Assigning Tasks to Workers by Referring to Their Schedules in Mobile Crowdsourcing

Mayumi Hadano[†], **Makoto Nakatsuji**[††], **Hiroyuki Toda**[†], and **Yoshimasa Koike**[†]
[†]NTT Service Evolution Laboratories, NTT Corporation
[†]{hadano.mayumi, toda.hiroyuki, koike.y}@lab.ntt.co.jp
[††]NTT Resonant, Inc., nakatuji@nttr.co.jp

Abstract

This paper focuses on task assignments to workers in mobile crowdsoucing systems. The current method does not work so well since it considers only workers who are ready to work at the time of optimization. Our method handles workers' day-long schedules, creates a 'time-extended' worker-task graph that expresses the relationships between workers and tasks over a time period and finds the best set of worker-task-time triples. Our evaluation using real world visiting logs shows it increases the rate of assigned tasks by more than 8.2% compared with a state-of-the-art assignment method.

Introduction

The spread of smart devices has made it feasible to develop a mobile crowdsourcing system, where crowd workers perform a variety of location-specific tasks through their mobile devices. Services of this sort have attracted a lot of interests from business, and they would be especially beneficial to local businesses seeking detailed information that could be cheaply gathered by crowd workers.

A representative task-assignment study (Kazemi and Shahabi 2012) in the mobile crowdsourcing field tries to maximize the number of assigned tasks and then to minimize the total costs by modeling the task assignment as a min-cost max flow problem. However, crowd workers usually perform tasks during their free time so they often appear, disappear, and move. As a result, it misses opportunities to assign tasks to workers who will appear after its optimization and decrease the number of assigned tasks. To tackle this problem, we propose a new task assignment under the condition that the system can refer to workers' schedules for the following reason; It has benefits for workers because sharing their schedules enables crowdsourcing system to avoid sending notification to workers when they are not free.

We briefly explain the steps of our method. First, all possible triples consisting of tasks, workers, and available time periods are created on the basis of the distance between the workers and the tasks as well as workers' available time by referring to their schedules. Next, directed graphs are created for each time period in the same way as the current method, and they are merged into one 'time-extended' graph covering all time periods (Fig. 1). It then finds best set of worker-task-time triples that maximize the number of assigned tasks and minimize the total cost (e.g. time and distance). To find this set, we regards extracting worker-task-time triples as min-cost max flow problem (Hassin 1982).

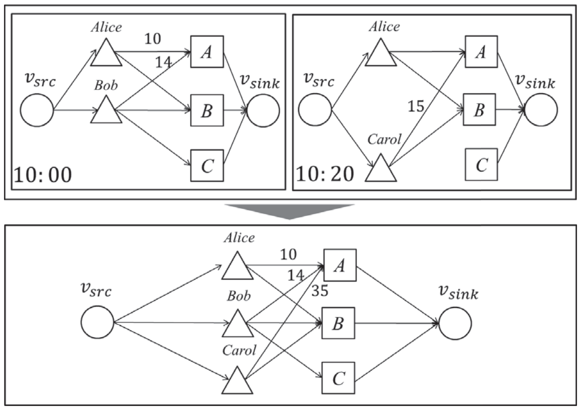

Figure 1: Task-worker graph generated by our method (top: the worker-task graph for each time period, bottom: the time-extended worker-task graph used in our method).

Method

This section explains the terminology and our method.

Definition 1 (Task). Task $t_i \in \mathbf{T}$ is submitted by requesters to the mobile crowdsourcing system. We specify the location of each task t_i by using a longitude and latitude. For simplicity, we do not care about reward of tasks.

Definition 2 (Worker). Worker $w_j \in \mathbf{W}$ performs a task in the mobile crowdsourcing system. We specify the location of each worker w_j by using a longitude and latitude. The set of workers who are available during the whole time period is denoted as $\mathbf{W}(\Phi)$. A worker who comes at time period ϕ_k is denoted as $w_j(\phi_k)$, and he shares the maximum number of tasks ($maxT_j(\phi_k)$) and available region ($r_j(\phi_k)$) with the system. He can move within the available region. We assume that worker who is assigned to a task by the system does not decline to perform the task.

Definition 3 (Worker-task graph). A worker-task graph $G(\mathbf{V}, \mathbf{E})$ is composed of a set of vertices and edges (Fig.

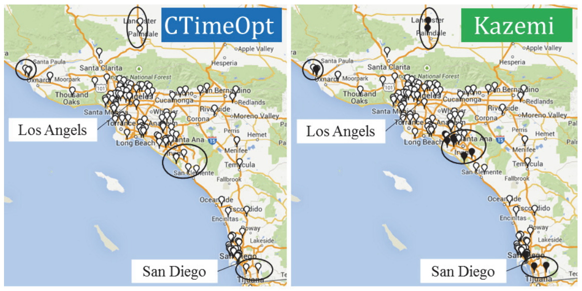

Figure 2: Spatial distribution of assigned tasks (white pins: assigned tasks, black pins: not-assigned tasks).

1). The vertices represent tasks and workers, and there are two special vertices, v_{src} and v_{sink}, that represent the source of flows and the sink of flows, respectively. The edges represent the possibilities of workers performing tasks. Each edge (u, v) in the worker-task graph carries a flow. The amount of flow passing from vertex u to vertex v is denoted as $f(u, v)$. Each edge has a capacity $\bar{f}(u, v)$ and cost $c(u, v)$ which is paid to send a unit flow. The way to connect the edges and compute the capacities and costs of the edges are shown in the study of (Kazemi and Shahabi 2012).

Our method is composed of two components. First component merges worker-task graphs of certain time periods into a 'time-extended' worker-task graph (Fig. 1). It generates edges from the source vertex to the worker vertices that exist over the whole time period. Next, it generates edges from the worker vertices to the task vertices if they are paired in a set of worker-task pairs $\mathbf{P}(\varPhi)$; each pair in this set represents the possibility of worker w_j performing task t_i during the whole time period. Then, it generates edges from the task vertices to the sink vertex.

Next component finds a set of assignment results. First, the component assigns capacities to the edges in the time-extended worker-task graph in the way explained in the previous methods (Kazemi and Shahabi 2012). Next, it assigns costs to edges. The costs that the workers should pay when performing tasks are assigned to the edges in the time-extended worker-task graph. It is computed as follows:

$$c^*(u,v)=\begin{cases} 0 & (u=v_{src}, v\in \mathbf{W}(\varPhi)) \\ time(w_j(\phi_k),t_i)+\phi_k & ((u,v)\in\mathbf{P}(\varPhi)) \\ 0 & (u\in\mathbf{T}(\phi_k), v{=}v_{sink}) \end{cases}$$

where $time(w_j(\phi_k), t_i)$ represents the time taken by worker $w_j(\phi_k)$ to complete tasks t_i. Thus, the objective function of min-cost max task assignment is different from previous method and formulated as follows:

$$\min \sum_{(u,v)\in\mathbf{E}} f(u,v)c^*(u,v). \tag{1}$$

The constraints are defined as

$$f(u,v) \leq \bar{f}(u,v), \quad \forall(u,v)\in\mathbf{E} \tag{2}$$

$$\sum_{v\in\mathbf{V}} f(u,v) = \sum_{v\in\mathbf{V}} f(v,u), \forall u \in \mathbf{V}\backslash\{v_{src}, v_{sink}\} \tag{3}$$

$$\sum_{u\in\mathbf{V}} f(v_{src}, u) = f_{max} \tag{4}$$

where f_{max} represents the result of the max flow problem. Then it solves the min-cost max flow problem by using the time-extended worker-task graph. In doing so, it obtains a set of worker-task-time triples by finding edges that receive the flows from worker vertices $w_j(\phi_k)$s to task vertices t_is.

Evaluation

Experimental setting. We used real-world visiting logs (Gowalla data[1]) to evaluate our method. In the same way as the previous studies did (Kazemi and Shahabi 2012; Deng, Shahabi, and Demiryurek 2013), we also regard check-in locations in visiting logs as the places of workers and tasks in the mobile crowdsourcing system. The tasks and workers were randomly generated from the check-in locations. The tasks were generated at 0:00 and expired at 23:59. The number of tasks and workers varied from 100 to 500. For each of our experiments, we ran 50 cases, and reported the average of the results. We compared two methods, Kazemi (Kazemi and Shahabi 2012) and CTimeOpt (our method).

Results. The number of tasks that had been assigned by CTimeOpt totaled 255.4, whereas Kazemi totaled 236.1 when the best parameter settings ($|W| = 400$ and $|T| = 300$). Hence, our method assigned 8.2% more tasks. Fig. 2 shows the part of spatial distributions of tasks that each method assigned to workers at 23:59. From this figure, we can see that CTimeOpt succeeded in assigning tasks located far from Los Angels to workers, whereas Kazemi failed to do so. This is because CTimeOpt can refer to the workers' schedules and thus it can wait to assign workers who can complete the tasks that are not covered by the regions of most workers. The rate of tasks assigned by CTimeOpt is higher than that by Kazemi when the number of workers is relatively greater than that of tasks.

Conclusion

This paper tackled the task assignment problem in the context of a mobile crowdsourcing system. While the existing task assignment method has a limitation in that it ignores the dynamic nature of workers and tasks, our method uses the changes in the workers' locations over time by referring to their schedules. Our evaluation showed that our method increased the coverage of assigned tasks by more than 8.2% compared with a state-of-the-art approach.

References

Deng, D.; Shahabi, C.; and Demiryurek, U. 2013. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *Proc. SIGSPATIAL'13*, 314–323.

Hassin, R. 1982. Minimum cost flow with set-constraints. *Networks* 12(1):1–21.

Kazemi, L., and Shahabi, C. 2012. Geocrowd: Enabling query answering with spatial crowdsourcing. In *Proc. SIGSPATIAL'12*, 189–198.

[1]See "http://snap.stanford.edu/data/loc-gowalla.html"