Profit-driven Task Assignment in Spatial Crowdsourcing

Jinfu Xia[1], **Yan Zhao**[1*], **Guanfeng Liu**[2], **Jiajie Xu**[1], **Min Zhang**[1] and **Kai Zheng**[3†]
[1]Institute of Artificial Intelligence, School of Computer Science and Technology, Soochow University
[2]Macquarie University
[3]University of Electronic Science and Technology of China
jfxia@stu.suda.edu.cn, zhaoyan@suda.edu.cn, guanfeng.liu@mq.edu.au,
{xujj, minzhang}@suda.edu.cn, zhengkai@uestc.edu.cn

Abstract

In Spatial Crowdsourcing (SC) systems, mobile users are enabled to perform spatio-temporal tasks by physically traveling to specified locations with the SC platforms. SC platforms manage the systems and recruit mobile users to contribute to the SC systems, whose commercial success depends on the profit attained from the task requesters. In order to maximize its profit, an SC platform needs an online management mechanism to assign the tasks to suitable workers. How to assign the tasks to workers more cost-effectively with the spatio-temporal constraints is one of the most difficult problems in SC. To deal with this challenge, we propose a novel Profit-driven Task Assignment (PTA) problem, which aims to maximize the profit of the platform. Specifically, we first establish a task reward pricing model with tasks' temporal constraints (i.e., expected completion time and deadline). Then we adopt an optimal algorithm based on tree decomposition to achieve the optimal task assignment and propose greedy algorithms to improve the computational efficiency. Finally, we conduct extensive experiments using real and synthetic datasets, verifying the practicability of our proposed methods.

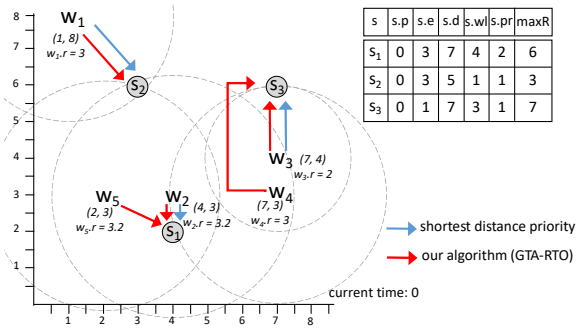
1 Introduction

Recent years have witnessed a revolution in Spatial Crowdsourcing (SC), where people with mobile sensing facilities can move as sensors and participate some location-based tasks instantaneously [Deng *et al.*, 2015; Tong *et al.*, 2016a; Tong *et al.*, 2016b; Song *et al.*, 2017; Li *et al.*, 2015; Zhao *et al.*, 2017; Tong *et al.*, 2018a; Tong *et al.*, 2017; Tong *et al.*, 2018b]. A typical SC system consists of a platform, task requesters and mobile workers. The role of platform is to provide task assignment services to task requesters and encourage workers to contribute the system. Most existing research in SC focuses on task assignment with different optimization strategies [Cheng *et al.*, 2015b; Cheng *et al.*, 2015a; Zhao *et al.*, 2019a]. Although task assignment has been studied for many years to improve the performance of SC, problems have been long-standing remain, which include how to optimize the profit for SC platform.

A primary driving force of the SC platform is its inherent cost effectiveness. In other words, it is in the SC platform's interests to maximize the profit during task assignment. Although several SC applications have been proposed, most of them assume that the SC platform assigns tasks to mobile workers voluntarily without considering the profit of SC system [Jain *et al.*, 2009; Cooper *et al.*, 2011], which are often difficult to engineer non-monetary incentive schemes for tedious and repetitive work [Singer and Mittal, 2013] and unrealistic for commercial SC platform due to its profit-driven nature. Moreover, workers are not willing to perform the assigned tasks without actual payments or credits as they have various participation cost (e.g., mobile device battery energy cost for sensing and data processing). Several profit-based task assignment mechanisms have been developed in crowdsourcing system. For instance, considering the profit of platform, Yang et al. [Yang *et al.*, 2012] design incentive mechanisms for mobile crowdsourcing systems based on two system models (i.e., platform-centric and user-centric model), but they mainly focus on improving the computational efficiency instead of improving the profit. [Shah-Mansouri and Wong, 2015] proposes a profit maximizing truthful auction mechanism, in which the platform acts as an auctioneer and workers act as sellers submitting their bids to the platform. However, when assigning tasks, the above researches fail to consider the spatio-temporal constraints, which play an essential role in SC. Meanwhile, they improve the total profit for the platform by designing different incentive mechanisms instead of paying attention to task assignment process.

Fig. 1 illustrates an example of profit-based task assignment problem with five workers (indicated as $\{w_1,...,w_5\}$) and three tasks (shown as $\{s_1, s_2, s_3\}$). Each worker is associated with her current location and reachable distance range (marked as $w.r$). Each task, published and expired at different time instances, is labelled with its workload ($s.wl$) and reward information (i.e., penalty rate $s.pr$ and maximum reward $s.maxR$). The problem is to assign tasks to the suitable workers so as to maximize the total platform profit. In SC, it is intuitive to assign the nearby tasks to workers without violating the spatio-temporal constraint, referred to as *shortest*

*Jinfu Xia and Yan Zhao made equal contribution.
†Corresponding Author



s	s.p	s.e	s.d	s.wl	s.pr	maxR
s_1	0	3	7	4	2	6
s_2	0	3	5	1	1	3
s_3	0	1	7	3	1	7

Figure 1: Running example

distance priority algorithm. Therefore, we can obtain a task assignment, $\{< s_1, w_2 >, < s_2, w_1 >, < s_3, w_3 >\}$ (shown in blue arrow lines in Fig. 1), with the platform profit of 5.7 (we assume the platform gets 80% of the reward as its profit). However, this assignment algorithm leaves w_4 and w_5 unassigned, which may contribute more profit for SC platform.

To address these challenges, we propose a profit-based task assignment framework that links the task assignment process with its economic performance in profit. We first formulate the Profit-driven Task Assignment (PTA) problem and show it is NP-hard. Then we establish a reward pricing model with the task's temporal constraints, wherein the total reward is directly associated with the payment of task requesters and the processing time of the task. When it comes to task assignment, we introduce an exact tree-decomposition-based algorithm that finds the optimal assignment result in terms of the total platform profit. For the sake of efficiency, we propose a Greedy Task Assignment (GTA) algorithm that tries to give priority to the tasks with the highest possible reward per unit of work and assign the closest worker to this task until it is finished before its expected completion time. Meanwhile, GTA with two Random Tuning Optimization strategies (GTA-RTO) are developed to prune the non-promising worker-task assignment pairs. The red arrow lines in Fig. 1 depict the task assignment by our GTA-RTO algorithm that generates the profit of 8.72.

Our primary contributions can be summarized as follows:

1) We formulate a PTA problem in SC. To the best of our knowledge, this is the first work that aims to maximize the profit for SC platform during task assignment process.

2) We develop a reward pricing model by considering both task's expected completion time and task's expiration time.

3) We develop both optimal and greedy task assignment algorithms to address the proposed problem.

4) Extensive experiments are conducted to verify the effectiveness and efficiency of the proposed methods.

2 Problem Statement

We define a set of preliminaries and formulate our PTA problem.

Definition 1 (Spatial Task) *A spatial task, denoted by* $s = < s.l, s.p, s.e, s.d, s.wl, s.maxR, s.pr >$*, is a task to be performed at location* $s.l$*, published at time* $s.p$*, expected to be finished at time* $s.e$ *and will expire at deadline* $s.d$*. Each task is also labelled with a required workload* $s.wl$ *to finish task* s *by a normal worker (we simply use the time required to finish a task to denote* $s.wl$*).* $s.maxR$ *is the maximum reward the task requester can provide and* $s.pr$ *is a penalty rate, which establishes a correlation between completion time and reward.*

Definition 2 (Worker) *A worker, denoted by* $w = < w.l, w.r >$*, is a person who is able to perform tasks only if she is paid. A worker is associated with her current location* $w.l$ *and her reachable circular range with* $w.l$ *as the center and* $w.r$ *as the radius, in which* w *can accept assignment of tasks.*

Definition 3 (Available Worker Set) *Given a task* s *to be assigned and a set of workers in the vicinity of* s*, the available worker set for task* s*, denoted as* $AWS(s)$*, should satisfy the following two conditions:* $\forall w \in AWS(s)$*,*

1) $t_{now} + t(w.l, s.l) < s.d$*, and*

2) $d(w.l, s.l) \leq w.r$*, and*

3) $\sum_{w \in AWS(s)} w.wl(s) = s.wl$*,*

where t_{now} *is current time,* $t(a, b)$ *is travel time from location* a *to* b*,* $d(a, b)$ *is travel distance from location* a *to* b*, and* $w.wl(s)(> 0)$ *is the workload assigned to* w *to finish* s*.*

Definition 4 (Platform Profit) *Given a task* s *to be assigned and an available worker set* $AWS(s)$ *for* s*, the profit of SC platform can be computed as* $P_{AWS(s)} = \alpha R_{AWS(s)}$*, where* $P_{AWS(s)}$ *and* $R_{AWS(s)}$ $(0 \leq R_{AWS(s)} \leq s.maxR)$ *are the profit of SC platform to finish* s *and the reward of* s *(i.e., the payment the task requirer provides) respectively when assigning task* s *to workers in* $AWS(s)$*, and* α $(0 < \alpha < 1)$ *is a parameter controlling the earnings that SC platform obtains from the task reward. The reward of* s*,* $R_{AWS(s)}$*, will be elaborated in Section 3.*

Note that we simply assume the profit of an SC platform is in proportion to the reward (e.g., the profit is 20% of the reward when $\alpha = 20\%$) and the remaining reward will be allocated to workers. We just focus on the profit of SC platform while ignoring the reward allocation mechanism of workers.

Definition 5 (Optimal Available Worker Set (OptAWS)) *An available worker set,* $AWS(s)$*, is optimal if every of its proper subsets can only achieve a* $less\text{-}than\text{-}P_{AWS(s)}$ *platform profit.*

Definition 6 (Spatial Task Assignment) *Given a set of workers* W *and a set of tasks* S*, a spatial task assignment, denoted by* A*, consists of a set of* $< task, AWS >$ *pairs in the form of* $< s_1, AWS(s_1) >, < s_2, AWS(s_2) >, \ldots,$ $< s_{|S|}, AWS(s_{|S|}) >$*, where* $\bigcap_{i=1}^{|S|} AWS(s_i) = \emptyset$*.*

Let $A.P$ denote the total profit of SC platform for task assignment A, i.e., $A.P = \sum_{<s, AWS(s)> \in A} P_{AWS(s)}$, and $\mathbb{A}$ denote all possible ways of assignments. The PTA problem can be formally stated as follows: given a worker set W and a task set S, the PTA problem aims to find the global optimal assignment A_{opt}, such that $\forall A_i \in \mathbb{A}$, $A_i.P \leq A_{opt}.P$. We then prove the hardness of PTA problem.

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

Theorem 1 can be proved by doing the reduction from the 0-1 knapsack problem, which is proven to be NP-hard [Vazirani, 2013].

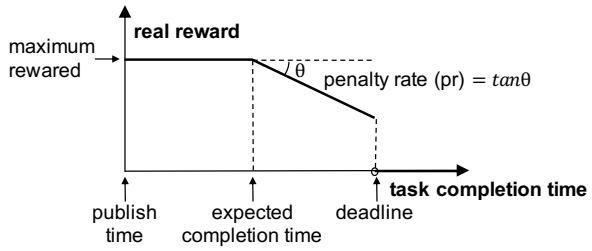


Figure 2: Task reward pricing model

3 Reward Pricing Model Construction

In order to design a cost-effective task assignment algorithm, a reasonable reward pricing model has to be established firstly to quantify the temporal constraints of tasks. In particular, we consider a single task s and one of its available worker set, where the the requester specifies the task's expected completion time $s.e$, deadline $s.d$, maximum reward $s.maxR$ and penalty rate $s.pr$. The model focuses on the task completion time and reward (i.e., the requester's real payment for the task), which are directly associated with the platform profit.

Definition 7 (Task Completion Time) *Given a task s and an available worker set $AWS(s)$, the task's completion time can be calculated as follows:*

$$T(AWS(s)) = s.t_0 + \frac{\sum_{w \in AWS(s)} (a_w(s.l) - s.p) + s.wl}{|AWS(s)|},$$

where $s.t_0$ ($s.t_0 \geq s.p$) is the assignment time, from which we assign task s to worker set $AWS(s)$, $a_w(s.l)$ is the arrival time of w at task s, $s.wl$ is the required workload (i.e., the required time to complete task s) and $\frac{\sum_{w \in AWS(s)} (a_w(s.l) - s.p) + s.wl}{|AWS(s)|}$ denotes the time beginning from $s.t_0$ and ending to the completion of task s.

With the two main time (i.e., $s.e$ and $s.d$) and maximum reward constraints, the mathematical model (shown in Fig. 2) of the real reward can be expressed as followed:

$$R_{AWS(s)} = \begin{cases} s.maxR, & T(AWS(s)) \leq s.e \\ s.maxR - s.pr* & \\ (T(AWS(s)) - s.e), & s.e < T(AWS(s)) \leq s.d \\ 0, & T(AWS(s)) > s.d \end{cases}$$

where $T(AWS(s))$ is s's completion time with $AWS(s)$.

4 Task Assignment

4.1 Optimal Task Assignment (OTA)

It is easy to know that the global optimal result is the union of one possible Optimal Available Worker Set (OptAWS) of all tasks. In this section, we apply a tree-decomposition-based algorithm [Zhao *et al.*, 2017; Zhao *et al.*, 2019b] to achieve the optimal task assignment, which consists of following steps:

1) Find the reachable workers for each task. The reachable worker subset for a task s, denoted as RW_s, should satisfy the following conditions: $\forall w \in RW_s$, $t(w.l, s.l) \leq s.d$ and $d(w.l, s.l) \leq w.r$. The above two conditions guarantee that w can reach s directly before it expires in her reachable range.

2) Find OptAWSs for each task. Given the reachable worker set for each task, we can utilize a dynamic programming algorithm that iteratively expands the sets of tasks in the ascending order of set size and find all OptAWSs in each iteration. The process is omitted due to space limit.

3) Apply the tree-decomposition-based algorithm [Zhao *et al.*, 2017] to find the optimal task assignment with maximal profits. In particular, we first use a tree-decomposition technique to separate all tasks into independent clusters (i.e., tasks in different clusters do not share same available workers) and organize them into a tree structure, such that the tasks in sibling nodes of the tree do not share the same available workers. Then the tree can be traversed in depth-first manner to find the optimal assignment.

Consider the example in Fig. 1. With OTA algorithm, we can get the profit of 8.72 for SC platform.

4.2 Greedy Task Assignment (GTA)

For the sake of efficiency, we propose a basic Greedy Task Assignment (GTA) solution to solve the PTA problem by giving higher priorities to the tasks with higher reward per unit of work and workers with less travel cost.

The procedure of GTA is shown in Algorithm 1, which takes a worker set W and a task set S as input, and returns a suitable task assignment set A, unassigned worker set W' and unassigned task set S'. We first sorts the tasks in S' in descending order according to their reward per unit of work (line 2). Then for each task s in S', we try to assign the first arrival worker in W' to perform s until s can be completed in expected time $s.e$ (line 3-13). If there are no sufficient workers to complete task s, s will be skipped and the workers will not be assigned (line 7-8). The time complexity of GTA is $O(max\{|S| \cdot \log|S|, |S| \cdot |W| \cdot \log|W|\})$. GTA can obtain a task assignment, $\{< s_2, \{w_1, w_2, w_5\} >, < s_3, \{w_3, w_4\} >\}$, with the profit of 5.3 in Fig. 1.

4.3 GTA with Random Tuning Optimization

GTA is a polynomial-time greedy algorithm that finds a task assignment set A with promising solutions. However, there are some issues in GTA. First, GTA always tries to complete a task in expected completion time while ignoring its penalty rate, which is also related to the final profit. Moreover, GTA assigns the closest workers without considering time utilization ratio of workers. For instance, when a task is far away from all the workers, assigning the closest workers to perform this task may lead to a low profit for the whole platform.

To overcome these limitations, we improve GTA with Random Tuning Optimization (GTA-RTO), which contains two tuning strategies: *coarse* and *fine tuning*. Coarse tuning randomly abandons the low-valuable tasks (e.g., tasks with low time utilization ratio of workers) and fine tuning randomly reassigns partial workers to find a better task assignment.

GTA-RTO is shown in Algorithm 2, which starts from a pre-assignment by GTA (line 1). Then the current task assignment $A' (= A)$, unassigned worker set W' and unassigned task set S' enter in the tuning loop (line 3 - 12), wherein the

Algorithm 1 GTA

Input: W, S
Output: A, W', S'
1 $A = \emptyset, W' = W, S' = S$;
2 sorting tasks in S' according to $\frac{s.maxR}{s.wl}$ in descending order;
3 **foreach** $s \in S'$ **do**
4 $W_{assign} = \emptyset$;
5 sorting workers in W' according to their arrival time in increasing order;
6 **while** s *is not completed in expected time* **do**
7 **if** W' *is* $\emptyset$ **then**
8 break;
9 w = closest worker in W';
10 $W_{assign} = W_{assign} \cup \{w\}$;
11 **if** s *is completed* **then**
12 $A = A \cup \{< s, W_{assign} >\}$;
13 $S' = S' - \{s\}$; $W' = W' - W_{assign}$;
14 return A, W', S';

Algorithm 2 GTA-RTO

Input: W, S
Output: A
1 $A', W', S' = GTA(W, S)$;
2 $A = A'$;
3 **repeat**
4 A', W'_C, S'_C = CT(A');
5 $W' = W' \cup W'_c$; $S' = S' \cup S'_C$;
6 A', W'_P, S'_P = FT(A');
7 $W' = W' \cup W'_c$; $S' = S' \cup S'_C$;
8 A_G, W', S' = GTA(W', S');
9 $A' = A' \cup A_G$;
10 **if** $A'.P > A.P$ **then**
11 $A = A'$;
12 **until** *there is no improvement until* n *rounds*;
13 return A;

Coarse Tuning (CT) algorithm and Fine Tuning (FT) algorithm are invoked in order, followed by GTA. Specifically, we first apply CT algorithm with A' as input to generate an unassigned worker set W'_C and an unassigned task set S'_C as well as a modified A' (line 4). Then worker set W' and task set S' are updated (line 5). In the similar way, we update A', W' and S' by invoking FT (line 6 - 7). GTA is invoked to generate a task assignment A_G based on W' and S' (line 8). Subsequently, a new task assignment is updated by the union of A' and A_G. Once the profit of A' is higher than that of A, we replace A with A' (line 9- 11). Algorithm 2 stops when there is no improvement for n rounds, in which n can be specified by the SC platform.

Coarse Tuning (CT)

CT aims to abandon the low-valuable tasks. The possibility $P^{CT}_{s,W}$ that task s with assigned worker set W will be abandoned in CT is defined as follows:

$$P^{CT}_{s,W} = p^{CT}_m + p^{CT}_t * \frac{\sum_{w \in W} a_w(s.l) - s.p}{(T(W) - s.p) * |W|} + p^{CT}_r * (1 - \frac{R(s,W)}{s.maxR}), \quad (1)$$

where $p^{CT}_m + p^{CT}_t + p^{CT}_r = 1$, $T(W)$ is the completion time of task s with assigned worker set W, $R(s,W)$ is the final reward of task s, and $a_w(s.l)$ is the time worker w arrives at task s. $\frac{\sum_{w \in W} a_w(s.l) - s.p}{(T(W) - s.p) * |W|}$ represents the travel time ratio of workers to complete task s and $\frac{R(s,W)}{s.maxR}$ is the rate of return. p^{CT}_m is the minimum possibility that a task can be abandoned. p^{CT}_t and p^{CT}_r are the parameters controlling the contributions of the time utilization ratio of workers and the rate of return.

CT algorithm is shown in Algorithm 3. The input variables is task assignment set A. The output variables are a new task assignment set A', unassigned worker set W' and unassigned task set S'. We first calculate the abandoned possibility of each task s in A' (line 3) and remove the task-worker pair $< s, AWS(s) >$ from A' if the abandoned possibility (line 4 and 5) is less than a random number ζ ($0 \leq \zeta \leq 1$).

Algorithm 3 CT

Input: A
Output: A', W', S'
1 $A' = A, W' = \emptyset, S' = \emptyset$;
2 **foreach** *task-worker pair* $< s, AWS(s) > \in A'$ **do**
3 calculate $p^{CT}_{s,AWS(s)}$ based on Equation 1;
4 **if** $p^{CT}_{s,AWS(s)} < \zeta$ **then**
5 $A' = A' - < s, AWS(s) >$;
6 $S' = S' \cup s$; $W' = W' \cup AWS(s)$;
7 return A', W', S';

Fine Tuning (FT)

In FT processing, no task will be completely abandoned and only a small number of workers will be reassigned, ensuring that tasks can be completed before deadline. The possibility $P^{FT}_{s,w,W}$ that worker $w (\in W)$ assigned to task s will be reassigned is defined as follows:

$$P^{FT}_{s,w,W} = p^{FT}_m + p^{FT}_t * \frac{a_w(s.l) - s.p}{T(W) - s.p}, \quad (2)$$

where $T(W)$ is the completion time of task s with assigned worker set W, and $a_w(s.l)$ is the time that worker w arrives at task s. p^{FT}_m is the minimum possibility that a worker will be reassigned, p^{FT}_t means how much the time utilization ratio of worker w affects the possibility, and $p^{FT}_m + p^{FT}_t = 1$.

Algorithm 4 outlines the FT algorithm with same input and output of CT algorithm. Firstly, we calculate the abandoned possibility of each task s in A' (line 4) and the reassigned possibility of each assigned worker w (line 5). Subsequently we randomly remove one worker w from the current task assignment by weights $P^{FT}_{s,w,AWS(s)}$ (i.e., a worker is more likely to be removed with a higher weight) when the abandoned possibility is less than a random number ζ ($0 \leq \zeta \leq 1$) and s can be finished without w (line 6 - 11).

Algorithm 4 FT

Input: A
Output: A', W', S'
$A' = A, W' = \emptyset, S' = \emptyset$;
foreach *task-worker pair* $< s, AWS(s) > \in A'$ **do**
  $A' = A' - < s, AWS(s) >$;
  calculate $p^{CT}_{s,AWS(s)}$ based on Equation 1;
  calculate $P^{FT}_{s,w,AWS(s)}$ for each worker $w \in AWS(s)$ based on Equation 2;
  while $p^{CT}_{s,AWS(s)} < \zeta$ **do**
    random choose a worker w by weights $P^{FT}_{s,w,AWS(s)}$;
    if *s cannot be completed without w* **then**
      break;
    $AWS(s) = AWS(s) - \{w\}$;
    $W' = W' \cup \{w\}$;
  $A' = A' \cup < s, AWS(s) >$;
return A', W', S';

Taking the case in Fig. 1, we employ GTA-RTO algorithm to get the task assignment, i.e., $\{< s_1, \{w_2, w_5\} >, < s_2, \{w_1\} >, < s_3, \{w_3, w_4\} >\}$, with profit of 8.72, which is same with the profit generated by the OTA algorithm. That demonstrates the superiority of our GTA-RTO algorithm, which can obtain a near optimal result.

5 Experiment

5.1 Experimental Setup

We perform experiments on two datasets, *gMission* and *synthetic dataset*. gMission is an open source SC platform [Chen *et al.*, 2014], where each task is associated with its publish time, location and reward. Since gMission data is not associated with an expected completion time and penalty rate, we generate the attributes following uniform distribution. For synthetic dataset, based on the observation from real dataset, the location and the publish time of a task follow uniform distribution, and its maximum reward follows Gaussian distribution.

We evaluate performance of the following algorithms: 1) OTA: Optimal Task Assignment; 2) GTA: basic Greedy Task Assignment; 3) GTA + CT: GTA with Coarse Tuning; 4) GTA + FT: GTA with Fine Tuning; 5) GTA-RTO: GTA with Random Tuning Optimization (including coarse and fine tuning); 6) k-MTA: Maximum Task Assignment [Kazemi and Shahabi, 2012] with Minimum Cost Maximum Flow technique. In MTA, the weight between the worker (w) vertex and task (s) vertex is set as $\frac{1}{d(w.l, s.l)}$, and capacity between the task vertex and the fictitious destination vertex is set to k ($k = 1, 2, 3$). Two metrics are compared among the algorithms: *Profit-gaining Ratio* (*PR*, the ratio between the real and optimal total platform profit) and *CPU time* for finding the task assignment. All the algorithms are implemented on an Intel Core i5-2400 CPU @ 3.10G HZ with 8 GB RAM. The default values of all parameters are summarized in Tab. 1.

Parameter	Default value
Number of tasks	500 (gMission), 5000 (synthetic)
Number of workers	500 (gMission), 5000 (synthetic)
Early stop round n	10
Parameters in CT	p^{CT}_m = 0.2, p^{CT}_t = 0.4, p^{CT}_r = 0.4
Parameters in FT	p^{FT}_m = 0.4, p^{FT}_t = 0.6

Table 1: Experiment parameters

5.2 Experimental Results

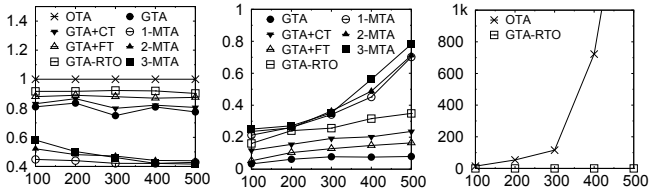
Effect of $|S|$. We first study the effect of $|S|$. In Fig. 3a and 4a, naturally OTA achieves the highest profit-gaining ratio (i.e., PR = 1), followed by GTA-RTO, GTA+FT, GTA+CT, GTA and k-MTA. GTA-RTO can improve the total profit by up to 25.56% when comparing with GTA. While the profits generated by all the MTA methods decline with the growth of $|S|$. In Fig. 3b, 3c, 4b and 4c, although the CPU cost of all methods increases as $|S|$ increases, the GTA related approaches perform better than the MTA approaches. OTA deteriorates much faster and cannot even return a result within tolerated time when $|S| > 400$ in gMission dataset and $|S| > 7000$ in synthetic dataset.

Effect of $|W|$. In Fig. 5a and 6a, GTA-RTO performs better than all the other greedy methods and MTA algorithms when $|W|$ varies. Fig. 5b, 5c, 6b and 6c depict that all greedy methods run faster than OTA and MTA, showing the superiority of the greedy methods.

Effect of p^{CT}_m. In this part we evaluate the effect of p^{CT}_m, a parameter in coarse tuning representing the basic probability when abandoning tasks. In Fig. 7a and 8a, GTA-RTO achieves the highest profit among the greedy methods especially when p^{CT}_m is a middle value, which reminds us that too conservative and too aggressive strategies are not suitable. In Fig. 7b and 8b, the CPU cost of coarse tuning increases when p^{CT}_m increases since a more radical exploration strategy is more likely to make more workers reassigned. Besides, the performance of GTA and GTA+FT k.pdf stable in Fig. 7 and 8 since only coarse tuning is affected by p^{CT}_m. Due to the similar results of gMission and synthetic dataset, we only report the results of synthetic dataset in the subsequent experiments.

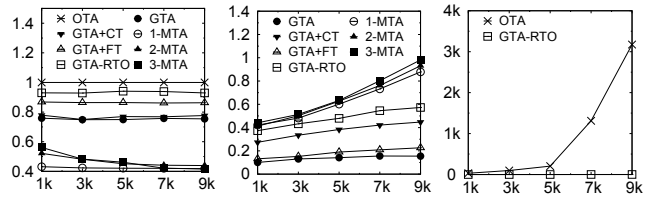
Effect of p^{FT}_m. We also evaluate the effect of p^{FT}_m, a basic probability parameter in fine tuning. In Fig. 9a, compared with the greedy approaches, GTA-RTO can obtain the highest profit. The CPU cost of fine tuning increases when p^{FT}_m increases (see Fig. 9b) with the same reason of the effect of p^{CT}_m. Moreover, the performance of GTA and GTA+CT does not change in Fig. 9 since only fine tuning is affected by p^{FT}_m.

Effect of n. Finally we study the effect of n, a parameter in GTA-RTO determining the termination condition. GTA-RTO will keep searching until the total profit stops improving for n iterations. In Fig. 10a, the profit of GTA increases when n grows since a higher n leads to a higher possibility to find a better result. Obviously, the CPU cost of GTA increases in Fig. 10b as n getting enlarged, since a higher n means more iterations during searching process.



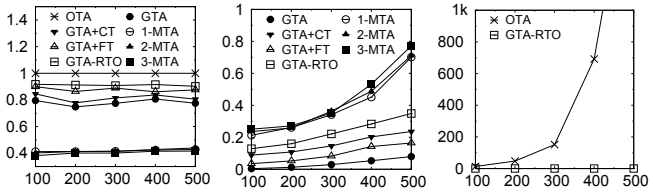
(a) PR (b) CPU cost (c) CPU cost

Figure 3: Effect of $|S|$ on gMission dataset



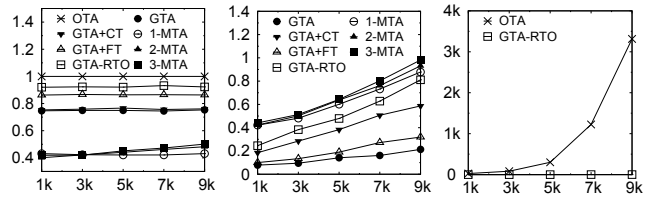
(a) PR (b) CPU cost (c) CPU cost

Figure 4: Effect of $|S|$ on synthetic dataset



(a) PR (b) CPU cost (c) CPU cost

Figure 5: Effect of $|W|$ on gMission dataset

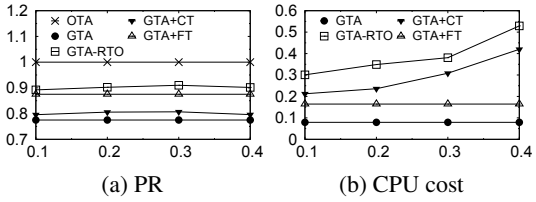


(a) PR (b) CPU cost (c) CPU cost

Figure 6: Effect of $|W|$ on synthetic dataset

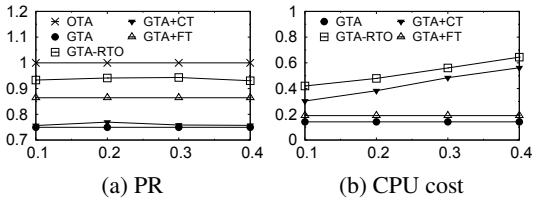
Cost and profit analysis. We compute the reward loss $s.rl$ (caused by overtime) for each task, i.e., $s.rl = (s.t_c - s.e) \times s.pr$ (when $s.t_c > s.e$), where $s.t_c$ is task's completion time, and further get the average reward loss, $\bar{rl}$. For the task s that cannot obtain maximum reward from the requester, the reward loss shows a linearly growth trend with the increasing $s.t_c$, matching the task reward pricing model. GTA with a low $\bar{rl}$ can only assign a small number of tasks and MTA with a high $\bar{rl}$ makes more tasks assigned, each of which generates a low platform profit. However, OTA and GTA-RTO can achieve a high profit by trading off various factors comprehensively.

Summary of our empirical study. Our empirical study can be summarize as follows: 1) OTA achieves the maximum profit but sacrifices a great deal of efficiency; 2) GTA-RTO achieves good balance between efficiency and effectiveness.



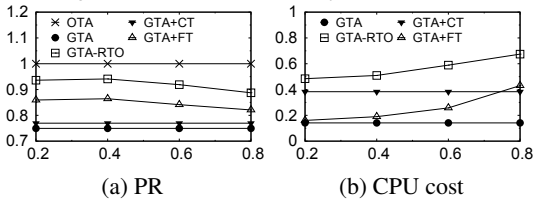
(a) PR (b) CPU cost

Figure 7: Effect of p_m^{CT} on gMission dataset



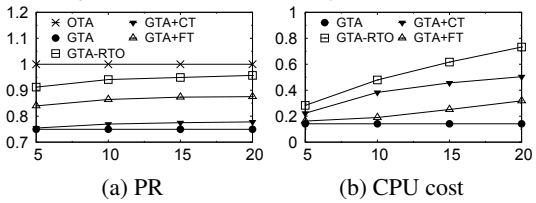
(a) PR (b) CPU cost

Figure 8: Effect of p_m^{CT} on synthetic dataset



(a) PR (b) CPU cost

Figure 9: Effect of p_m^{FT} on synthetic dataset



(a) PR (b) CPU cost

Figure 10: Effect of n on synthetic dataset

6 Conclusions

In this paper we study a novel SC problem, called Profit-driven Task Assignment (PTA), to find the optimal task assignment with maximal profit for SC platform. In order to achieve high effectiveness and efficiency, we addressed a few challenges by designing a reward pricing model to quantify the relationship between the task reward and its completion time, and developing optimal and efficient algorithms to assign tasks. To the best of our knowledge, it is the first work in SC that maximizes the profit from the point of SC platform when assigning tasks. Extensive empirical study based on both real and synthetic datasets confirms the superiority of our proposed algorithms.

Acknowledgements

This work was partially supported by NSFC (No. 61836007, 61832017, 61532018, 61525205 and 61872258), Alibaba Innovation Research (AIR) and A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD).

References

[Chen *et al.*, 2014] Zhao Chen, Rui Fu, Ziyuan Zhao, Zheng Liu, Leihao Xia, Lei Chen, Peng Cheng, Caleb Chen Cao, Yongxin Tong, and Chen Jason Zhang. gmission: A general spatial crowdsourcing platform. *VLDB*, 7(13):1629–1632, 2014.

[Cheng *et al.*, 2015a] Peng Cheng, Xiang Lian, Lei Chen, and Jinsong Han. Task assignment on multi-skill oriented spatial crowdsourcing. *TKDE*, 28(8):2201–2215, 2015.

[Cheng *et al.*, 2015b] Peng Cheng, Xiang Lian, Zhao Chen, Rui Fu, Lei Chen, Jinsong Han, and Jizhong Zhao. Reliable diversity-based spatial crowdsourcing by moving workers. *VLDB*, 8(10):1022–1033, 2015.

[Cooper *et al.*, 2011] Seth Cooper, Firas Khatib, Ilya Makedon, Hao Lu, Janos Barbero, David Baker, James Fogarty, Zoran Popović, et al. Analysis of social gameplay macros in the foldit cookbook. In *FDG*, pages 9–14, 2011.

[Deng *et al.*, 2015] Dingxiong Deng, Cyrus Shahabi, and Linhong Zhu. Task matching and scheduling for multiple workers in spatial crowdsourcing. In *SIGSPATIAL*, page 21, 2015.

[Jain *et al.*, 2009] Shaili Jain, Yiling Chen, and David C. Parkes. Designing incentives for online question and answer forums. In *EC*, pages 129–138, 2009.

[Kazemi and Shahabi, 2012] Leyla Kazemi and Cyrus Shahabi. Geocrowd: Enabling query answering with spatial crowdsourcing. In *SIGSPATIAL*, pages 189–198, 2012.

[Li *et al.*, 2015] Yu Li, Manlung Yiu, and Wenjian Xu. Oriented online route recommendation for spatial crowdsourcing task workers. *SSTD*, pages 137–156, 2015.

[Shah-Mansouri and Wong, 2015] Hamed Shah-Mansouri and Vincent W. S. Wong. Profit maximization in mobile crowdsourcing: A truthful auction mechanism. In *ICC*, pages 3216–3221, 2015.

[Singer and Mittal, 2013] Yaron Singer and Manas Mittal. Pricing mechanisms for crowdsourcing markets. In *WWW*, pages 1157–1166, 2013.

[Song *et al.*, 2017] Tianshu Song, Yongxin Tong, Libin Wang, Jieying She, Bin Yao, Lei Chen, and Ke Xu. Trichromatic online matching in real-time spatial crowdsourcing. In *ICDE*, pages 1009–1020, 2017.

[Tong *et al.*, 2016a] Yongxin Tong, Jieying She, Bolin Ding, Lei Chen, Tianyu Wo, and Ke Xu. Online minimum matching in real-time spatial data: Experiments and analysis. *VLDB*, 9(12):1053–1064, 2016.

[Tong *et al.*, 2016b] Yongxin Tong, Jieying She, Bolin Ding, and Libin Wang. Online mobile micro-task allocation in spatial crowdsourcing. In *ICDE*, pages 49–60, 2016.

[Tong *et al.*, 2017] Yongxin Tong, Libin Wang, Zimu Zhou, Bolin Ding, Lei Chen, Jieping Ye, and Ke Xu. Flexible online task assignment in real-time spatial data. *VLDB*, 10(11):1334–1345, 2017.

[Tong *et al.*, 2018a] Yongxin Tong, Libin Wang, Zimu Zhou, Lei Chen, Bowen Du, and Jieping Ye. Dynamic pricing in spatial crowdsourcing: A matching-based approach. In *SIGMOD*, pages 773–788, 2018.

[Tong *et al.*, 2018b] Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Lei Chen, Jieping Ye, and Ke Xu. A unified approach to route planning for shared mobility. *VLDB*, 11(11):1633–1646, 2018.

[Vazirani, 2013] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[Yang *et al.*, 2012] Dejun Yang, Guoliang Xue, Fang Xi, and Tang Jian. Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing. In *MobiCom*, pages 173–184, 2012.

[Zhao *et al.*, 2017] Yan Zhao, Yang Li, Yu Wang, Han Su, and Kai Zheng. Destination-aware task assignment in spatial crowdsourcing. In *CIKM*, pages 297–306, 2017.

[Zhao *et al.*, 2019a] Yan Zhao, Jinfu Xia, Guanfeng Liu, Han Su, Defu Lian, Shuo Shang, and Kai Zheng. Preference-aware task assignment in spatial crowdsourcing. In *AAAI*, pages 80–87, 2019.

[Zhao *et al.*, 2019b] Yan Zhao, Kai Zheng, Yang Li, Han Su, Jiajun Liu, and Xiaofang Zhou. Destination-aware task assignment in spatial crowdsourcing: A worker decomposition approach. *TKDE*, pages 219–233, 2019.