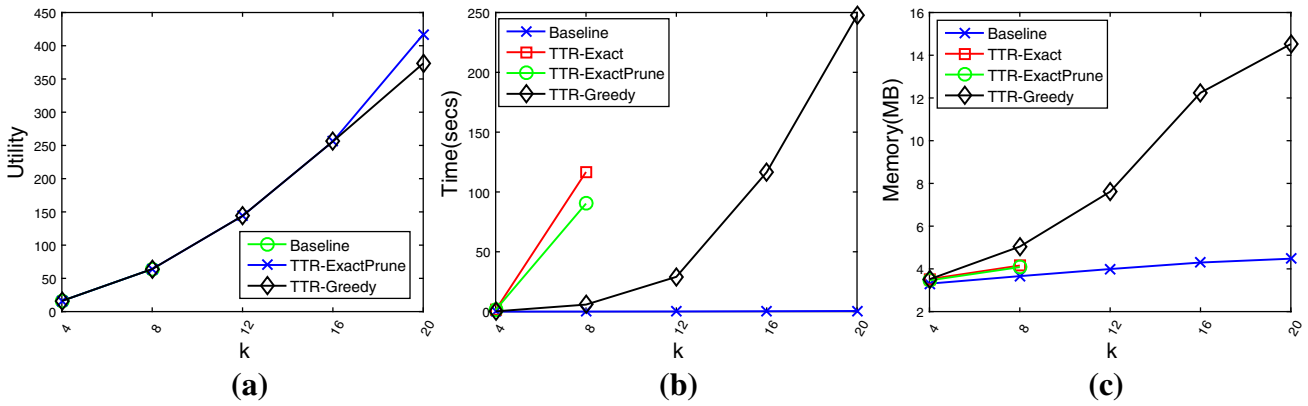


Fig. 4 Performance on the real dataset. **a** Utility in real data, **b** time in real data and **c** memory in real data

the mean and the standard deviation of $|E_w|$ in the normal distribution.

Based on the two-level-based framework, we evaluate an approximation algorithm (Algorithm 1 + Algorithm 2), called TTR-Greedy, and two exact algorithms (Algorithm 1 + Algorithm 3), called TTR-Exact (which does not use the proposed pruning rules) and TTR-ExactPrune, and a baseline algorithm in terms of total utility score, running time and memory cost, and study the effect of varying parameters on the performance of the algorithms. The total utility score is the total price of the top-k teams that we obtain from the algorithms, since we prove that each team can keep the same approximate ratio in the framework. Meanwhile, the baseline algorithm uses a simple random greedy strategy, which first finds the best team, then randomly removes a worker from the best team from the set of workers, and iteratively finds the other $k-1$ best teams following the two steps above. The algorithms are implemented in Visual C++ 2010, and the experiments were performed on a machine with Intel(R) Core(TM) i5 2.40GHz CPU and 4GB main memory.

5.2 Evaluation for Top$k$TR

In this subsection, we test the performance of our proposed algorithms for the Top$k$TR problem through varying different parameters.

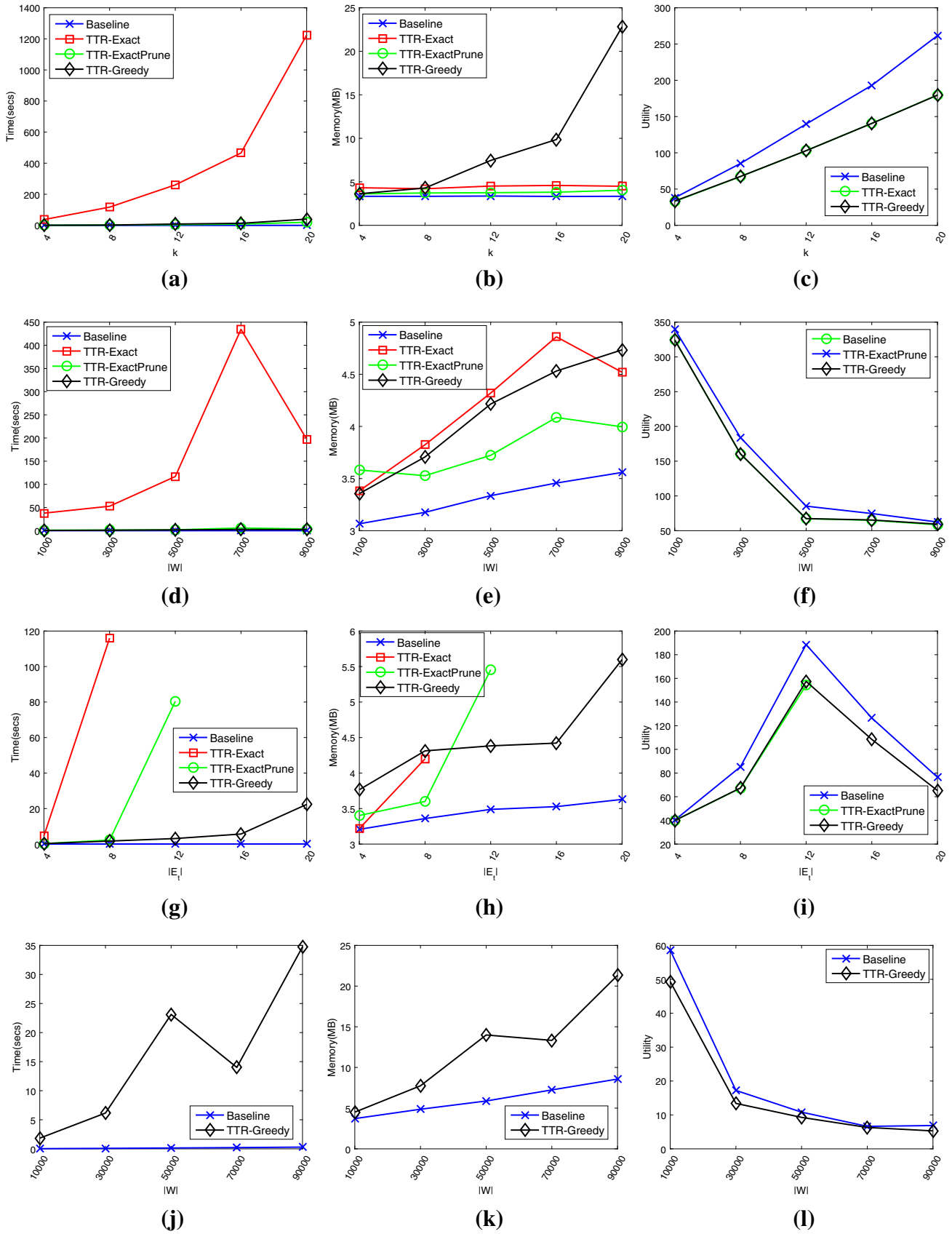
Effect of Cardinality of W The results of varying $|W|$ are presented in Fig. 2a–c. Since TTR-Exact and TTR-ExactPrune return the same utility results, only utility results of TTR-ExactPrune are plotted. We can first observe that the utility decreases as $|W|$ increases, which is reasonable as more high-quality workers can are available. Also, we can see that TTR-Greedy is nearly as good as the exact algorithms. As for running time, TTR-Exact consumes more time with more workers due to larger search space while the TTR-ExactPrune is quite efficient due to its pruning techniques. The other algorithms do not vary much in running time. For memory, TTR-ExactPrune is the most efficient while TTR-Exact and TTR-Greedy are less efficient.

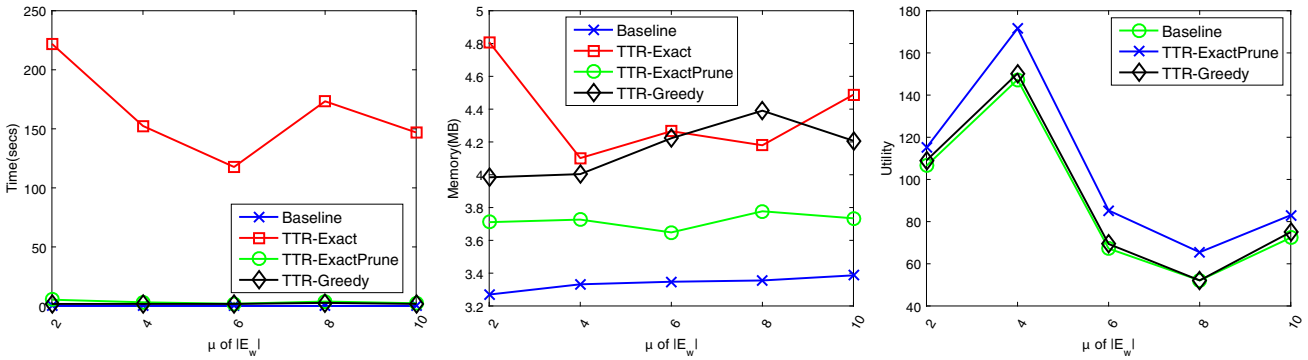
Fig. 5 Results on varying k, $|W|$, E_t and B in the uniform distribution. ▶ **a** Utility of varying k, **b** time of varying k, **c** memory of varying k, **d** utility of varying $|W|$, **e** utility of varying $|W|$, **f** utility of varying $|W|$, **g** utility of varying $|E_t|$, **h** time of varying $|E_t|$, **i** utility of varying $|E_t|$, **j** utility of varying B, **k** time of varying B and **l** memory of varying B

Effect of Parameter k The results of varying k are presented in Fig. 2d–f. We can observe that the utility, running time and memory generally increase as k increases, which is reasonable as more teams need to be recommended. Again, we can see that TTR-Greedy is nearly as good as the exact algorithms but runs much faster. Also, we can see that the pruning techniques are quite effective as TTR-ExactPrune is much faster than TTR-Exact. Finally, TTR-Greedy is the most inefficient in terms of memory consumption.

Effect of the Number of Required Skills in Tasks The results are presented in Fig. 2g–i. We can see that the utility values increase first with increasing number of required skills $|E_t|$ but decrease later when $|E_t|$ further increases. The possible reason is that when $|E_t|$ is not large, the required skills are still quite diverse and thus more workers need to be hired to complete the task as $|E_t|$ increases. However, as $|E_t|$ becomes too large, many workers may use their own multiple skills to complete the task and thus less workers may be needed. As for running time and memory, we can observe that the values generally increase. Again, TTR-ExactPrune is highly inefficient compared with the other algorithms. Notice that the exact algorithms run very long time when $|E_t|$ is large, so we do not plot their results when $|E_t|$ is larger than 12.

Effect of the Distribution of the Number of Skills per Each Worker (μ *and* σ) The results are presented in Fig. 3a–f.

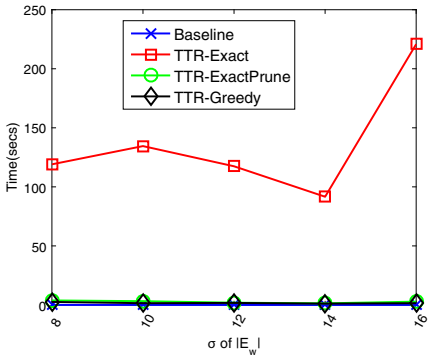

Baseline
TTR-Exact
TTR-ExactPrune
TTR-Greedy
Time(secs)
Memory(MB)
Utility
k
|W|
|E_t|
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
(j)
(k)
(l)



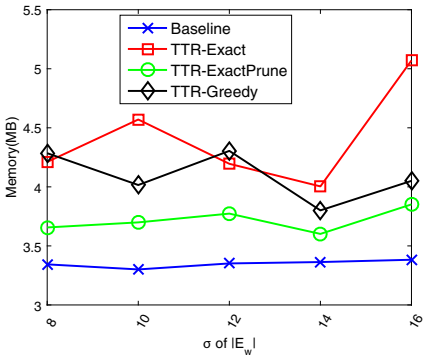
(a)

(b)

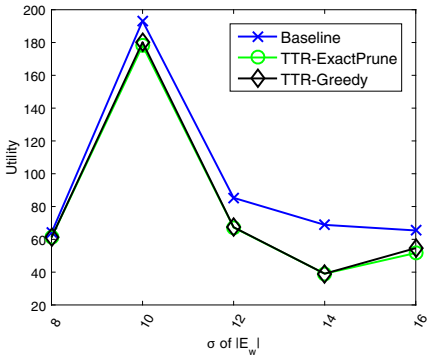
(c)



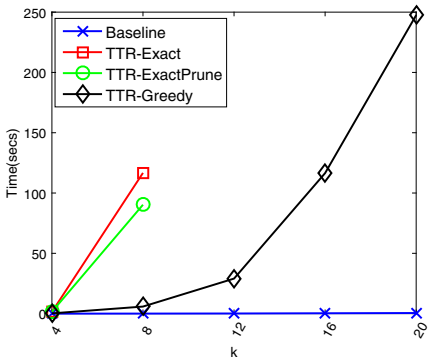
(d)



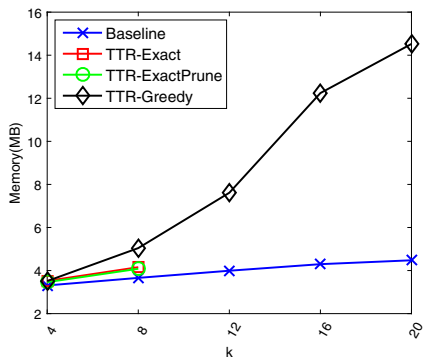
(e)



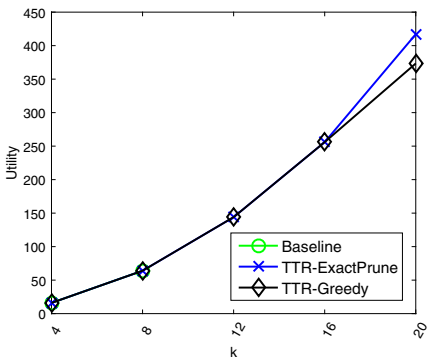
(f)



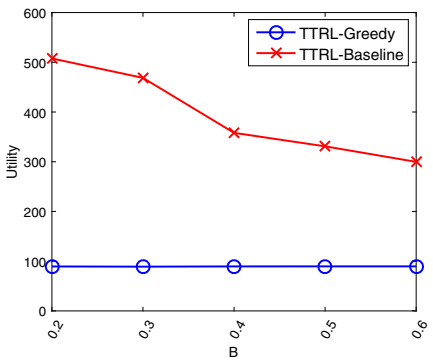
(g)



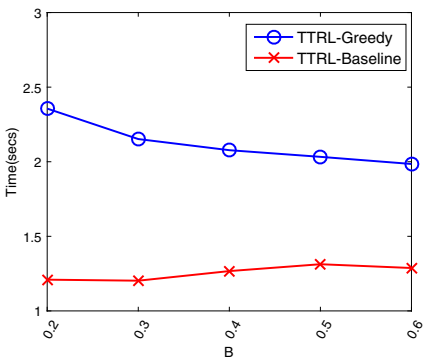
(h)



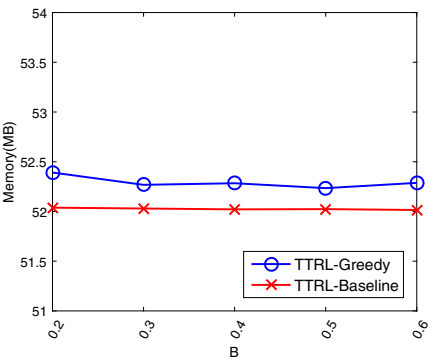
(i)



(j)



(k)



(l)

◀**Fig. 6** Results on varying k, $|W|$, E_t and B in the normal distribution. **a** Utility of varying k, **b** time of varying k, **c** memory of varying k, **d** utility of varying $|W|$, **e** utility of varying $|W|$, **f** utility of varying $|W|$, **g** utility of varying $|E_t|$, **h** time of varying $|E_t|$, **i** utility of varying $|E_t|$, **j** utility of varying B, **k** time of varying B and **l** memory of varying B

We can first observe that the utility value first increases as μ and σ increase and then drops when μ and σ further increase. The possible reason is that when μ and σ first increase, the skills of workers are more diverse and may not cover the requirements of the tasks and thus more workers are still needed. However, as μ and σ further increase, many workers can utilize their multiple skills and thus less workers are needed. As for running time, TTR-Exact is again very inefficient. Finally, for memory, TTR-ExactPrune is more efficient than TTR-Exact and TTR-Greedy.

Scalability The results are presented in Fig. 3g–i. Since the exact algorithms are not efficient enough, we only study the scalability of TTR-Greedy. We can see that the running time and memory consumption TTR-Greedy is still quite small when the scale of data is large.

Real Dataset The results on real dataset are shown in Fig. 4a–c, where we vary k. We can observe similar patterns as those in Fig. 2d–f. Notice that the exact algorithms are not efficient enough on the dataset, so no result of them when k is larger than 8 is presented.

Conclusion For utility, TTR-Greedy is nearly as good as the exact algorithms, and TTR-Greedy and the exact algorithms all perform better than the baseline algorithm do. As for running time, TTR-Exact is the most inefficient, while TTR-ExactPrune is much more efficient than TTR-Exact due to its pruning techniques but is still slower than TTR-Greedy.

5.3 Evaluation for Top$k$TRL

In this subsection, we test the performance of our proposed algorithms for the Top$k$TRL problem through varying different parameters. In particular, we extend the baseline algorithm to TTRL-Baseline. The TTRL-Baseline algorithm tries to find the leader for each team gained from the baseline algorithm and abandon the team without leader until there are k teams. We compare TTRL-Baseline with the greedy algorithm (Algorithm 4) called TTRL-Greedy in terms of total utility score, running time and memory cost.

We use the real dataset from gMission to generate the graph of social network. Specifically, the collaborative cost of worker w_1 and w_2 is calculated by $\frac{\mathrm{MAX_c}-\mathrm{c}(\mathrm{w_1},\mathrm{w_2})}{\mathrm{MAX}_c}$, where $c(w_1, w_2)$ is the number of the times that w_1 and w_2 have cooperated, and MAX_c is the maximal number of the times that any two workers have cooperated. For the synthetic dataset, we generate the graph where the weight of edges follows either normal or uniform distribution in the range of 0–1. According to the statistics of the real dataset, we set the mean value of the normal distribution as 0.4, and the standard deviation as 0.3. The experimental results are shown as follows.

Effect of Parameter k The results of varying k are presented in Fig. 5a–c when following the uniform distribution and Fig. 6a–c when following the normal distribution. We can observe that the utility of TTRL-Greedy is much smaller than that of TTRL-Baseline. And TTRL-Greedy spends more time and memory than TTRL-Baseline in both distributions.

Effect of Parameter $|W|$ The results of varying $|W|$ are presented in Fig. 5d–f when following the uniform distribution and Fig.6d–f when following the normal distribution. We can observe that the utility decreases as more efficient workers join, and the effect of $|W|$'s growth on the running time is not obvious. We can also find that the running time of the uniform distribution is larger than that of the normal distribution, and a possible reason is that the number of edges with low weight in the normal distribution is larger than that in the uniform distribution. As a result it takes more time to find the worker with low collaborative cost.

Effect of Parameter $|E_t|$ The results of varying E_t are presented in Fig. 5g–i when following the uniform distribution and Fig. 6g–i when following the normal distribution. We can observe that the utility, running time and memory increase as $|E_t|$ increases. The running time and memory of TTRL-Greedy increase when $|E_t|$ is greater than 12. However, considering that $|E_t|$ would not be too large the TTRL-Greedy algorithm is still efficient.

Effect of Parameter B The results of varying B are presented in Fig. 5j–l when following the uniform distribution and Fig. 6j–l when following the normal distribution. We can observe that the utility of TTRL-Greedy is not affected by the growth of the budget, because TTRL-Greedy has already tried to find the teams with low utility. However, in TTRL-Baseline the growth of budget relaxes the restrictions to select the workers, meaning that some better workers can be added. Meanwhile, in terms of utility, TTRL-Greedy always performs better. And for the running time and memory, TTRL-Greedy performs better when B is large in uniform distribution.

Conclusion TTRL-Greedy outperforms TTRL-Baseline significantly in terms of utility. Meanwhile the efficiency of the proposed framework is good, though sometimes it is less efficient than the baseline algorithm.

6 Related Work

In this section, we review related works from two categories, spatial crowdsourcing and team formation.

6.1 Spatial Crowdsourcing

Crowdsourcing has been widely studied in [1, 2, 17–21]. Most works on spatial crowdsourcing study the task assignment problem. Kazemi and Shahabi [6] and To et al. [10] aim to maximize the number of tasks that are assigned to workers. Furthermore, the conflict-aware spatial task assignment problems are studied [22–24]. Recently, the issue of online task assignment in dynamic spatial crowdsourcing scenarios is proposed [7]. Kazemi et al. [8] further studies the reliability of crowd workers based on [6]. To et al. [9] studies the location privacy protection problem for the workers. Kazemi et al. [8] studies the route planning problem for a crowd worker and tries to maximize the number of completed tasks. The corresponding online version of [8] is studied in [25]. Wang et al. [3] studies the entity resolution problem in crowdsourcing. And Franklin et al. [1] studies the problem of using crowd workers to answer queries which is difficult for machine. In addition, Liu et al. [2] proposes a crowdsourcing system to support the deployment of various crowdsourcing applications. Although the aforementioned works study the task allocation problem on spatial crowdsourcing, they always assume that spatial crowdsourcing tasks are simple micro-tasks and ignore that some real spatial crowdsourced tasks often need to be collaboratively completed by a team of crowd workers.

6.2 Team Formation Problem

Another closely related topic is the team formation problem [14], which aims to find the minimum cost team of experts according to skills and relationships of users in social networks. Anagnostopoulos et al. [26, 27] further studies the workload balance issue in the static and dynamic team formation problem. The capacity constraint of experts is also considered as an variant of the team formation problem in [15]. Moreover, the problems of discovering crowd experts in social media market are also studied [28, 29]. Rangapuram et al. [30] studies the team formation problem in the densest subgraphs. And Rahman et al. [31] tries to analyze the worker's skills through the record of participated tasks. Feng et al. [32] studies the variant problem to maximum the influence of the team. The above works only consider to find the minimum cost team, namely top-1 team, instead of top-k teams without free riders. In addition, we address the spatial scenarios rather than the social networks scenarios.

7 Conclusion

In this paper, we study a novel spatial crowdsourcing problem, called the *Top-k team recommendation in spatial crowdsourcing* (Top$k$TR) and its variant, called the Top$k$TRL problem. Then, we prove that the two proposed problems are NP-hard. To address the Top$k$TR problem, we design a two-level-based framework, which not only includes an exact algorithm with pruning techniques to get the exact solution but also seamlessly integrates an approximation algorithm to guarantee theoretical approximation ratio. Furthermore, the aforementioned framework can easily be extended to address the Top$k$TRL problem with the same approximation ratio. Finally, we conduct extensive experiments which verify the efficiency and effectiveness of the proposed approaches.

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

References

1. Franklin MJ, Kossmann D, Kraska T, Ramesh S, Xin R (2011) Crowddb: answering queries with crowdsourcing. In: SIGMOD, pp 61–72
2. Liu X, Lu M, Ooi BC, Shen Y, Wu S, Zhang M (2012) CDAS: a crowdsourcing data analytics system. PVLDB 5(10):1040–1051
3. Wang J, Kraska T, Franklin MJ, Feng J (2012) Crowder: crowdsourcing entity resolution. PVLDB 5(11):1483–1494
4. Tong Y, Cao CC, Chen L (2014) TCS: efficient topic discovery over crowd-oriented service data. In: SIGKDD. 861–870
5. Tong Y, Cao CC, Zhang CJ, Li Y, Chen L (2014) Crowdcleaner: data cleaning for multi-version data on the web via crowdsourcing. In: ICDE, pp 1182–1185
6. Kazemi L, Shahabi C (2012) Geocrowd: enabling query answering with spatial crowdsourcing. In: GIS, pp 189–198
7. Tong Y, She J, Ding B, Wang L, Chen L (2016) Online mobile micro-task allocation in spatial crowdsourcing. In: ICDE, pp 49–60
8. Kazemi L, Shahabi C, Chen L (2013) Geotrucrowd: trustworthy query answering with spatial crowdsourcing. In: GIS, pp 304–313

9. To H, Ghinita G, Shahabi C (2014) A framework for protecting worker location privacy in spatial crowdsourcing. PVLDB 7(10):919–930
10. To H, Shahabi C, Kazemi L (2015) A server-assigned spatial crowdsourcing framework. ACM Trans Spat Algorithm Syst 1(1):2
11. She J, Tong Y, Chen L, Cao CC (2016) Conflict-aware event-participant arrangement and its variant for online setting. IEEE Trans Knowl Data Eng 28(9):2281–2295
12. 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
13. Gao D, Tong Y, She J, Song T, Chen L, Xu K (2016) Top-k team recommendation in spatial crowdsourcing. In: WAIM, pp 735–746
14. Lappas T, Liu K, Terzi E (2009) Finding a team of experts in social networks. In: SIGKDD, pp 467–476
15. Majumder A, Datta S, Naidu K (2012) Capacitated team formation problem on social networks. In: SIGKDD, pp 1005–1013
16. Chen Z, Fu R, Zhao Z, Liu Z, Xia L, Chen L, Cheng P, Cao CC, Tong Y, Zhang CJ (2014) gmission: a general spatial crowdsourcing platform. PVLDB 7(13):1629–1632
17. Venetis P, Garcia-Molina H, Huang K, Polyzotis N (2012) Max algorithms in crowdsourcing environments. In: WWW, pp 989–998
18. Howe J (2009) Crowdsourcing: why the power of the crowd is driving the future of business. Three Rivers Press, New York, p 311
19. Thebault-Spieker J, Terveen LG, Hecht B (2015) Avoiding the south side and the suburbs: the geography of mobile crowdsourcing markets. In: CSCW, pp 265–275
20. Karger DR, Oh S, Shah D (2014) Budget-optimal task allocation for reliable crowdsourcing systems. Oper Res 62(1):1–24
21. Alfarrarjeh A, Emrich T, Shahabi C (2015) Scalable spatial crowdsourcing: a study of distributed algorithms. In: MDM, pp 134–144
22. She J, Tong Y, Chen L (2015) Utility-aware social event-participant planning. In: SIGMOD, pp 1629–1643
23. She J, Tong Y, Chen L, Cao CC (2015) Conflict-aware event-participant arrangement. In: ICDE, pp 735–746
24. Tong Y, She J, Meng R (2016) Bottleneck-aware arrangement over event-based social networks: the max–min approach. World Wide Web J 19(6):1151–1177
25. Li Y, Yiu M, Xu W (2015) Oriented online route recommendation for spatial crowdsourcing task workers. In: SSTD, pp 861–870
26. Anagnostopoulos A, Becchetti L, Castillo C, Gionis A, Leonardi S (2010) Power in unity: forming teams in large-scale community systems. In: CIKM, pp 599–608
27. Anagnostopoulos A, Becchetti L, Castillo C, Gionis A, Leonardi S (2012) Online team formation in social networks. In: WWW, pp 839–848
28. Cao CC, She J, Tong Y, Chen L (2012) Whom to ask? Jury selection for decision making tasks on micro-blog services. PVLDB 5(11):1495–1506
29. Cao CC, Tong Y, Chen L, Jagadish HV (2013) Wisemarket: a new paradigm for managing wisdom of online social users. In: SIGKDD, pp 455–463
30. Rangapuram SS, Bühler T, Hein M (2013) Towards realistic team formation in social networks based on densest subgraphs. In: WWW, pp 1077–1088
31. Rahman H, Thirumuruganathan S, Roy SB, Amer-Yahia S, Das G (2015) Worker skill estimation in team-based tasks. PVLDB 8(11):1142–1153
32. Feng K, Cong G, Bhowmick SS, Ma S (2014) In search of influential event organizers in online social networks. In: SIGMOD, pp 63–74