Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Friendship-aware task planning in mobile crowdsourcing*

Yuan LIANG, Wei-feng LV, Wen-jun WU‡, Ke XU

(*State Key Laboratory of Software Development Environment, School of Computer Science,*
Beihang University, Beijing 100191, China)

E-mail: liangyuan120@nlsde.buaa.edu.cn; lwf@nlsde.buaa.edu.cn; wwj@nlsde.buaa.edu.cn; kex@nlsde.buaa.edu.cn

Received Dec. 23, 2016; Revision accepted Jan. 9, 2017; Crosschecked Jan. 10, 2017

Abstract: Recently, crowdsourcing platforms have attracted a number of citizens to perform a variety of location-specific tasks. However, most existing approaches consider the arrangement of a set of tasks for a set of crowd workers, while few consider crowd workers arriving in a dynamic manner. Therefore, how to arrange suitable location-specific tasks to a set of crowd workers such that the crowd workers obtain maximum satisfaction when arriving sequentially represents a challenge. To address the limitation of existing approaches, we first identify a more general and useful model that considers not only the arrangement of a set of tasks to a set of crowd workers, but also all the dynamic arrivals of all crowd workers. Then, we present an effective crowd-task model which is applied to offline and online settings, respectively. To solve the problem in an offline setting, we first observe the characteristics of task planning (CTP) and devise a CTP algorithm to solve the problem. We also propose an effective greedy method and integrated simulated annealing (ISA) techniques to improve the algorithm performance. To solve the problem in an online setting, we develop a greedy algorithm for task planning. Finally, we verify the effectiveness and efficiency of the proposed solutions through extensive experiments using real and synthetic datasets.

Key words: Mobile crowdsourcing; Task planning; Greedy algorithms; Simulated annealing
http://dx.doi.org/10.1631/FITEE.1601860 **CLC number:** TP3

1 Introduction

Research on crowd intelligence has attracted much attention in the current artificial intelligence (AI) 2.0 era (Pan, 2016). Along with the rapid development of cloud computing and the mobile Internet (Tong *et al.*, 2015b; Cheng *et al.*, 2016), mobile crowdsourcing applications have become increasingly popular. Many web applications, such as Gigwalk, Uber, and Meetup, have been developed. These applications provide tasks that allow crowd workers to sign up online (using the Internet) and accomplish tasks offline (in the real world), and have attracted millions of crowd workers creating more than 230 000 million events per month (Musthag and Ganesan, 2013; Tong *et al.*, 2016b).

Current mobile crowdsourcing platforms simply list the tasks to be undertaken and allow crowd workers to choose the tasks they wish to accomplish. Unfortunately, it is time-consuming for crowd workers to check such a long task list and select their preferences. It would be far more convenient if the platform providers were able to make personalized suggestions for each crowd worker on the tasks in which they might be most interested. Imagine the following scenario. Amy, Bob, and Cathy are friends who are interested in programming. On a Friday evening, Amy logs into the website and finds a programming task that requires three skills: art, C++ language, and planning. She is good at art designing and would like to take part in this task. She also wants to accomplish this task with her two friends.

‡ Corresponding author

* Project supported by the National High-Tech R&D Program (863) of China (No. 2014AA015203)

ORCID: Wen-jun WU, http://orcid.org/0000-0003-0953-0115

In addition, Bob is good at C++ language and Cathy is good at planning. However, Bob does not want to travel a long distance to accomplish the task. The two other workers do not want to accomplish the task with strangers. A crowd-task system often encounters similar situations in which some crowd workers would like to choose a task that requires some given skills together and should also consider the travel cost between the locations of workers and tasks. Therefore, for task planning, it is necessary to consider the following factors: (1) the distance between crowd workers and tasks, (2) the interest/skill similarity between crowd workers and tasks, and (3) the friendships among all crowd workers (Li *et al.*, 2014; Armenatzoglou *et al.*, 2015; She *et al.*, 2015a; 2015b; 2016; Tong *et al.*, 2016c). Unfortunately, no existing works consider all the above factors.

In addition to resolving task planning by considering the above three factors, a crowd-task arrangement strategy should consider that crowd workers may dynamically register into tasks online so that the available worker quotas for the tasks change at different times. Imagine the following scenario. Amy would like to participate in a programming contest and signs up. When she signs up to the contest website, there are a lot of available spots. Later, Bob and Cathy also decide to participate in this contest with Amy. However, now the spots are taken up by others. As a result, they cannot go to the same contest together. Therefore, how to match newly arriving crowd workers with tasks such that the crowd sourcing system can offer its workers the maximized satisfaction considering the friendships between workers is an important problem. Unfortunately, no existing studies focus on this problem. Unlike the static scenarios in which the crowd sourcing platform always knows all the crowd workers and tasks in the platform, in dynamic scenarios, the platform can never have complete information of the crowd workers. Thus, existing methods developed for static scenarios cannot be used to address dynamic scenarios.

To further illustrate the motivation, we present a small example as follows:

Example 1 Suppose that there are three tasks (v_1, v_2, v_3) and six workers $(u_1, u_2, \ldots, u_6)$ in a crowdsourcing platform. The locations of workers/tasks are represented by their longitudes and latitudes. We can calculate the distances between workers and tasks using the Euclidean distance based on the coordinates of workers and tasks. The distances between all the workers and tasks are shown in Table 1. When each worker registers in the crowdsourcing platform, they are required to choose some labels that represent the types of tasks they are interested in, such as football games and hiking. For each task, when it is created, the person creating this task is required to choose labels for this task to describe the category that it belongs to, such as lecture, concert, and film. Then, we can use the similarity between the worker-preference labels and the task labels to represent the interest of each worker in each task. The calculation method for this can be found in She *et al.* (2015a). The similarity of workers and tasks in this example is also shown in Table 1. Furthermore, all crowd workers form a social graph that represents the friendships among workers (Fig. 1). Finally, each task has a capacity, which is the maximum number of workers that are allowed to accomplish the task. In this example, the capacities of tasks v_1, v_2, and v_3 are 2, 4, and 3, respectively, as shown in the parentheses attached to each task in Table 1. The utility function, which evaluates the satisfaction (or happiness) of workers, should consider three factors: the distances between workers and tasks (the smaller the better), the similarities between workers and tasks (the larger the better), and the friendships among workers (friends are better than strangers). A feasible task arrangement is $\langle u_1, v_3\rangle$, $\langle u_2, v_1\rangle$, $\langle u_3, v_2\rangle$, $\langle u_4, v_1\rangle$, $\langle u_5, v_3\rangle$, and $\langle u_6, v_3\rangle$.

Table 1 The distance and similarity between workers and tasks

Worker	Distance			Similarity		
	v_1 (2)	v_2 (4)	v_3 (3)	v_1 (2)	v_2 (4)	v_3 (3)
u_1	429.003	324.232	248.558	0.48	0.52	0.55
u_2	385.754	780.134	653.009	0.69	0.66	0.58
u_3	588.770	88.393	181.986	0.42	0.50	0.44
u_4	635.756	781.092	700.514	0.59	0.39	0.46
u_5	261.343	313.735	165.946	0.62	0.68	0.64
u_6	353.734	265.025	145.121	0.39	0.61	0.43

The numeral in the parentheses represents the capacity of the task

For the online setting, our example can be seen as a bipartite graph $G = (U, V)$, in which the left vertices are workers and the right vertices are tasks. In this example, suppose that the order of arrival of the workers is u_6, u_5, u_4, u_3, u_2, and u_1. When u_6 arrives, he/she chooses to attend v_2 because it is the closest and most desirable task. When u_5

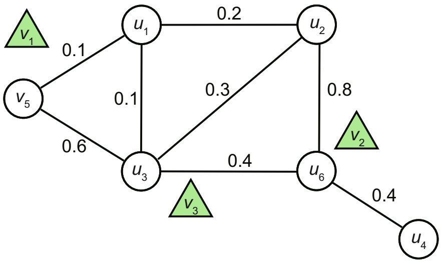

Fig. 1 Social graph of all workers

arrives, because he/she has no friends that have been matched to any task, he/she chooses to attend v_3 by considering the distance and similarity to each task. Then, u_4 arrives. He/She has a friend, u_6, who has already been matched to a task. Therefore, he/she should consider three factors: the distance, the similarity, and the friendship. Following the above steps, a feasible arrangement is $\langle u_1, v_3\rangle$, $\langle u_2, v_1\rangle$, $\langle u_3, v_2\rangle$, $\langle u_4, v_1\rangle$, $\langle u_5, v_3\rangle$, and $\langle u_6, v_2\rangle$.

To summarize, in this paper we propose the following:

1. We propose an offline planning problem that considers all factors, i.e., distance, similarity, and friendship.

2. We further propose an online planning problem based on the offline version.

3. We propose greedy algorithms to solve the offline and online arrangement problems, respectively.

4. We conduct a comprehensive experimental study on both synthetic and real datasets from Meetup.

2 Problem statement

We first introduce some concepts and then formally define two versions of task assignment for the offline and online scenarios. In our problem, we assume that there is a crowd-sourcing platform that people can use to publish the task, and some definitions are similar to those in Tone *et al.* (2016a; 2016b) and She *et al.* (2016).

Definition 1 (Crowd worker) A worker is defined as $u(x_u, y_u)$, where x_u and y_u represent the longitude and latitude of the worker, respectively. Each worker has a set of attributes (e.g., a_u for worker u).

Definition 2 (Task) A task is defined as $v(x_v, y_v, \delta_v)$, where x_v represents the longitude, y_v represents the latitude, and δ_v represents the capacity of task v. Each task also has a set of attributes (e.g., a_v for task v).

Basically, we consider three factors during task assignment: the distances between workers and tasks, the similarity between workers and tasks, and the friendships among all workers. Consequently, we provide the following definitions:

Definition 3 (Distance) Because both workers and tasks have physical locations, we use Euclidean distance $d = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}$ to compute the distance between a worker and a task.

Definition 4 (Similarity) Each worker has attributes that represent the attraction of task v for worker u. Each task has attributes as well. We compute the attribute similarity between workers and tasks as follows:

$$s(u,v) = \frac{\sum_{i=1}^{n} a_u(i) a_v(i)}{\sqrt{\sum_{i=1}^{n} a_u^2(i)} \sqrt{\sum_{i=1}^{n} a_v^2(i)}} \in [0,1],$$

where $a_u(i)$ denotes the ith attribute of workers, $a_v(i)$ the ith attribute of tasks, and n the number of attributes of workers and tasks.

Definition 5 (Social graph) Let $G = (U, E, W)$ represent the social graph, where U denotes the set of workers, W denotes the set of edge weights (i.e., the friendships among workers), and E denotes the set of edges (i.e., the social connections between workers).

Definition 6 (Offline planning) We are given a set of workers U, each of whom has a set of attributes a_u, a longitude x_u, and a latitude y_u. We also have a set of tasks V, each of which has a capacity δ_v, a set of attributes a_v, and a longitude x_v and latitude y_v. Using the Euclidean distance, a similarity function, and a social graph, we need to find an arrangement between workers and tasks that maximizes the total utility $\mu(U, V, \alpha, \beta, \gamma)$ and no task exceeds its capacity, δ_v. The total utility is defined as

$$\mu(U, V, \alpha, \beta, \gamma) = \frac{\alpha}{\sum\limits_{i=1}^{n} d(u_i, v_i)} + \beta \sum_{u \in U} s(u,v) + \gamma \sum_{\substack{e=(u_i,u_j)\in V,\\ i \neq j}} w(u_i, u_j),$$

where $w(u_i, u_j)$ denotes the social relationship between u_i and u_j, $\alpha, \beta, \gamma \in (0,1)$ are used to adjust the relative importance of the three factors under the condition $\alpha + \beta + \gamma = 1$. The first term of $\mu(\cdot)$ evaluates the sum of all the distances between each worker and the corresponding assigned tasks, the second term of $\mu(\cdot)$ evaluates the innate affinity between workers and tasks, and the last term of $\mu(\cdot)$ evaluates the friendships among workers.

Definition 7 (Online planning) We are given a set of tasks V, each of which has a capacity δ_v, attributes a_v, longitude x_v, and latitude y_v, and a set of workers U, each of whom arrives one by one and has a set of attributes a_u, a longitude x_u and a latitude y_u. Using the Euclidean distance formula, a similarity function, and a social graph, online planning is to find an arrangement between workers and tasks with maximized total utility $\mu(U, V, \alpha, \beta, \gamma)$, such that:

1. The three constraints of offline planning are satisfied.

2. The planning for a newly arriving worker u must be completed before the next worker appears, and it must be irrevocable (Tong *et al.*, 2016a; 2016b; She *et al.*, 2016).

3 Solution to the offline setting

In this section, we present a set of solutions to the offline planning problem.

3.1 Characteristic of task planning

As defined in offline planning, friendship plays an important role in crowd workers' accomplishment of tasks. Workers with closer friendship tend to work together to accomplish a task. In contrast, workers in different tasks are usually strangers. Based on this characteristic of task assignment, the characteristic of task planning (CTP) algorithm is proposed as follows.

Given a set of workers and a set of tasks, we want to find a collaboration such that each worker gains maximum satisfaction. We first assign pairs of workers and tasks to a heap $H = \langle u, v, g\rangle$, which represents the potential arrangements of pairs of workers and tasks, by considering both the distances between workers and tasks and their intrinsic similarity. H is ordered by the non-decreasing potential gain g. If we do not consider the social graph, the potential gain is defined as

$$g(u, v|\varnothing) = \frac{\alpha}{d(u, v)} + (1 - \alpha)s(u, v).$$

In contrast, if we consider the social graph, the potential gain is defined as

$$g(u, v|S_v) = \frac{\alpha}{d(u, v)} + \beta s(u, v) + \gamma \sum_{v_{u'}=v_u} w(u, u'),$$

where S_v represents the set of workers assigned to task v. Then, we extract the worker-task pair with the smallest $g(u, v|\varnothing)$ from heap H. If the task has not exceeded its capacity, we will assign worker u to task v. Let $M(u)$ denote the arrangement of worker u; then, if the neighbors of worker u (i.e., u') have not been assigned, we update $g(u, v|S_v)$ based on heap H. Finally, we extract worker u' with the smallest potential gain and assign u' to task v that satisfies $|S_v| < \delta_v$. This process can be repeated as needed until there are no more available tasks or until all workers have been assigned to tasks.

Algorithm 1 illustrates the CTP procedure. Here, $M(u) \leftarrow \varnothing$ denotes that worker u is not assigned. Lines 1–3 compute the potential arrangement utility according to the pairs of workers and tasks and put them into heap H. Lines 5–14 first extract the worker-task pair with the smallest potential gain that contains worker u, task v, and gain value g and then assign worker u to task v. Then, we compute the potential arrangement utility of worker u's neighbors and update H. Finally, we find the minimum potential gain that contains worker u's neighbors, task v, and the gain utility. Finally, we assign worker u's neighbors to task v.

Example 2 Here is the process of running our CTP algorithm on Example 1. All distances are normalized into range $[0, 1]$. Then, we calculate $g(u, v|\varnothing)$ for all pairs of workers and tasks and find that H has 18 potential gains and 18 pairs of workers and tasks, i.e., $\langle u_1, v_1, 1.33\rangle$, $\langle u_1, v_2, 1.11\rangle$, $\langle u_1, v_3, 0.97\rangle$, $\langle u_2, v_1, 1.01\rangle$, $\langle u_2, v_2, 1.53\rangle$, $\langle u_2, v_3, 1.45\rangle$, $\langle u_3, v_1, 1.75\rangle$, $\langle u_3, v_2, 0.82\rangle$, $\langle u_3, v_3, 1.04\rangle$, $\langle u_4, v_1, 1.41\rangle$,

Algorithm 1 Characteristic of task planning (CTP)

```
Initialize: heap $H$, $M(u) \leftarrow \varnothing$, $\forall u$, and $S_v \leftarrow \varnothing$, $\forall v$.
 1: for all $(u, v) \in U \times V$ s.t. $\alpha d(u, v) + \frac{1-\alpha}{s(u, v)} > 0$ do
 2:     Insert $\{u, v, g(u, v|\varnothing)\}$ into $H$
 3: end for
 4: Heapify $H$
 5: while $H \neq \varnothing$ do
 6:     Extract the worker-task pair with the minimum
        potential gain ($\{u, v, g(u, v|S_v)\}$) from $H$
 7:     if $|S_v| < \delta_v$ and $M(u) \leftarrow \varnothing$ then
 8:         $S_v \leftarrow S_v \cup \{u\}$
 9:         for all $u'$: $w(u, u') > 0$ and $M(u') = \varnothing$ do
10:             Update $\{u', v, g(u', v|S_v)\}$ into $H$
11:         end for
12:         Heapify $H$
13:     end if
14: end while
15: return final arrangement and the total utility
```

$\langle u_4, v_2, 1.91\rangle$, $\langle u_4, v_3, 1.66\rangle$, $\langle u_5, v_1, 0.91\rangle$, $\langle u_5, v_2, 0.93\rangle$, $\langle u_5, v_3, 0.77\rangle$, $\langle u_6, v_1, 1.36\rangle$, $\langle u_6, v_2, 0.83\rangle$, and $\langle u_6, v_3, 1.02\rangle$. From the above results, we find that 0.77 is the smallest value of all the potential gains and that the capacity of v_3 is 3; therefore, u_5 can be assigned to task v_3. Note that u_5 has two friends: u_1 and u_3. Next, we update heap H for all workers. This process is repeated until each worker obtains the minimum potential utility. The final arrangement is $\langle u_1, v_3\rangle$, $\langle u_2, v_2\rangle$, $\langle u_3, v_3\rangle$, $\langle u_4, v_1\rangle$, $\langle u_5, v_3\rangle$, and $\langle u_6, v_1\rangle$, and the final total utility is 1.49.

We analyze the computation complexity of the CTP algorithm as follows: The CTP algorithm takes at most $O(|U||V|)$ time to initialize heap H and insert the utilities of all worker-task pairs into the heap. In the following iterations, at most $|U||V|$ worker-task pairs are extracted from H, but only $|U|$ pairs are inserted into the arrangement. Along with each insertion operation, at most d elements in H are updated, leading to $O(D\log(|U||V|))$ swapping-element operations in H (D is the degree of $u \in U$). The above analysis indicates that the final time complexity is $O(|U||V| + |U|D_{\max}\log(|U||V|))$.

3.2 Greedy offline planning

The CTP algorithm observes only the characteristics of collaboration to complete the task; many workers do not attain better satisfaction (happiness). To solve this problem, we propose a greedy algorithm, which assigns workers to tasks by considering three factors: the distance between workers and tasks, the similarities between workers and tasks, and the friendship among workers. The details of the greedy algorithm are as follows.

Let H contain a tuple $\langle u, v, g\rangle$ representing the potential arrangements of pairs of workers and tasks. If the potential arrangement utility does not consider the friendship among workers, it is defined as

$$g(u, v|\varnothing) = \frac{\alpha}{d(u, v)} + (1-\alpha)s(u, v).$$

If the potential arrangement utility considers the friendship among workers, it is defined as

$$g(u, v|S_v) = \frac{\alpha}{d(u, v)} + \beta s(u, v) + \gamma \sum_{v_u = v_{u'}} w(u, u').$$

First, we extract the pair with the largest $g(u, v|\varnothing)$ from heap H. If the task has not exceeded its capacity, we assign worker u to task v. If the neighbors of worker u (i.e., u') have not been assigned, we update $g(u, v|S_v)$ based on heap H. Finally, we extract the largest potential arrangement utility of pairs of worker u and task v that satisfy $|S_v| < \delta_v$. This process can be repeated as needed until either all workers are assigned or there are no more available tasks.

Algorithm 2 illustrates the procedure of greedy offline planing. Lines 1–3 compute the potential gain according to pairs of workers and tasks and place them into heap H. Lines 4–12 first extract the pair with the largest potential arrangement utility that contains worker u, task v, and gain value g, and then assign worker u to task v. Next, we compute the potential gain of worker u's neighbors and update H. Finally, we find the maximum potential arrangement utility that contains the neighbors of worker u, task v, and the gain utility, and assign worker u's neighbors to task v.

Algorithm 2 Greedy offline planning

```
Initialize: heap $H$, $M(u) \leftarrow \varnothing$, $\forall u$, and $S_v \leftarrow \varnothing$, $\forall v$.
1: for all $(u, v) \in U \times V$ s.t. $\frac{\alpha}{d(u, v)} + (1-\alpha)\cdot s(u, v) > 0$ do
2:     Insert $\{u, v, g(u, v|\varnothing)\}$ into $H$
3: end for
4: while $H \neq \varnothing$ do
5:     Extract the worker-task pair with the largest potential gain ($\{u, v, g(u, v|S_v)\}$) from $H$
6:     if $|S_v| < \delta_v$ and $M(u) = \varnothing$ then
7:         $S_v \leftarrow S_v \cup \{u\}$
8:         for all $u'$: $w(u, u') > 0$ and $M(u') = \varnothing$ do
9:             Update $\{u', v, g(u', v|S_v)\}$ into $H$
10:        end for
11:    end if
12: end while
13: return final matching and the total utility
```

Example 3 Here is the process of running our greedy offline planning on Example 1. All distances are normalized into range $[0, 1]$ where, for each division, the maximum possible distance is $d_{\max}$. Then, we compute $g(u, v|\varnothing)$ and find that H has 18 potential gains and 18 pairs of workers and tasks, i.e., $\langle u_1, v_1, 0.67\rangle$, $\langle u_1, v_2, 0.79\rangle$, $\langle u_1, v_3, 0.91\rangle$, $\langle u_2, v_1, 0.92\rangle$, $\langle u_2, v_2, 0.78\rangle$, $\langle u_2, v_3, 0.72\rangle$, $\langle u_3, v_1, 0.57\rangle$, $\langle u_3, v_2, 1.50\rangle$, $\langle u_3, v_3, 0.93\rangle$, $\langle u_4, v_1, 0.73\rangle$, $\langle u_4, v_2, 0.50\rangle$, $\langle u_4, v_3, 0.59\rangle$, $\langle u_5, v_1, 0.93\rangle$, $\langle u_5, v_2, 0.96\rangle$, $\langle u_5, v_3, 1.08\rangle$, $\langle u_6, v_1, 0.60\rangle$, $\langle u_6, v_2, 0.94\rangle$, and $\langle u_6, v_3, 1.04\rangle$. From the above results, we find that 1.50 is the largest obtainable potential gain and that the capacity of v_2 is 4; therefore, u_3 can be assigned to v_2. Note that u_3 has four friends u_1, u_2, u_5, and u_6. We update heap H for u_3's neighbors, which results in $\langle u_1, v_2, 0.89\rangle$, $\langle u_2, v_2, 1.08\rangle$, $\langle u_5, v_2, 1.56\rangle$, and $\langle u_6, v_2, 1.34\rangle$ in H, and delete $\langle u_3, v_1, 0.57\rangle$ and $\langle u_3, v_3, 0.93\rangle$ from H. Now, 1.56 is the largest obtainable gain in this step; therefore, we assign u_5 to v_2, which satisfies $S_{v_2} \leq 4$, and update heap H again. Now, we find that u_5's neighbors are u_1 and u_3, but u_3 has already been assigned to a task. Therefore, we need to consider only u_1. In this step, we obtain $\langle u_1, v_2, 0.99\rangle$ in heap H. Now, we find that 1.34 is the largest gain and assign u_6 to v_1. However, u_6 has two neighbors, u_2 and u_4, who have not yet been matched to any task. Therefore, we obtain only $\langle u_2, v_1, 2.18\rangle$ and $\langle u_4, v_2, 0.90\rangle$ in heap H. Then, we find that 2.18 is the largest gain in this step; therefore, we assign u_2 to v_1, which satisfies the constraint that the capacity of $|S_{v_2}|$ is exactly 4. Note that u_2 has a friend, u_1, who has not yet been matched. Thus, we update $\langle u_1, v_2, 1.19\rangle$ in heap H. Finally, we would like to assign u_4 to v_2, which provides the maximum potential gain. However, the capacity of $|S_{v_2}|$ is 4 and assigning u_4 to v_2 would violate the constraint $S_{v_2} \leq 4$. Therefore, we must choose the second largest gain, 0.91, and assign u_4 to v_3. The final arrangement is $\langle u_1, v_2\rangle$, $\langle u_2, v_1\rangle$, $\langle u_3, v_2\rangle$, $\langle u_4, v_3\rangle$, $\langle u_5, v_2\rangle$, and $\langle u_6, v_1\rangle$, and the final total utility is 1.61.

Similar to the CTP algorithm, the worst-case time complexity of the greedy offline planning algorithm is $O(|U||V| + |U|D_{\max}\log(|U||V|))$.

3.3 Improved simulated annealing

The simple greedy algorithm falls easily into local optima. In this section, to address this limitation, we propose a hybrid heuristic to optimize task assignment. In addition, by combining the greedy algorithm with another heuristic algorithm, we can obtain a better solution. Therefore, we propose the improved simulated annealing (ISA) approach to solve our problem. Simulated annealing is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a metaheuristic for approximating global optimization over a large search space (Kirkpatrick *et al.*, 1987).

For crowd-tasking, we first set constant R, temperature T, and randomly assign a set of workers to a set of tasks satisfying $|S_v| \leq \delta_v$. The total utility of this current arrangement is $\text{old}_f = f_0$. As the temperature decreases, we randomly choose a worker u and suppose that u is assigned to v_i in this step. Next, we randomly choose a task v_j and assign u to v_j such that $i \neq j$ and $|S_{v_j}| < \delta_{v_j}$. The total utility of this arrangement is new_f. Let $\Delta f = \text{new}_f - \text{old}_f$. If $\Delta f \geq 0$, we can assign u to v_j with probability $p = 1$; if $\Delta f \leq 0$, we can assign u to v_j with probability $p = \exp(-|\Delta f|/(RT))$. When we find that $\Delta f \leq 0$ for $|T/2|$ consecutive times, we increase the temperature until a $\Delta f \geq 0$ is found that satisfies $|S_v| \leq \delta_v$. Finally, we reduce the temperature until it decreases to zero, at which point the process stops. Note that this approach to solving task assignment does not guarantee an optimal solution. Although we can find an optimal solution within the solutions space, we cannot guarantee that no better solutions exist. Therefore, to obtain better results, we consider the total utility, which is the maximum value of the entire solutions space. More details are shown in Algorithm 3.

The details of each iteration are as follows. Let the worker-task pair $\langle u, v_i\rangle$ contain worker u and task v_i in the current iteration. If v_j is another task and v_j satisfies $|S_{v_j}| < \delta_{v_j}$, assign u to v_i in the current step (i.e., $u \in S_{v_i}$). We then attempt to change the arrangement of u by assigning u to v_j (i.e., $u \in S_{v_j}$). Otherwise, other workers cannot change their current arrangements. More specifically, let M_i denote the ith arrangement and M_j represent the jth arrangement. If $|S_{v_j}|$ can accommodate an additional worker (i.e., $M_i - (u, v_i) = M_j - (u, v_j)$), we can compute the total utility of M_i and M_j. For each u in U, we know that the utility of arrangement M_i is equal to the utility of arrangement M_j. At each iteration, we change one worker in an arrangement. This process can be repeated for each temperature decrease.

Algorithm 3 illustrates the procedure of ISA. Line 1 randomly assigns a set of workers to a set of tasks that satisfy $|S_v| \leq \delta_v$. Note that the current utility value is f_0. Line 5 randomly chooses a worker u, where we suppose that u is matched to v_i. In addition, we randomly choose a task v_j in which u is matched to v_j such that $i \neq j$ and $|S_{v_j}| < \delta_{v_j}$. The total utility of this matching is new_f. Lines 6–10 compare the current solution and the neighboring solution; we choose the neighboring solution with a

Algorithm 3 Improved simulated annealing (ISA)

```
Initialize: $S_v \leftarrow \varnothing$, $\forall v$, $f$_increasing_count $= 0$, $R$,
    $T_0$, $\Delta_T$, $n$.
 1: Randomly assign all workers to tasks satisfying
    $|S_v| \leq \delta_v$ with total utility $f_0$.
 2: $\mathrm{new}_f = f_0,\ T = T_0$
 3: while $T \geq 0$ do
 4:   $\mathrm{old}_f = \mathrm{new}_f$
 5:   Randomly choose a worker $u$, randomly change its
      assigned task to an available task $v_j$, and obtain
      a new total utility $\mathrm{new}_f$ with $\Delta f = \mathrm{new}_f - \mathrm{old}_f$
 6:   if $\Delta f \geq 0$ then
 7:     $u$ with $p = 1$ matched to $v_j$
 8:   else
 9:     $u$ with $p = \exp\left(-\frac{|\Delta f|}{RT}\right)$ matched to $v_j$
10:   end if
11:   if $\Delta f$ is positive for $n$ consecutive times then
12:     while $\mathrm{new}_f - \mathrm{old}_f < 0$ do
13:       $T = T + \Delta T$, $\mathrm{old}_f = \mathrm{new}_f$
14:       Randomly choose a worker $u$, randomly
          change its assigned task to an available task
          $v_j$, and obtain a new total utility $\mathrm{new}_f$ with
          $\Delta f = \mathrm{new}_f - \mathrm{old}_f$
15:     end while
16:   end if
17:   $T = T - \Delta T$
18: end while
19: return arrangement of workers and tasks
```

certain probability. In lines 11–16, if $\Delta f \leq 0$ for $|T/2|$ consecutive times, we increase the temperature until $\Delta f \geq 0$. Finally, when the temperature decreases to zero, we select the maximum total utility of all the solutions.

Example 4 To run our ISA algorithm for Example 1, we randomly assign six workers to three tasks. The current arrangement is $\langle u_1, v_2\rangle$, $\langle u_2, v_1\rangle$, $\langle u_3, v_2\rangle$, $\langle u_4, v_2\rangle$, $\langle u_5, v_1\rangle$, and $\langle u_6, v_3\rangle$, which satisfies $|S_v| < \delta_v$, and the current arrangement utility is 1.18. Then, we randomly select a worker. Suppose that the worker selected is u_3 and that the arrangement of the current step is v_2. We change the arrangement of u_3 from v_2 to v_3, but the other arrangements unchanged. The total utility of this step is 1.16. Because $1.16 < 1.18$, u_3 is assigned to v_3 with a probability $p = 1$, and u_3 stays in v_2 with a probability $p = \exp(-|\Delta f|/(RT))$. Therefore, the current arrangement is $\langle u_1, v_2\rangle$, $\langle u_2, v_1\rangle$, $\langle u_3, v_2\rangle$, $\langle u_4, v_2\rangle$, $\langle u_5, v_1\rangle$, and $\langle u_6, v_3\rangle$. Then, we randomly select another worker u_2. The arrangement of the current step is v_1, and we change the current arrangement of u_2 from v_1 to v_2, which satisfies $|S_v| < \delta_v$. In this step, the arrangement is $\langle u_1, v_2\rangle$, $\langle u_2, v_2\rangle$, $\langle u_3, v_3\rangle$, $\langle u_4, v_2\rangle$, $\langle u_5, v_1\rangle$, and $\langle u_6, v_3\rangle$, and the total utility is 1.13. This process can be repeated as the temperature decreases until it reaches zero. Finally, the arrangement is $\langle u_1, v_3\rangle$, $\langle u_2, v_2\rangle$, $\langle u_3, v_2\rangle$, $\langle u_4, v_1\rangle$, $\langle u_5, v_2\rangle$, and $\langle u_6, v_2\rangle$, and the total utility is 1.90.

Here, we analyze the time complexity of ISA. In the initialization step, a set of workers are randomly assigned to a set of tasks, which takes $O(|U|)$ time. Due to the possible increment of the temperature in the algorithm, there are at least $T_0/\Delta T$ iterations in the temperature decrement procedure. Therefore, the overall time complexity of ISA is at least $O(|U| + T_0/\Delta T)$.

4 Greedy algorithm for online planning

In this section, we present a solution to the online scenario for task assignment. Note that the solution to the offline scenario cannot be used to solve the online setting because we do not have full information about the workers in the online setting. In the offline setting, we have full information of crowd workers and tasks. However, in the online scenario, the order in which workers appear cannot be known in advance because workers arrive randomly.

To solve the problem in the online setting, we propose a greedy solution (OnlineGreedy) in which workers arrive sequentially and can arrive in a random order. The input to this problem is regarded as a bipartite graph $G = (U, V)$, in which the vertices in U arrive in a random order but in which the vertices in V are fixed. When a worker arrives, that worker is matched to a task as soon as possible. Such a decision, once made, is irrevocable. We first let $N(u)$ be the set of u's neighbors who have been matched to tasks. When a new worker u arrives and has no friends matched to any task, then we compute the potential arrangement utility

$$g(u, v) = \frac{\alpha}{d(u, v)} + (1 - \alpha)s(u, v)$$

and find the maximum $g(u, v)$. If this task v satisfies $|S_v| < \delta_v$, we assign the newly arriving worker u to task v. However, when the newly arriving worker u has several friends who have already been matched to tasks, we then compute the potential arrangement utility

$$g(u, u', v) = \frac{\alpha}{d(u, v)} + \beta s(u, v) + \gamma w(u, u')$$

and assign the newly arriving worker u to a task v such that the arrangement obtains the maximum $g(u, u', v)$ and that task v satisfies $|S_v| < \delta_v$. This process can be repeated as needed until there are insufficient available tasks or until all crowd workers have been assigned.

In the online planning case, from a practical perspective, if many workers who have not attended any tasks arrive together and there are no available tasks to be assigned, these workers may be assigned to new large-capacity tasks such as art exhibitions or theatrical shows.

Algorithm 4 illustrates the procedure of the greedy algorithm for online planning. In lines 1–2, as each worker u arrives, we let $N(u)$ represent the set of u's neighbors that have been matched to task v satisfying $|S_v| < \delta_v$. In lines 3–9, we determine whether the arriving workers have friends that have been matched to tasks. When $N(u) = \varnothing$, we compute potential gain $g(u, v)$ and select the maximum potential gain that contains a pair of worker and task. Next, we assign this worker to a task that satisfies $|S_v| < \delta_v$. In lines 11–15, if $N(u) \neq \varnothing$, we assume that u' is a friend of the newly arriving worker u and that u' is participating in task v; consequently, we compute $g(u, u', v)$. Then, we select the maximum $g(u, u', v)$ in the current computation. If the current task v satisfies $|S_v| < \delta$, we assign the newly arriving u to v.

Example 5 Here, we return to Example 1, but in an online scenario. We assume that the order of arriving workers is u_5, u_6, u_2, u_4, u_3, and u_1; i.e., the first arriving worker is u_5. Then we find that assigning u_5 to v_3 can obtain the maximum potential gain. The next arriving worker is u_6, who has no friends that have been matched to tasks; therefore, we compute the potential gain only according to $g(u, v)$ and assign u_6 to v_3. The next arriving worker is u_2, who also has no friends that have been matched to tasks. We find that assigning u_2 to v_1 can obtain the maximum potential gain. The next arriving worker is u_4, whose friend u_6 has already been matched to a task. We compute the potential gain $g(u, u', v)$ and assign u_4 to v_3. When u_3 arrives, he/she has three friends, u_2, u_5, and u_6, who have been matched to tasks. Then, we compute the potential gain $g(u, u', v)$ and assign u_3 to v_3 to obtain the maximum potential gain. However, the capacity of v_3 is 3, and that task already has three workers: u_5, u_6, and u_4. Therefore, u_3 cannot be assigned to v_3; instead, we can assign u_3 to v_2 such that he/she receives the maximum satisfaction. When u_1 arrives, he/she also has three friends (u_2, u_3, and u_5) who have been assigned to tasks; thus, u_1 is assigned to v_1. Therefore, the final arrangement is $\langle u_1, v_1\rangle$, $\langle u_2, v_1\rangle$, $\langle u_3, v_2\rangle$, $\langle u_4, u_3\rangle$, $\langle u_5, v_3\rangle$, and $\langle u_6, v_3\rangle$, and the total utility is 1.27.

Algorithm 4 OnlineGreedy

```
Initialize: $S_v \leftarrow \varnothing$, $\forall v$.
 1: for arrival of vertex $u$ of $U$ do
 2:   Let $N(u)$ be the set of $u$'s neighbors who have
      been matched to tasks $v$ satisfying $|S_v| < \delta_v$
 3:   if $N(u) = \varnothing$ then
 4:     for each $v \in V$ do
 5:       if $|S_v| < \delta_v$ then
 6:         Calculate $\{u, v, g(u, v)\}$
 7:       end if
 8:     end for
 9:     Assign worker $u$ to task $v$ with the maximum
        $g(u, v)$
10:   else
11:     for each $u' \in N(u)$ do
12:       $v = V(u')$   // $v$ is the task assigned to $u'$
13:       Calculate $\{u, v, g(u, u', v)\}$
14:     end for
15:     Assign worker $u$ to task $v$ to obtain the maxi-
        mum $g(u, u', v)$
16:   end if
17: end for
18: return final arrangement and the total utility
```

During each new worker's arrival, the online planning algorithm first chooses one of the two candidate loops to follow. This selection process takes $O(D)$ time (D is the degree of the new arrival). Then, the algorithm will take $O(|V|)$ time to address the first loop or $O(D)$ time to address the second loop. All these operations contribute to a time complexity of $O(|U|(D_{\max} + |V|))$.

5 Experiments

5.1 Experimental setup

In this section, we describe the experiment setup for evaluating our proposed algorithms. We use both real and synthetic datasets for the experiments.

We used the Meetup dataset (Liu *et al.*, 2012) as the real dataset. In the Meetup dataset, each worker is associated with some tags and a location. The tasks in this dataset are not explicitly associated with tags, but we used the tags of the group that created the task as the tags of the task itself. We also used preprocessed datasets from She *et al.* (2015b). Similar to She *et al.* (2015b), we used three datasets from VA, Auckland, and Singapore, which consist of 225 tasks and 2012 workers, 37 tasks and 569 workers, and 87 tasks and 1500 workers, respectively. Because the task capacities were not provided in the datasets, we generated capacities for the tasks by following normal and uniform distributions.

For the synthetic data, we generated attribute values and locations following a normal distribution and generated the task capacities by following normal and uniform distributions. The statistics and configuration of the synthetic data are listed in Table 2. Default settings are denoted in bold font.

Table 2 Synthetic dataset settings

Factor	Setting
$\|V\|$	10, 20, **50**, 100, 200, 500
$\|U\|$	100, 200, 500, **1000**, 2000, 5000
α, β, γ	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
δ_v	Normal: $\mu \in \{25, 50, 75, 100, 125\}, \sigma = 50$ Uniform: $[1, 200]$
$\|D\|$	0.1, 0.3, 0.5, 0.7, 0.9
Scalability ($\|U\|$)	10 000, 20 000, 30 000, 40 000, 50 000

For the online setting, because workers' arrivals are unknown, we randomly tested 50 different arrival sequences for each setting. The synthetic datasets were created in Python, and all algorithms were implemented in C++ and executed under the Linux Ubuntu operating system. The experiments were performed on a computer with a 2.40 GHz 16-core Intel Xeon E5620 CPU and 12 GB memory.

5.2 Evaluation for task planning

In this section, we evaluated the proposed algorithms in terms of arrangement utility, running time, and memory cost. We tested the performances of the proposed algorithms by varying the following parameters: the size of U, the social degree d, the size of V, the capacity of V, the distribution of δ_v, and the balance parameters α, β, and γ.

5.2.1 Results on synthetic data

1. Effect of $|V|$. Figs. 2a–2c show the arrangement utility, running time, and memory cost, respectively, when varying $|V|$ in task planning while the other parameters are set to default values. We can make the following observations. First, the arrangement utility decreases as $|V|$ increases. This is because when $|V|$ increases, workers have more options and will choose a task that results in a utility most acceptable to them. Second, the running time decreases as $|V|$ varies, because as $|V|$ increases, workers require less time to choose tasks. Third, the memory usage increases as $|V|$ increases, which is natural, because the data becomes larger.

2. Effect of $|U|$. Figs. 2d–2f show the arrangement utility, running time, and memory cost, respectively, when varying $|U|$. We can make the following observations. First, the arrangement utility increases when $|U|$ increases, because we must compute the arrangement utilities for more workers when $|U|$ increases. Second, the running time increases as $|U|$ increases. This is because when $|U|$ is larger and $|V|$ is fixed, more workers must be calculated. Third, the memory cost increases as $|U|$ increases, because as $|U|$ increases, it requires more memory.

3. Effect of capacity. We first evaluated the results when δ_v varies following a normal distribution. When δ_v increases, the total capacity of v increases as well. The arrangement utility, running time, and memory cost are shown in Figs. 2g–2i, respectively. We can find that: (1) The arrangement utility generally increases as δ_v increases. This is reasonable, because tasks can accommodate more interested workers when their capacity increases. (2) The running time changes little among all the algorithms. (3) Varying δ_v has little effect on the memory cost of any of the algorithms.

Next, we studied the results obtained when $|\delta_v|$ was generated by a uniform distribution. Figs. 2j–2l show the results of arrangement utility, running time, and memory cost, respectively. The values of δ_v were generated uniformly ($[1, 20]$, $[1, 50]$, $[1, 100]$, $[1, 150]$, and $[1, 200]$) and the other parameters were set to default values. When δ_v increases, the total capacity of v increases as well. We can make the following observations. First, the arrangement utility increases as $|\delta_v|$ increases. Second, varying δ_v causes no obvious changes in running time. However, the greedy

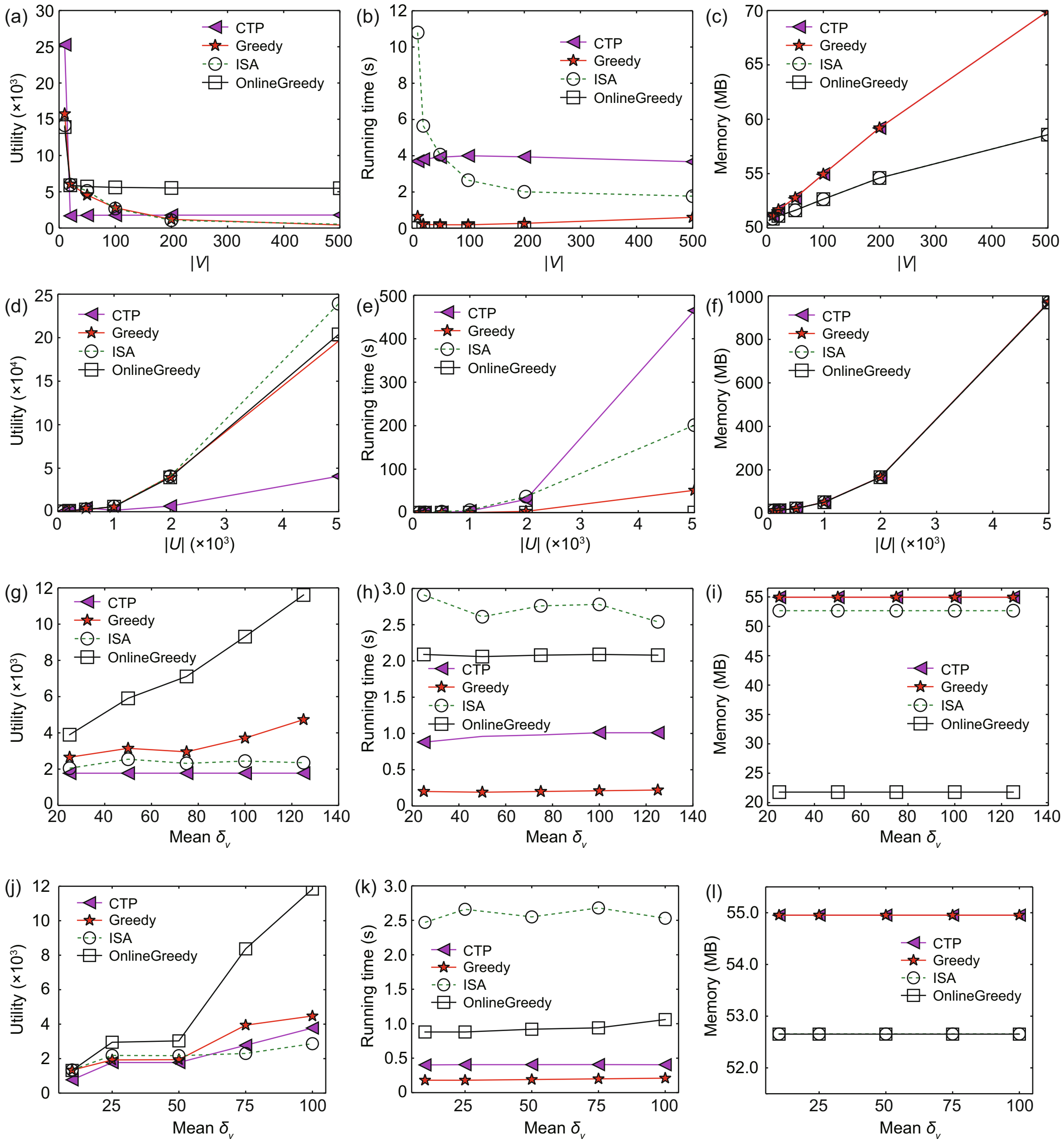

Fig. 2 Results on a synthetic dataset using the proposed algorithms: (a) utility, (b) running time, and (c) memory when varying task size $|V|$; (d) utility, (e) running time, and (f) memory when varying worker size $|U|$; (g) utility, (h) running time, and (i) memory when varying capacity δ_v following a normal distribution; (j) utility, (k) running time, and (l) memory when varying capacity δ_v following a uniform distribution

algorithm takes the least time of all the algorithms. This is because when δ_v increases, a task requires more workers. Crowd workers accomplish tasks in a greedy fashion. Third, varying δ_v has little effect on the memory costs of any of the algorithms.

4. Effect of degree. Figs. 3a–3c show the results of the arrangement utility, running time, and memory cost, respectively, when varying the degree of the social graph. As shown, the arrangement utility, running time, and memory costs all increase as $|d|$ increases, because an increase in $|d|$ requires more computations.

5. Varying the contributions of spatial distance, similarities, and social graph. Figs. 3d–3f show the results of the arrangement utility, running time, and memory cost, respectively, with $\langle\alpha, \beta = \gamma =$

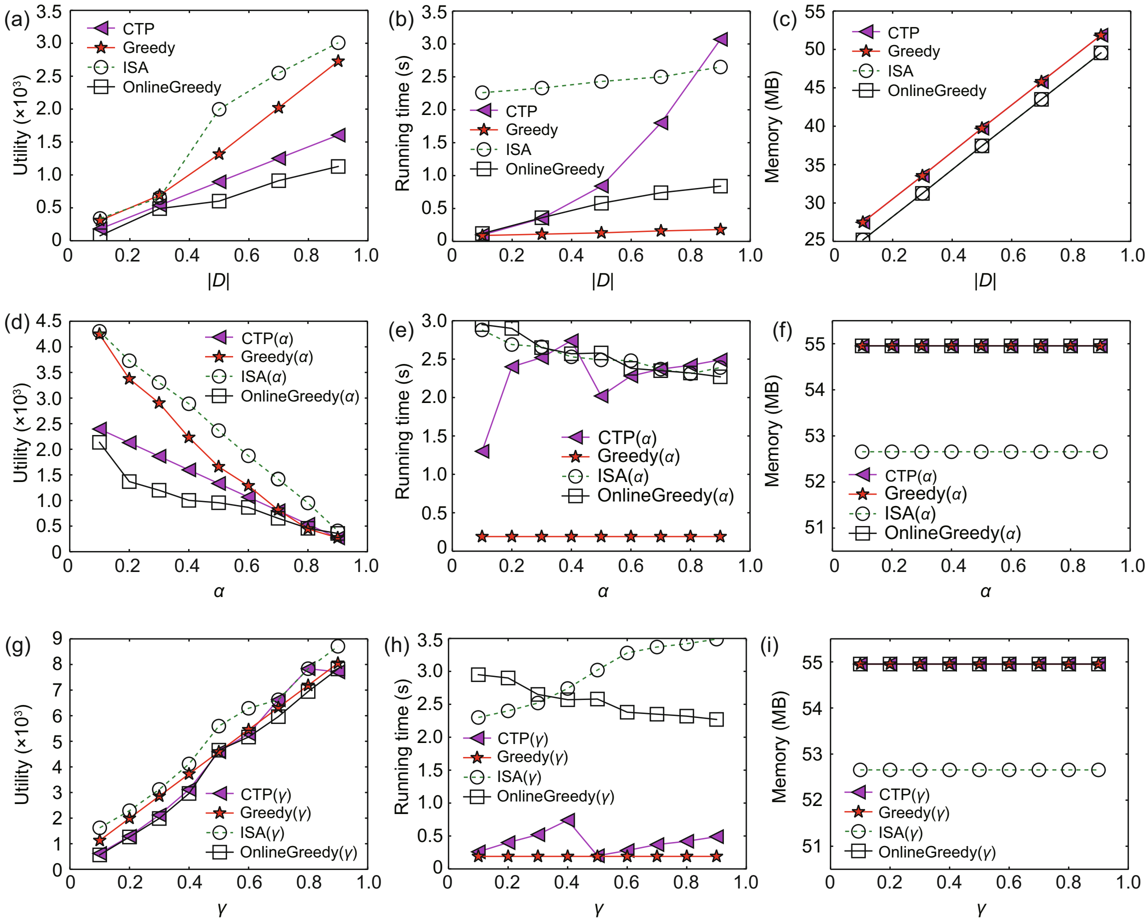

Fig. 3 Results on a real dataset using the proposed algorithms: (a) utility, (b) running time, and (c) memory when varying social graph degree d; (d) utility, (e) running time, and (f) memory when varying α; (g) utility, (h) running time, and (i) memory when varying γ

$(1-\alpha)/2\rangle$ or $\langle\beta, \alpha=\gamma=(1-\beta)/2\rangle$. In addition, Figs. 3g–3i show the results of the arrangement utility, running time, and memory cost, respectively, as α, β, γ vary by $\langle\gamma, \alpha=\beta=(1-\gamma)/2\rangle$. We can make the following observations. First, the arrangement utility decreases when α or β increases. Second, the arrangement utility increases when γ increases. Third, running time and memory cost have the same trends when we choose one of the three combinations.

5.2.2 Results on real dataset

1. Effect of capacity. Figs. 4a–4c show the results of the arrangement utility, running time, and memory cost, respectively, on the real dataset Auckland, which consists of 2012 workers and 225 tasks, when the capacity values were generated by following a normal distribution. The task capacities were not given; we generated the task capacities by following a normal distribution. The results on this real dataset present patterns similar to the results with synthetic data. The results of the arrangement utility, running time, and memory cost are shown in Figs. 4d–4f, respectively, when the capacity values were generated by following a uniform distribution. Similar result patterns were obtained for the other two datasets when capacity values were generated by following normal and uniform distributions.

2. Varying the contribution of spatial distance, similarities, and social graph. Figs. 4g–4i show the arrangement utility, running time, and memory, respectively, when α, β, and γ are set as follows: $\langle\alpha=0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, \beta=\gamma=(1-\alpha)/2\rangle$, or $\langle\beta=0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, \alpha=\gamma=(1-\beta)/2\rangle$. Figs. 4j–4l show the results of the arrangement utility, running time, and memory, respectively, when varying $\gamma=$

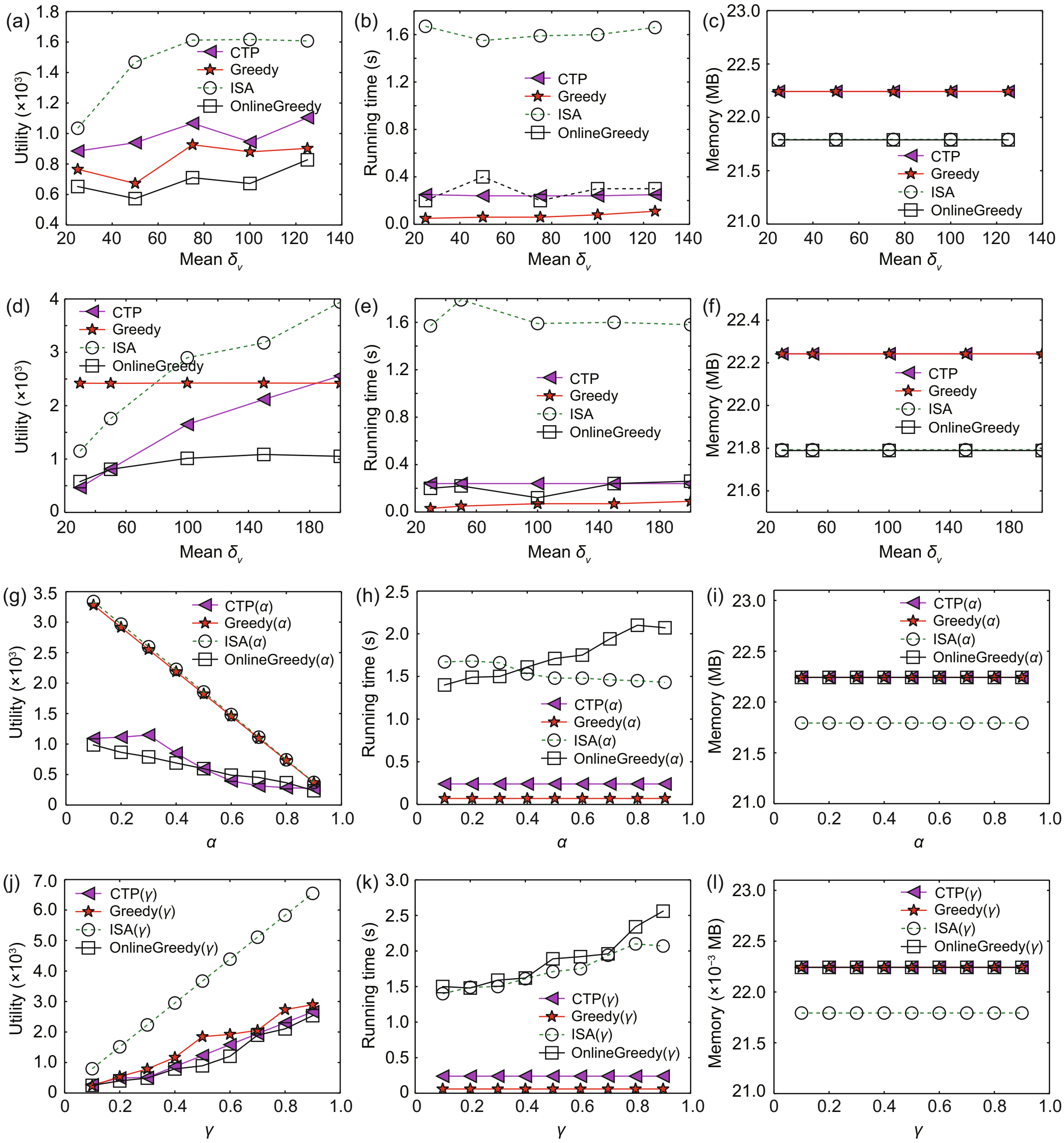

Fig. 4 Results on a real dataset using the proposed algorithms: (a) utility, (b) running time, and (c) memory when varying capacity δ_v following a normal distribution; (d) utility, (e) running time, and (f) memory when varying δ_v following a uniform distribution; (g) utility, (h) running time, and (i) memory when varying α; (j) utility, (k) running time, and (l) memory when varying γ

$0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$ and $\alpha = \beta = (1-\gamma)/2$. The results show similar patterns to the results with synthetic data. Note that the arrangement utility increases when γ increases.

3. Scalability. We studied the scalability of all the algorithms under both the offline and online settings. Specifically, we set $|V|$ to 100 and $|U|$ to 10 000, 20 000, 30 000, 40 000, and 50 000. Because $|U|$ is relatively large, we set the total capacity of tasks to 1.2 times the number of workers; the other parameters were set to default values. The results are shown in Figs. 5a–5c in terms of arrangement utility, running time, and memory cost, respectively. We can observe that the arrangement utility, running time, and memory cost of all algorithms grow linearly with the size of the data. In addition, the results show that all the algorithms are scalable in terms of both time and memory cost.

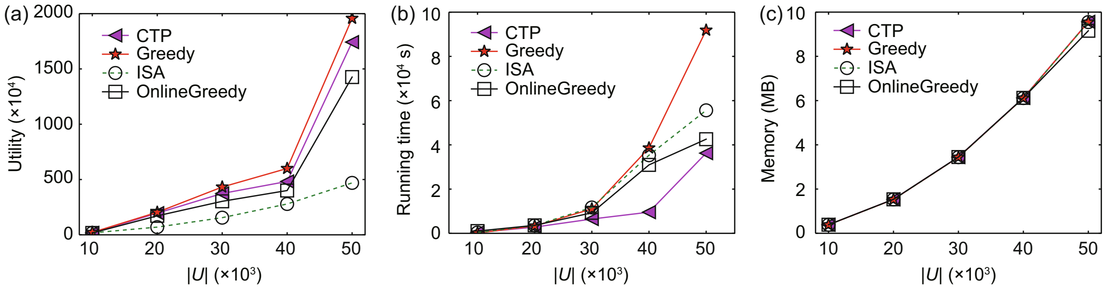

Fig. 5 Results of utility (a), running time (b), and memory (c) for scalability test

5.2.3 Summary

The CTP, Greedy, and ISA algorithms are efficient for offline task planning. The ISA algorithm performs better than other algorithms in calculating the arrangement utility, but it is less efficient with regard to running time in more cases. The OnlineGreedy algorithm is quite effective for online planning, even when compared with the offline algorithms, which have complete information about workers and tasks.

6 Related work

In this section, we review the related work in three categories: mobile crowdsourcing, event-based social networks (EBSNs), and online matching.

1. Mobile crowdsourcing. Recently, as a novel human-machine collaborative computation paradigm (Zhang *et al.*, 2014a), data-driven crowdsourcing has already attracted much attention in the computer science community. In particular, some fundamental issues of crowdsourcing have been widely studied, for example, data cleaning (Tong *et al.*, 2014b; Zhang *et al.*, 2015), topic discovery (Tong *et al.*, 2014a), taxonomy construction (Meng *et al.*, 2015), and expert discovery (Cao *et al.*, 2012; 2013).

With the development of the mobile Internet and distributed systems (Tong *et al.*, 2016d), more and more real applications of mobile crowdsourcing are emerging, e.g., Uber and Gigwalk. Note that mobile crowdsourcing is also called spatial crowdsourcing or spatio-temporal crowdsourcing (Kazemi and Shahabi, 2012). The existing research on mobile crowdsourcing focuses mainly on two types of problems: task assignment and quality control. For task assignment, Tong *et al.* (2016b) proposed a general bipartite-matching-based framework to address dynamic task allocation in online mobile crowdsourcing platforms. The problem of collaborative task recommendation in mobile crowdsourcing has also been proposed recently (Gao *et al.*, 2016). For the quality control problem, different from the traditional web-based crowdsourcing that usually adopts uncertain data processing techniques to control the correct ratio of crowd workers (Cao *et al.*, 2012; Tong *et al.*, 2012a; 2012b; 2012c; 2015a; Yang *et al.*, 2012; Sun and Chen, 2013), the goal of quality control in mobile crowdsourcing changes to minimize the total waiting time that the crowd workers experience in arriving at the specific location of tasks (Zhang *et al.*, 2014b; Tong *et al.*, 2016a). Although the aforementioned works study various problems of mobile crowdsourcing, most of them do not address the task planning problem based on the friendship among different crowd workers.

2. Event-based social networks. Many existing studies have been performed on EBSNs (Liu *et al.*, 2012). The unique features of ESBNs were first considered by Liu *et al.* (2012). However, research in this field did not consider the effects of dynamic worker arrivals. Recently, Li *et al.* (2014) introduced the social event organization (SEO) problem, assigning workers to tasks in such a way that their overall innate and social affinities are maximized. However, the solution in Li *et al.* (2014) considers only two factors: attribute similarities and friendships among workers. They neglected the spatial influences of tasks and workers. They also neglected how to recommend a task to a newly arriving worker in real time to obtain the greatest satisfaction. A novel approach for EBSNs was developed in Armenatzoglou *et al.* (2015), which introduced multi-criterion social graph partitioning—a game-theoretic approach for EBSNs—that considers two factors: the distance

between workers and tasks and the friendships among workers. In Armenatzoglou *et al.* (2015), the model was based on graph partitioning. A social graph is partitioned into a set of tasks such that workers at the same task have a high social connectivity. Their solution is based on a game-theoretic framework and each worker is considered as a player. Players were first randomly assigned to tasks; then, they began changing tasks according to their best responses until they reached a Nash equilibrium. Armenatzoglou *et al.* (2015) did not consider situations in which tasks have limited capacities and in which workers arrive sequentially. She *et al.* (2015b) introduced a global event-participant arrangement with a conflict and capacity (GEACC) problem, focusing on the conflicts between different tasks and on generating task planning from a global view. However, they also failed to consider newly arriving workers. Tong *et al.* (2016c) introduced a general model to recommend suitable social tasks to potential workers according to the following three factors: the location influence of tasks and workers, attribute similarities between tasks and workers, and friendships among workers. However, they did not consider dynamic worker arrivals or how to recommend tasks to them such that they would gain maximum satisfaction. Consequently, all these studies differ from our research.

3. Online matching. In recent years, there have been a series of studies on online matching, such as Karp *et al.* (1990), Mehta (2012), Ting and Xiang (2015), and Tong *et al.* (2016b). The input of online matching is a weighted bipartite graph $G = (L, R, E, U)$, whose left-hand vertices L are known beforehand, but the right-hand vertices R are unknown and arrive one by one. Once a right-hand vertex $r \in R$ arrives, the edges $(l, r) \in E$ incident to r and their corresponding weights $U(l, r) \in U$ are revealed, and r must either match a left-hand vertex l or remain unmatched thereafter (Burkard *et al.*, 2009; Tong *et al.*, 2016b). In particular, Ting and Xiang (2015) introduced weighted online bipartite matching problems. Bipartite graph matching and assignment problems have been widely studied for decades. Related research has been surveyed by Burkard *et al.* (2009) and West (2001). Besides classical bipartite graph matching, another closely related work is the assignment problem (Burkard *et al.*, 2009). However, the original assignment problem does not consider the capacity and social friendship constraints proposed in our problem. Thus, our problem differs from previous works in that it is more difficult (NP-hard) due to the social graph. These works also did not consider the social graphs of workers.

7 Conclusions

In this paper, we identified offline and online task planning variants for mobile crowdsourcing. For the offline planning problem, we devised three algorithms to solve it, named CTP, Greedy, and ISA. For online planning, we also proposed an OnlineGreedy algorithm, which resolves the problem scenario in which the full information is unknown. Finally, we verified the effectiveness and efficiency of the proposed solutions through extensive experiments on both real and synthetic datasets.

While these results are quite promising, there are significant opportunities for further improvement. In real applications, tasks and workers appear dynamically and their spatio-temporal information cannot be known in advance. However, this scenario requires immediate responses from mobile crowdscourcing platforms. There is a challenging problem: How to assign the tasks to suitable workers in real-time dynamic environments and model the two-online scenario?

References

Armenatzoglou, N., Pham, H., Ntranos, V., *et al.*, 2015. Real-time multi-criteria social graph partitioning: a game theoretic approach. Proc. ACM SIGMOD Int. Conf. on Management of Data, p.1617-1628. http://dx.doi.org/10.1145/2723372.2749450

Burkard, R.E., Dell'Amico, M., Martello, S., 2009. Assignment Problems. Society for Industrial and Applied Mathematics, Philadelphia.

Cao, C.C., She, J., Tong, Y., *et al.*, 2012. Whom to ask? Jury selection for decision making tasks on micro-blog services. *Proc. VLDB Endow.*, **5**(11):1495-1506. http://dx.doi.org/10.14778/2350229.2350264

Cao, C.C., Tong, Y., Chen, L., *et al.*, 2013. WiseMarket: a new paradigm for managing wisdom of online social users. Proc. ACM SIGGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.455-463. http://dx.doi.org/10.1145/2487575.2487642

Cheng, Y., Yuan, Y., Chen, L., *et al.*, 2016. DistR: a distributed method for the reachability query over large uncertain graphs. *IEEE Trans. Parall. Distr. Syst.*, **27**(11):3172-3185. http://dx.doi.org/10.1109/TPDS.2016.2535444

Gao, D., Tong, Y., She, J., *et al.*, 2016. Top-k team recommendation in spatial crowdsourcing. Proc. Int. Conf.

on Web-Age Information Management, p.191-204. http://dx.doi.org/10.1007/978-3-319-39937-9_15

Karp, R.M., Vazirani, U.V., Vazirani, V.V., 1990. An optimal algorithm for on-line bipartite matching. Proc. 22nd Annual ACM Symp. on Theory of Computing, p.352-358. http://dx.doi.org/10.1145/100216.100262

Kazemi, L., Shahabi, C., 2012. GeoCrowd: enabling query answering with spatial crowdsourcing. Proc. 20th Int. Conf. on Advances in Geographic Information Systems, p.189-198. http://dx.doi.org/10.1145/2424321.2424346

Kirkpatrick, S., Gelatt, J.C.D., Vecchi, M.P., 1987. Optimization by simulated annealing. *Science*, **220**(4598): 671-680. http://dx.doi.org/10.1126/science.220.4598.671

Li, K., Lu, W., Bhagat, S., *et al.*, 2014. On social event organization. Proc. 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.1206-1215. http://dx.doi.org/10.1145/2623330.2623724

Liu, X., He, Q., Tian, Y., *et al.*, 2012. Event-based social networks: linking the online and offline social worlds. Proc. 18th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.1032-1040. http://dx.doi.org/10.1145/2339530.2339693

Mehta, A., 2012. Online matching and ad allocation. *Found. Trends Theor. Comput. Sci.*, **8**(4):265-368. http://dx.doi.org/10.1561/0400000057

Meng, R., Tong, Y., Chen, L., *et al.*, 2015. CrowdTC: crowdsourced taxonomy construction. Proc. IEEE Int. Conf. on Data Mining, p.913-918. http://dx.doi.org/10.1109/ICDM.2015.77

Musthag, M., Ganesan, D., 2013. Labor dynamics in a mobile micro-task market. Proc. SIGCHI Conf. on Human Factors in Computing Systems, p.641-650. http://dx.doi.org/10.1145/2470654.2470745

Pan, Y.H., 2016. Heading toward artificial intelligence 2.0. *Engineering*, **2**(4):409-413. http://dx.doi.org/10.1016/J.ENG.2016.04.018

She, J., Tong, Y., Chen, L., 2015a. Utility-aware social event-participant planning. Proc. ACM SIGMOD Int. Conf. on Management of Data, p.1629-1643. http://dx.doi.org/10.1145/2723372.2749446

She, J., Tong, Y., Chen, L., *et al.*, 2015b. Conflict-aware event-participant arrangement. Proc. 31st IEEE Int. Conf. on Data Engineering, p.735-746. http://dx.doi.org/10.1109/ICDE.2015.7113329

She, J., Tong, Y., Chen, L., *et al.*, 2016. Conflict-aware event-participant arrangement and its variant for online setting. *IEEE Trans. Knowl. Data Eng.*, **28**(9):2281-2295. http://dx.doi.org/10.1109/TKDE.2016.2565468

Sun, Y., Chen, C.C., 2013. A novel social event recommendation method based on social and collaborative friendships. Int. Conf. on Social Informatics, p.109-118. http://dx.doi.org/10.1007/978-3-319-03260-3_10

Ting, H.F., Xiang, X.Z., 2015. Near optimal algorithms for online maximum edge-weighted *b*-matching and two-sided vertex-weighted *b*-matching. *Theor. Comput. Sci.*, **607**(2):247-256. http://dx.doi.org/10.1016/j.tcs.2015.05.032

Tong, Y., Chen, L., Cheng, Y., *et al.*, 2012a. Mining frequent itemsets over uncertain databases. *Proc. VLDB Endow.*, **5**(11):1650-1661. http://dx.doi.org/10.14778/2350229.2350277

Tong, Y., Chen, L., Ding, B., 2012b. Discovering threshold-based frequent closed itemsets over probabilistic data. Proc. IEEE 28th Int. Conf. on Data Engineering, p.270-281. http://dx.doi.org/10.1109/ICDE.2012.51

Tong, Y., Chen, L., Yu, P.S., 2012c. UFIMT: an uncertain frequent itemset mining toolbox. Proc. 18th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.1508-1511. http://dx.doi.org/10.1145/2339530.2339767

Tong, Y., Cao, C.C., Chen, L., 2014a. TCS: efficient topic discovery over crowd-oriented service data. Proc. 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.861-870. http://dx.doi.org/10.1145/2623330.2623647

Tong, Y., Cao, C.C., Zhang, C.J., *et al.*, 2014b. CrowdCleaner: data cleaning for multi-version data on the web via crowdsourcing. Proc. IEEE 30th Int. Conf. on Data Engineering, p.1182-1185. http://dx.doi.org/10.1109/ICDE.2014.6816736

Tong, Y., Chen, L., She, J., 2015a. Mining frequent itemsets in correlated uncertain databases. *J. Comput. Sci. Technol.*, **30**(4):696-712. http://dx.doi.org/10.1007/s11390-015-1555-9

Tong, Y., She, J., Chen, L., 2015b. Towards better understanding of App functions. *J. Comput. Sci. Technol.*, **30**(5):1130-1140. http://dx.doi.org/10.1007/s11390-015-1588-0

Tong, Y., She, J., Ding, B., *et al.*, 2016a. Online minimum matching in real-time spatial data: experiments and analysis. *Proc. VLDB Endow.*, **9**(12):1053-1064. http://dx.doi.org/10.14778/2994509.2994523

Tong, Y., She, J., Ding, B., *et al.*, 2016b. Online mobile micro-task allocation in spatial crowdsourcing. Proc. 32nd IEEE Int. Conf. on Data Engineering, p.49-60. http://dx.doi.org/10.1109/ICDE.2016.7498228

Tong, Y., She, J., Meng, R., 2016c. Bottleneck-aware arrangement over event-based social networks: the max-min approach. *World Wide Web J.*, **19**(6):1151-1177. http://dx.doi.org/10.1007/s11280-015-0377-6

Tong, Y., Zhang, X., Chen, L., 2016d. Tracking frequent items over distributed probabilistic data. *World Wide Web J.*, **19**(4):579-604. http://dx.doi.org/10.1007/s11280-015-0341-5

West, D.B., 2001. Introduction to Graph Theory. Pearson.

Yang, D., Shen, C., Lee, W., *et al.*, 2012. On socio-spatial group query for location-based social networks. Proc. 18th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.949-957. http://dx.doi.org/10.1145/2339530.2339679

Zhang, C.J., Chen, L., Tong, Y., 2014a. MaC: a probabilistic framework for query answering with machine-crowd collaboration. Proc. 23rd ACM Int. Conf. on Information and Knowledge Management, p.11-20. http://dx.doi.org/10.1145/2661829.2661880

Zhang, C.J., Tong, Y., Chen, L., 2014b. Where to: crowd-aided path selection. *Proc. VLDB Endow.*, **7**(14): 2005-2016. http://dx.doi.org/10.14778/2733085.2733105

Zhang, C.J., Chen, L., Tong, Y., *et al.*, 2015. Cleaning uncertain data with a noisy crowd. Proc. 31st IEEE Int. Conf. on Data Engineering, p.6-17. http://dx.doi.org/10.1109/ICDE.2015.7113268