



Extra Budget-Aware Online Task Assignment in Spatial Crowdsourcing

Lun Jin¹, Shuhan Wan¹, Detian Zhang^{1(✉)}, and Ying Tang²

¹ Institute of Artificial Intelligence, School of Computer Science and Technology, Soochow University, Suzhou, China

{ljin,20195227042}@stu.suda.edu.cn, detian@suda.edu.cn

² Yibin Cowin Automobile Co., Ltd., Yibin, China

tangying@mychery.com

Abstract. With the proliferation of mobile devices, wireless networks and sharing economy, spatial crowdsourcing (e.g., ride hailing, food delivery, citizen sensing services) is becoming popular recently. In spatial crowdsourcing (SC), workers need to physically move to specific locations to conduct the assigned tasks which incurs travel cost. In this paper, we propose a novel SC problem, namely extra budget-aware online task assignment (EBOTA), where tasks have extra budget to subsidize the extra travel cost of workers. EBOTA concerns the strategy of assigning each task to proper worker such that the total satisfaction of completed tasks can be maximized. To address the EBOTA problem, we first propose an efficient algorithm called Deadline-exact algorithm which always computes the optimal assignment for the newly appearing object. Because of Deadline-exact's high time complexity which may limit its feasibility in real world, we propose another two practical algorithms, i.e., Threshold-based algorithm and Priority-based algorithm. Finally, we verify the effectiveness and efficiency of the proposed methods through extensive experiments on real dataset.

Keywords: Spatial crowdsourcing · Online task assignment · Extra budget

1 Introduction

With the popularization of GPS-equipped smart devices and the development of high-speed wireless networks (e.g. 5G), spatial crowdsourcing has drawn increasing attention in recent years. Typical spatial crowdsourcing services include ride hailing(e.g. Uber and Didi), food delivery(e.g. Ele.me and GrubHub) and citizen sensing services(e.g. OpenStreetMap).

As there are massive tasks and workers in spatial crowdsourcing platforms, one of the main problems is how to assign these tasks to workers properly. Existing works focus on assigning tasks to workers according to different goals, such as maximizing the total number of completed tasks [1, 22], maximizing expected

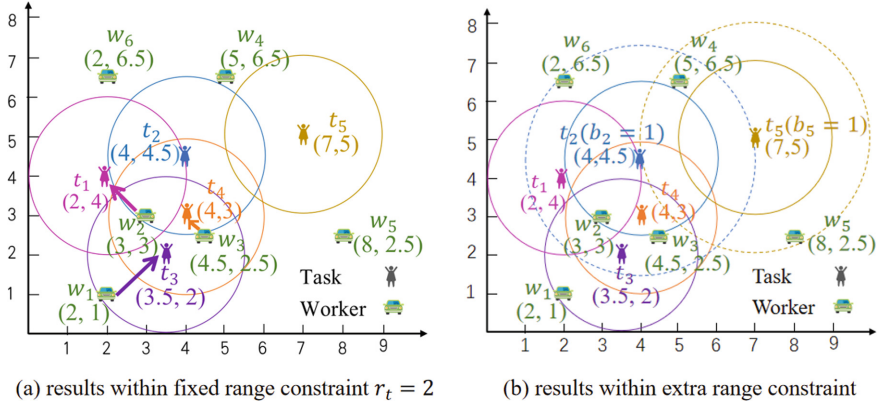


Fig. 1. Tasks without vs with extra budget

total payoff [20, 23], minimizing maximum delay [5]. An implicit assumption shared by these works is that a worker is only willing to perform tasks within his spatial vicinity because the worker needs to move to the location of these tasks, which may lead some remote tasks cannot be assigned. For example, Fig. 1(a) shows the locations of 5 tasks and 6 workers. Each circle centered by a task indicates that task can only be assigned to workers within the circle. If all of tasks only have such fixed range constraint, only 3 pairs can be matched, i.e., (t_1, w_2) , (t_3, w_1) and (t_4, w_3) , while t_2 and t_5 cannot be assigned to any workers.

Unfortunately, most of existing studies only consider that task requesters have such fixed range constraint or budget [6, 11, 17], which cannot solve this problem. The most closest work is incentive mechanism [2, 19, 21, 25], which formulates the pricing strategy to attract workers to participate. However, not every task requester is willing to accept raising price, and some remote tasks still may be not assigned if the newly attracted workers are far from them.

To overcome this challenge, we propose Extra-Budget Aware Online Task Assignment (EBOTA) problem. In our problem: (1) different tasks have different extra budget to subsidize and attract remote workers, and the SC platform will conduct task assignment not only based on the fixed but also the extra budget of tasks. A task requester can raising his extra budget to increase the accomplishment probability of the task. In the above example, if t_2 and t_5 have extra budget to subsidize workers (i.e., $b_2 = b_5 = 1$), the platform will increase their range constraint, i.e., the dash circles as shown in Fig. 1b, which makes w_4 and w_5 can serve them, respectively. (2) online dynamic task assignment is considered. Since in some real-time spatial crowdsourcing services, tasks and workers arrive dynamically and their temporal and spatial information are only known when they are arrived. Task assignment needs to be performed before the tasks and workers leave to maximize the total satisfaction.

In general, we make the following contributions in this paper:

- We formally define the Extra-Budget Aware Online Task Assignment (EBOTA) problem. As far as we know, we are the first one to propose this problem.
- We propose three effective algorithms to solve the problem, i.e., Deadline-exact algorithm, Threshold-based algorithm, and Priority-based algorithm.
- We conduct extensive experiments to prove the effectiveness and efficiency of our algorithms.

The rest of this paper is organized as follows. Section 2 formally defines the EBOTA problem. The optimal algorithm is proposed in Sect. 3 and online algorithms are proposed in Sect. 4. Extensive experiments on real dataset are presented in Sect. 5. Section 6 reviews some related work. Finally, Sect. 7 concludes this work.

2 Problem Definitions

Definition 1. (*Task*) A task, denoted by $t = \langle l_t, a_t, d_t, r_t, b_t \rangle$, appears on the platform with location l_t in the 2D space at time a_t and its required deadline is d_t . r_t is the radius of t centered in l_t which is the fixed range constraint of t , and b_t is its provided extra budget.

The fixed range constraint r_t is the default range for all tasks given by the SC platform. The extra budget is used for subsidizing the extra travel cost of workers, so b_t is proportion to the extra range constraint. In this paper, we assume the proportion is at the ratio of 1. For simplicity, we directly take b_t as the extra range constraint in the rest of the paper. Therefore, a task can only be conducted by the workers within the range $r_t + b_t$.

Definition 2. (*Worker*) A worker, denoted by $w = \langle l_w, a_w, d_w, u_w \rangle$, is released on the platform at time a_w and at location l_w in the 2D space, and its service deadline is d_w . His service score is u_w which is graded by the task requester after completing the task.

Definition 3. (*Travel cost*) The travel cost, denoted by $cost(t, w)$, is determined by the travel distance from l_w to l_t .

Travel distance, which can be measured by any types of distance such as Euclidean distance or road network distance. In this paper, we use Euclidean distance as the travel distance and take it as the travel cost directly for simplicity.

Definition 4. (*Extra travel cost*) Extra travel cost, denoted by e_t , is the actual travel cost exceeding the fixed range constraint provided by the task requester, i.e., $e_t = cost(t, w) - r_t$.

Since each task has an extra budget b_t , the extra travel cost e_t should be no more than it, i.e., $e_t \leq b_t$. In this paper, as the travel cost is measured by the Euclidean distance, we have $e_t = cost(t, w) - r_t = EDist(t, w) - r_t$. After

the worker completing the task, the task requester will score him according to the satisfaction with his service. Since the extra travel cost of the task needs to be paid by the task requester, the satisfaction function considers not only the service score of workers, but also the extra travel cost of the task. With the increase of extra travel cost, the satisfaction becomes smaller. Therefore, we define the satisfaction based on the worker service score u_w and the extra travel cost e_t as below:

Definition 5. (*Satisfaction*) Satisfaction is the satisfaction of the task requester who releases the task t to his matched worker w . It is denoted by:

$$S(t, w) = \begin{cases} u_w, & b_t = 0, \\ u_w \times (1 - \frac{e_t}{b_t}), & b_t > 0. \end{cases} \tag{1}$$

In addition, we can add other factors to the satisfaction according to other needs.

Definition 6. (*Extra-Budget Aware Online Task Assignment (EBOTA) problem*) Given a set of tasks T and a set workers W , each task or worker arrives sequentially. EBOTA calculates a feasible matching result M whose total satisfaction is maximized, i.e., $Maxsum(M) = \sum_{t \in T, w \in W} S(t, w)$, and is subjected to the following constraints:

1. Extra budget constraint, i.e., $e_t \leq b_t$.
2. Deadline constraint, i.e., $a_t < d_w$ and $a_w < d_t$.
3. Range constraint, i.e., $cost(t, w) \leq r_t + b_t$.

3 The Optimal Algorithm

In this section, we introduce the optimal solution for the EBOTA problem, where the platform has acquired the entire spatial and temporal information of workers and tasks before matching. Therefore, the optimal solution can not be applied to the online scenario. We take the optimal solution as one of the compared algorithms to show the efficiency of our proposed algorithms.

Given a set of tasks $T = \{t_1, t_2, \dots\}$ and a set of workers $W = \{w_1, w_2, \dots\}$, we construct a bipartite graph $G = (V, E)$ with V as the set of vertices, and E as the set of edges. The set V contains $|T| + |W|$ vertices including task vertices and worker vertices. The set E contains all possible edges from task vertices to worker vertices, i.e., if all three constraints in Definition 6 are satisfied between task t and worker w , there is an edge (t, w) from t to w , and its weight is $S(t, w)$ defined in Definition 5. Then, we can use an existing flow algorithm (e.g., Kuhn-Munkres (KM) algorithm [12]) to obtain the optimal result. The total amount of pairs is $min(|T|, |W|)$ and only the weight of which greater than 0 are the final matched pairs. The whole procedure of Optimal algorithm is illustrated in Algorithm 1.

Take Fig. 1(b) for example, we first construct a bipartite graph for all possible workers and tasks based constraints in Definition 6. Table 1 presents the arrival

Algorithm 1 Optimal algorithm

Input: $T, W, S(.,.)$
Output: the matched pair set M

- 1: $M \leftarrow \emptyset$
- 2: **for** each task node $t_i \in T$ **do**
- 3: **for** each worker node $w_j \in W$ **do**
- 4: **if** (t_i, w_j) satisfies all constraints **then**
- 5: add an edge between t_i and w_j , $weight(t_i, w_j) \leftarrow S(t_i, w_j)$
- 6: **else**
- 7: add an edge between t_i and w_j , $weight(t_i, w_j) \leftarrow 0$
- 8: $M \leftarrow KM(T, W)$
- 9: **return** M

Table 1. Arrival order

| Arrival order | t_3 | w_2 | t_1 | t_4 | w_1 | t_2 | w_3 | w_6 | t_5 | w_5 | w_4 |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| Arrival time | 8:01 | 8:02 | 8:03 | 8:05 | 8:06 | 8:08 | 8:10 | 8:11 | 8:12 | 8:14 | 8:15 |
| Leaving time | 8:08 | 8:09 | 8:10 | 8:12 | 8:13 | 8:15 | 8:17 | 8:18 | 8:19 | 8:21 | 8:22 |

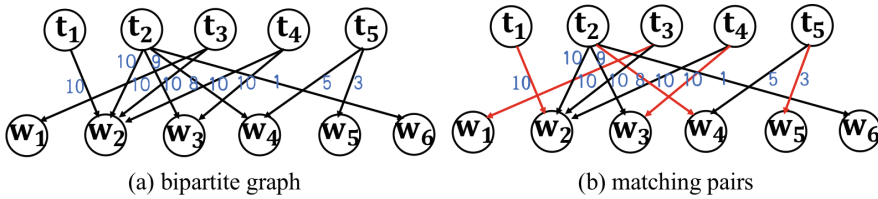


Fig. 2. The key steps of solving optimal algorithm

and leaving time of tasks and workers, and we assume the service score of each worker is 10, i.e., $u_w = 10$. Since only three workers, i.e., w_1, w_2 and w_3 , are online during t_1 's active time period. However, w_1 and w_3 are out of the range of t_1 , while w_2 is in the range, so there is only one edge connected, i.e., (t_1, w_2) . The weight of (t_1, w_2) is the satisfaction between this pair, i.e., $S(t_1, w_2) = u_2 = 10$ (as $b_1 = 0$). From Table 1, we can see all six workers are online during t_2 's active time period. With the help of extra budget $b_2 = 1$, the range constraint of t_2 is increased from $r_t = 2$ to $r_t + b_2 = 3$, which makes w_2, w_3, w_4 and w_6 are in the range. Therefore, four edges are connected, i.e., (t_2, w_2) , (t_2, w_3) , (t_2, w_4) and (t_2, w_6) . Since $cost(t_2, w_2) < r_t$, i.e., $e_t = 0$, we have $S(t_2, w_2) = u_2 = 10$. While $cost(t_2, w_3) > r_t$, we have $S(t_2, w_3) = u_3 \times (1 - \frac{cost(t_2, w_3) - r_t}{b_3}) = 9$. Similarly, we can compute $S(t_2, w_4) = 8$ and $S(t_2, w_6) = u_6 = 1$. By the same way, the edges for t_3, t_4 and t_5 can be constructed as shown in Fig. 2(a).

After the whole bipartite graph have been built, the KM algorithm is run to get the assignment. The allocation result is shown by the red line in Fig. 2(b), i.e., $\{(t_1, w_2), (t_2, w_4), (t_3, w_1), (t_4, w_3), (t_5, w_5)\}$. Therefore, the total satisfaction is $S = S(t_1, w_2) + S(t_2, w_4) + S(t_3, w_1) + S(t_4, w_3) + S(t_5, w_5) = 10 + 8 + 10 + 10 + 3 = 41$.

Complexity Analysis. The time and space complexity of the Optimal algorithm are $O(\max(|T|^3, |W|^3))$ and $O(|T||W|)$ respectively.

4 Online Algorithms

4.1 Greedy Algorithm

Greedy is a straightforward solution for EBOTA. It always chooses a pair with the highest satisfaction when a new object arrives, so the result of Greedy is local optimal. The performance of Greedy is susceptible to the order of tasks' and workers' appearance.

In our running example, when t_3 arrives, there are no workers (see Table 1). When w_2 arrives, there is only a task t_3 online and w_2 is in the range of it (see Fig. 1b). Therefore, t_3 is assigned to w_2 . When t_1 and t_4 arrive, there are no workers until w_1 arrives, but it is not in their range. Similarly, there is no worker in t_2 's range. When w_3 arrives, as the current task set is $T = \{t_1, t_2, t_4\}$ and it is in the range of t_2 and t_4 , so w_3 can be assigned to t_2 or t_4 . Since $S(t_2, w_3) = 9 < S(t_4, w_3) = 10$, (t_4, w_3) is selected by the Greedy. Using the same way, we can get the other matched pairs (t_2, w_6) , (t_5, w_5) . Therefore, the total satisfaction of Greedy is $S = 10 + 10 + 1 + 3 = 24$.

Complexity Analysis. For each new arrival object, the time and space complexity of the Greedy algorithm are $O(\max(|T|, |W|))$ and $O(\max(|T|, |W|))$.

4.2 Deadline-Exact Algorithm

Since Greedy is local optimal, we propose Deadline-exact algorithm to try to obtain the global optimal matching. The main idea is to determine the assignment between tasks and workers only at their deadline. When a task or worker reaches its deadline, we try to get the global optimal assignment for it.

Algorithm 2 presents the detailed steps of Deadline-exact algorithm. In line 1, the result set M , the unassigned tasks set T' and the unassigned workers set W' are set empty. In line 2, when an object o reaches its deadline d , the algorithm finds the assignment for it. In lines 3–7, the algorithm finds each unassigned object o' who arrives before time d . If the object o' is a task, the algorithm adds it to T' . If the object o' is a worker, the algorithm adds it to W' . In line 8, the KM algorithm is run based on current T' and W' to match tasks and workers. If o is a task and matched in M' , we assign it to the worker who is matched with it in M' as shown in lines 9–14. If o is a worker and matched in M' , we assign it to the task who is matched with it in M' as shown in lines 15–20. In line 21, the algorithm returns the final matched pair set M .

Back to our running example, t_3 is the first one to reach the deadline (i.e., 8:08, see Table 1). Currently, the unassigned task and worker sets are $T' = \{t_1, t_2, t_3, t_4\}$ and $W' = \{w_1, w_2\}$, respectively. After running KM algorithm, t_3 is assigned to w_1 . By the same way, t_1 is assigned to w_2 , t_4 is assigned to w_3 , t_2 is assigned to w_4 and t_5 is assigned to w_5 . The total satisfaction is $S = 10 + 10 + 10 + 8 + 3 = 41$.

Algorithm 2 Deadline-exact algorithm

Input: $T, W, S(.,.)$
Output: the matched pair set M

- 1: $M \leftarrow \emptyset$, the unassigned tasks set $T' \leftarrow \emptyset$, the unassigned workers set $W' \leftarrow \emptyset$
- 2: **for** each object o reaches its the deadline d **do**
- 3: **for** each object o' arrives before time d **do**
- 4: **if** o' is a task **then**
- 5: add o' into T'
- 6: **else**
- 7: add o' into W'
- 8: $M' \leftarrow \text{KM}(T', W')$
- 9: **if** o is a task **then**
- 10: **if** o is matched in M' **then**
- 11: $w \leftarrow$ the worker assigned to o in M'
- 12: $M \leftarrow M \cup (o, w)$
- 13: remove w from W'
- 14: remove o from T'
- 15: **if** o is a worker **then**
- 16: **if** o is matched in M' **then**
- 17: $t \leftarrow$ the task assigned to o in M'
- 18: $M \leftarrow M \cup (t, o)$
- 19: remove t from T'
- 20: remove o from W'
- 21: **return** M

Complexity Analysis. For each new arrival object, the time and space complexity of the Deadline-exact algorithm are $O(\max(|T|^3, |W|^3))$ and $O(|T||W|)$.

4.3 Threshold-Based Algorithm

Deadline-exact algorithm takes high time complexity. Random-threshold greedy (Greedy-RT) [16] can alleviate the impact of the order of Greedy algorithm and improve the efficiency of Deadline-exact algorithm. Inspired by this algorithm, we devise Threshold-based algorithm. Threshold-based algorithm first produces a random threshold τ , then the pair whose satisfaction is lower than τ is denied.

The details of Threshold-based algorithm is shown in Algorithm 3. In line 1, the result set M , the unassigned tasks set T' and the unassigned workers set W' are set empty. In lines 2–4, the algorithm randomly chooses a threshold according to the maximum satisfaction S_{max} [16], which can be got from historical data. In line 5, we iteratively process each new arrival object. If the object is a task (lines 6–13), the algorithm filters a worker subset W^* where each worker w satisfies all constraints and the satisfaction of the pair (o, w) exceeds the threshold. If W^* is not empty, the algorithm chooses the worker who has the highest satisfaction with o . If W^* is empty, the algorithm adds the object to the unassigned tasks set. If the object is a worker (lines 14–21), the algorithm filters a task subset T^* where each task t satisfies all constraints and satisfaction of the pair (o, t)

Algorithm 3 Threshold-based algorithm

Input: $T, W, S(\cdot, \cdot)$
Output: the matched pair set M

- 1: $M \leftarrow \emptyset$; the unassigned tasks set $T' \leftarrow \emptyset$; the unassigned workers set $W' \leftarrow \emptyset$
- 2: $\theta \leftarrow \lceil \ln(S_{max} + 1) \rceil$
- 3: $k \leftarrow$ randomly choosing an integer from $\{0, 1, \dots, \theta - 1\}$ with probability $\frac{1}{\theta}$
- 4: $\tau \leftarrow e^k$
- 5: **for** each new arrival object o **do**
- 6: **if** o is a task **then**
- 7: $t \leftarrow o$
- 8: $W^* \leftarrow \{w | S(t, w) \geq \tau, e_t < b_t, a_t < d_w, a_w < d_t, cost(t, w) \leq r_t + b_t\}$
- 9: **if** $W^* \neq \emptyset$ **then**
- 10: $w \leftarrow \underset{w \in W^*}{argmax} S_{max}$
- 11: $M \leftarrow M \cup (o, w)$; remove w from W'
- 12: **else**
- 13: $T' \leftarrow T' \cup o$
- 14: **else**
- 15: $w \leftarrow o$
- 16: $T^* \leftarrow \{t | S(t, w) \geq \tau, e_t < b_t, a_t < d_w, a_w < d_t, cost(t, w) \leq r_t + b_t\}$
- 17: **if** $T^* \neq \emptyset$ **then**
- 18: $t \leftarrow \underset{t \in T^*}{argmax} S_{max}$
- 19: $M \leftarrow M \cup (t, o)$; remove t from T'
- 20: **else**
- 21: $W' \leftarrow W' \cup o$
- 22: **return** M

exceeds the threshold. If T^* is not empty, the algorithm chooses the task who has the highest satisfaction with o . If T^* is empty, the algorithm adds the object to the unassigned workers set. The algorithm returns the final matched pair set M in line 22.

In our running example, suppose $S_{max} = 10$, we have $\theta = \lceil \ln(10 + 1) \rceil = 3$ and $k = \{0, 1, 2\}$. If $k = 0$, the threshold is $e^0 = 1$. When t_3 arrives, no worker is online until w_2 arrives. Task t_3 is assigned to w_2 because $S(t_3, w_2) > 1$. The algorithm continuously assigns t_4 to w_3 , t_2 to w_6 and t_5 to w_5 . Finally, we can compute the total satisfaction is $S = 10 + 10 + 1 + 3 = 24$. Similarly, the total satisfaction are 31 and 28 if thresholds are e^1 and e^2 , respectively. Threshold-based algorithm randomly chooses one threshold, so the expectation of the total satisfaction is $S = \frac{24+31+28}{3} = 28$.

Complexity Analysis. For each new arrival object, the time and space complexity of Threshold-based algorithm are both $O(\max(|T|, |W|))$.

Algorithm 4 Priority-based algorithm with historical threshold

Input: $T, W, S(.,.)$
Output: the matched pair set M

- 1: $M \leftarrow \emptyset$, the unassigned tasks set $T' \leftarrow \emptyset$, the unassigned workers set $W' \leftarrow \emptyset$
- 2: $\tau \leftarrow \tau^*$
- 3: **for** each new arrival object o **do**
- 4: **if** o is a task **then**
- 5: $W^* \leftarrow \{w | S(o, w) \geq \tau, e_t < b_t, a_t < d_w, a_w < d_t, cost(t, w) \leq r_t + b_t\}$
- 6: **if** $W^* \neq \emptyset$ **then**
- 7: $w \leftarrow \underset{w \in W^*}{max} p_w$
- 8: $M \leftarrow M \cup (o, w)$; remove w from W'
- 9: **else**
- 10: $T' \leftarrow T' \cup o$
- 11: **else**
- 12: $T^* \leftarrow \{t | S(t, o) \geq \tau, e_t < b_t, a_t < d_w, a_w < d_t, cost(t, w) \leq r_t + b_t\}$
- 13: **if** $T^* \neq \emptyset$ **then**
- 14: $t \leftarrow \underset{t \in T^*}{max} p_t$
- 15: $M \leftarrow M \cup (t, o)$; remove t from T'
- 16: **else**
- 17: $W' \leftarrow W' \cup o$
- 18: **return** M

4.4 Priority-Based Algorithm with Historical Threshold

Because the thresholds of Threshold-based algorithm are randomly selected, different thresholds have different impacts on the results, the performance of the algorithm can not be guaranteed. However, the daily actions of people are similar [26], we can utilize the threshold θ^* according to historical data which achieves best result in history. In addition, when a new task/worker o arrives, the oldest unmatched worker/task o' better to be assigned to o first to avoid expiration. Therefore, we bring in the wait time priority as follows:

$$wait = \frac{a_o - a_{o'}}{d_{o'} - a_{o'}},$$

where a_o is the arrival time of the new task/worker o , i.e., the current time of the system, $a_{o'}$ and $d_{o'}$ are the arrival time and deadline of worker/task o' , respectively.

Besides, the pair with the high satisfaction is beneficial not only to the worker but also to the task requester. Therefore, both satisfaction and wait time decide the priority of the pair. The priority of the o' is computed as follows:

$$p_{o'} = \alpha \times \frac{S(o, o')}{S_{max}} + (1 - \alpha) \times wait,$$

where α is a system parameter and can be got from historical data that yields the best result.

Based on the above point, we devise Priority-based algorithm with historical threshold, as shown in Algorithm 4. In line 1, the result set M , the unassigned tasks set T' and the unassigned workers set W' are set empty. In line 2, the algorithm sets a threshold according to the historical threshold. In line 3, the algorithm iteratively processes each new arrival object. In lines 4–10, if the object is a task, the algorithm filters a worker subset W^* where each worker w satisfies all constraints and the satisfaction of the pair (o, w) exceeds the threshold. If W^* is not empty, the algorithm chooses the worker who has the highest priority p_w . If W^* is empty, the algorithm adds the object to the unassigned tasks set. In lines 11–17, if the object is a worker, the algorithm filters a task subset T^* where each task t satisfies all constraints and satisfaction of the pair (t, o) exceeds the threshold. If T^* is not empty, the algorithm chooses the task who has the highest priority p_t . If T^* is empty, the algorithm adds the object to the unassigned workers set. The algorithm returns the final matched pair set M in line 18.

Assume that the best threshold is 2 and α is 0.3 based on historical data. We use this threshold in our example. When t_3 arrives, no online worker can serve it. When w_2 arrives, the task t_3 is assigned to w_2 because this pair has the highest priority $p_3 = \alpha \times \frac{S(w_2, t_3)}{S_{max}} + (1 - \alpha) \times \frac{a_{w_2} - a_{t_3}}{d_{t_3} - a_{t_3}} = 0.3 \times \frac{10}{10} + (1 - 0.3) \times \frac{1}{7} = 0.4$. Using the same method, we assign t_4 to w_3 , t_5 to w_5 and t_2 to w_4 . Thus, we obtain the total satisfaction is $S = 10 + 10 + 3 + 8 = 31$.

Complexity Analysis. For each new arrival object, the time and space complexity of Priority-based algorithm are both $O(\max(|T|, |W|))$.

5 Experimental Study

5.1 Experiment Setup

We use the taxi data from Didi Chuxing, which contains order data in Chengdu from November 1 to November 30, 2016. We randomly select the data of orders from 9:00 am to 10:00 am on November 6th, 2016 which has 12753 tasks and 10892 workers. The order data has the information of pick-ups and drop-offs, start and end of billing time. We use the pick-up location and the starting time of billing as the location information and arrival time of the task respectively. Since the worker becomes available again after the passenger gets off the car, the drop off location is used as the location of the worker and the end billing time is used as the arrival time of the worker. We use the dataset of November 5th, 2016 to get the optimal threshold $\tau = 5.5$ and $\alpha = 0.2$ that achieve the best result for Priority-based algorithm. Table 2 depicts our experimental settings, where the default values of parameters are in bold font.

Performance Metrics. We use the following metrics to evaluate the effectiveness and efficiency of the proposed methods:

Table 2. Experiments settings

| Factor | Setting |
|----------------------------|---|
| Fixed range constraint r | 600, 800, 1000 , 1200, 1400 |
| Extra range constraint b | (400,600), (600,800), (800,1000), (1000,1200), (1200,1400) |
| Time period p | 7:00–8:00, 8:00–9:00, 9:00–10:00 , 10:00–11:00, 11:00–12:00 |
| Service score u | 10 |

- **Running Time.** The running time represents the total execution time of an algorithm.
- **Average Response Time of Tasks.** The response time of a task refers to the time it takes to be answered by the platform. If a task is not assigned to any worker eventually, its response time will equal to its deadline.
- **Total satisfaction.** The total satisfaction is the sum of the satisfaction of all the task requesters in the platform. Algorithms achieving higher satisfaction are better.
- **Average Satisfaction.** The average satisfaction represents the satisfaction of each task requester. For each task requester, the high satisfaction means she satisfies this matching.

Compared Algorithms. We evaluate the performance of the representative algorithms, i.e., Optimal algorithm (OPT), Greedy algorithm (Greedy), Deadline-exact algorithm (Exact), Threshold-based algorithm (Threshold) and Priority-based algorithm with historical threshold (Priority). All the algorithms are implemented in Java, run on a machine with Intel(R) Core (TM) i7-7700 CPU @ 3.60 GHz and 16 GB RAM.

5.2 Results on the Real Dataset

Effect of the Fixed Range Constraint. As shown in Fig. 3(a), the running time of Exact is obviously higher than the others due to the high time complexity of KM algorithm. As for average response time of tasks in Fig. 3(b), Greedy, Threshold and Priority outperform Exact as they always try to assign a task or a worker the moment it appears. Average response time of tasks of Exact is the highest since it determines the allocation between the workers and tasks only at their deadlines. In Fig. 3(c), we can observe that the total satisfaction increases reasonably as the fixed range constraint increases. This is because more workers will be located in the fixed range constraint of each task. Also, we can observe that Exact, Threshold and Priority are more effective than Greedy and Exact is only slightly worse than OPT. From Fig. 3(d), we can see that Priority is the most effective, the reason is that Priority filters low satisfaction pairs.

Effect of the Extra Range Constraint. As depicted in Fig. 4(a), Exact takes more time than other algorithms. From Fig. 4(b) we can see that Exact

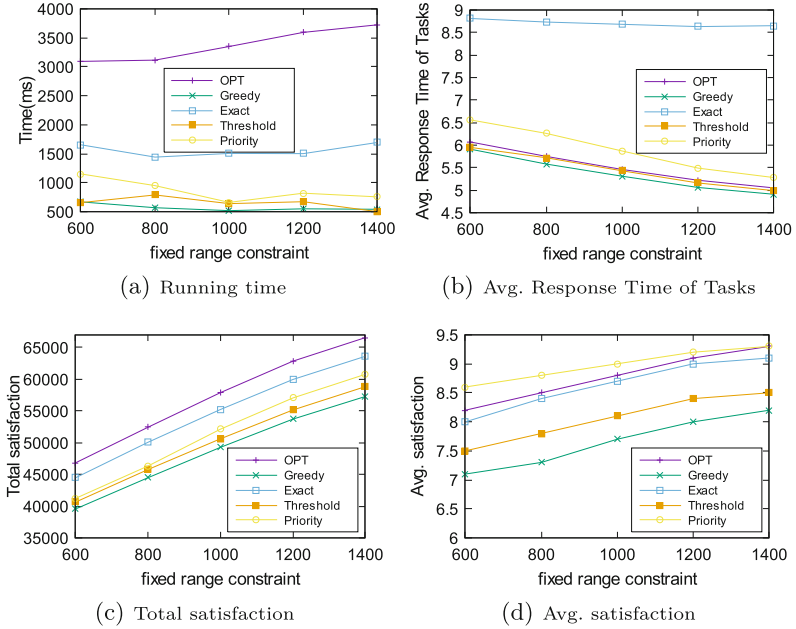


Fig. 3. Effect of the fixed range constraint on real dataset

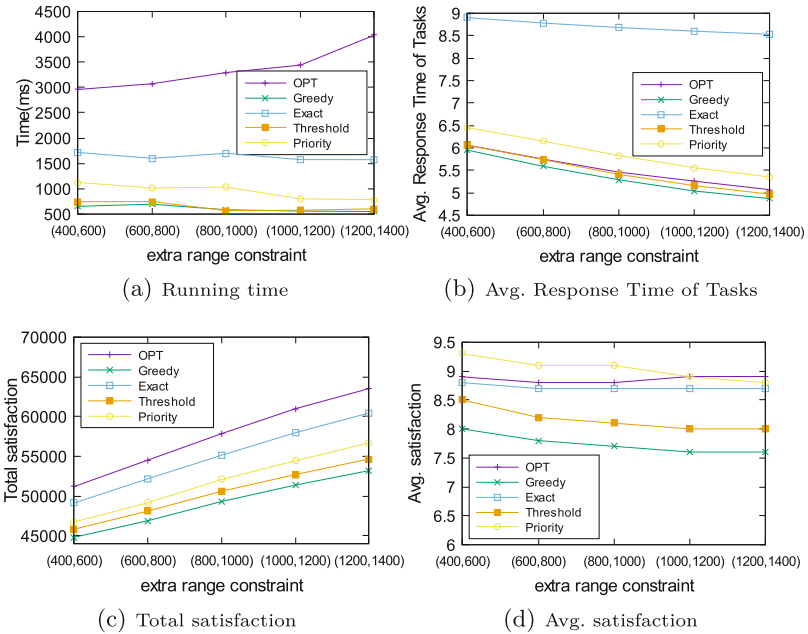


Fig. 4. Effect of the extra range constraint on real dataset

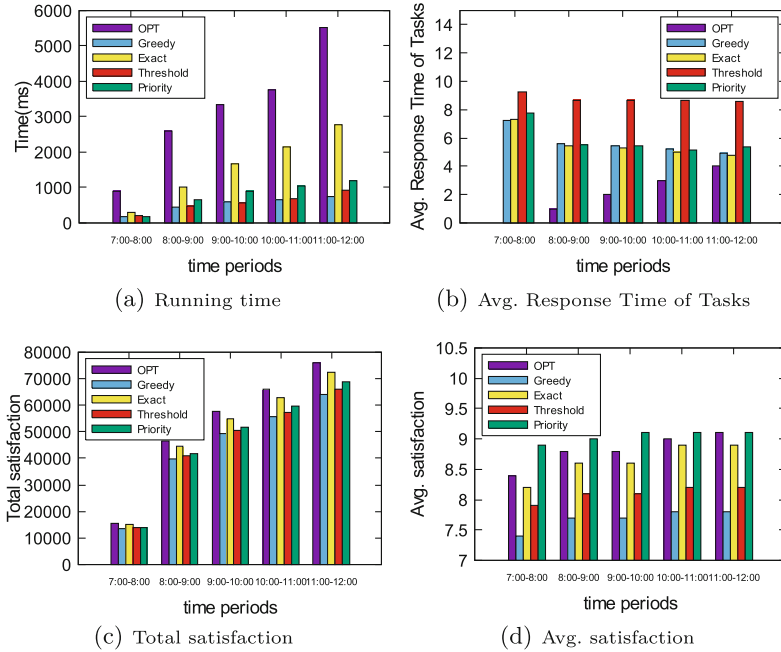


Fig. 5. Effect of different time periods in a day on real dataset

performs worse than others because the decision of assignment is often made at the deadline of tasks or workers. In Fig. 4(c), the total satisfaction increases as the extra range constraint increases which is reasonable as more workers will be in the extra range constraint of tasks. Exact outperforms the other algorithms because it can get global optimal result at the deadline of tasks or workers. Similarly, priority performs better than other algorithms in Fig. 4(d).

Table 3. The number of tasks and workers at different time periods

| Time period | 7:00–8:00 | 8:00–9:00 | 9:00–10:00 | 10:00–11:00 | 11:00–12:00 |
|-------------------------|-----------|-----------|------------|-------------|-------------|
| Number of tasks $ T $ | 5237 | 9199 | 11019 | 11670 | 12677 |
| Number of workers $ W $ | 3377 | 8145 | 9731 | 11052 | 12314 |

Effect of Different Time Periods. To show the effectiveness of algorithms in different time periods of a day, we conduct a set of experiments on different time periods. Table 3 shows the number of tasks and workers in different time periods. As we can see in Fig. 5(a), the number of tasks and workers at 11:00–12:00 is the largest and the running time is the longest. In Fig. 5(b), at 7:00–8:00,

the number of tasks and workers is the smallest and average response time of tasks is the longest because more tasks can not be answered before deadlines. In Fig. 5(c), total satisfaction at 11:00–12:00 is the highest, the reason is that the number of tasks and workers are the largest and more pairs can be matched. From Fig. 5(d) we can see that average satisfaction at different time periods is similar because the location distribution at different time is similar.

6 Related Work

6.1 Task Assignment in Spatial Crowdsourcing

Task assignment is one of the most important problems in spatial crowdsourcing [3, 14, 18]. It is divided into offline task assignment and online task assignment according to the arrival scenario of tasks and workers.

In the offline scenario [10, 17], the spatial and temporal information of workers and tasks is pre-known. Kazemi et al. [10] reduce the matching to the maximum flow problem [1], and use Ford-Fulkerson algorithm [9] to calculate the exact solution. In the online scenario, workers and tasks dynamically appear on the platform. The information of requests or workers is not pre-known. Some methods are proposed to maximize the overall utility (i.e., the total number of performed tasks or the total rewards of the assigned tasks) [4, 7, 14, 16, 20]. Ting et al. [16] propose Greedy-RT algorithm. They first randomly sample a threshold, and then match the pair whose weight is higher than the threshold. Tong et al. [20] first propose that each task and worker may appear at anytime and anywhere in spatial crowdsourcing. They also extend the Greedy-RT algorithm [16] to solve their problem. A threshold-based randomized framework is proposed to solve the problem. Song et al. [15] first consider that a task requires multiple workers to complete. Chen et al. [4] study the fair assignment of tasks to workers in spatial crowdsourcing. Cheng et al. [7] propose a cross online matching which enables a platform to borrow some unoccupied workers from other platforms. Tong and Liu et al. [11, 22] focus on maximizing the total number of completed tasks. Tong et al. [22] set that workers can move to other locations in advance and use the offline-guide-online technique [8] to increase the number of potential matches. Liu et al. [11] first propose that a task requester releases a batch of tasks, requiring workers to complete as many tasks as possible within his fixed budget for rewarding workers. Some methods have been proposed to minimize the waiting time [5, 13]. Chen et al. [5] first consider reducing the total waiting time of all tasks. Zhao et al. [13] first consider reducing the average waiting time of users, so as to improve the user experience. However, these studies have not focused on the tasks with extra budget. Wan et al. [24] consider tasks have extra budget but their task assignment is in offline scenario. We design task assignment algorithms assuming that the task has an extra budget in online scenario.

6.2 Incentive Mechanism in Spatial Crowdsourcing

The incentive mechanism problem determines the reward motivating workers to perform the tasks. In reality, supply and demand often change in space and

time [19]. The reward should be determined according to the dynamic supply and demand. While the platform determines the reward, the workers decide whether to accept it or not. In Uber, an effective solution is surge pricing. Banerjee et al. [2] use Markov process to determine the reward to workers. Tong et al. [21] define the Global Dynamic Pricing problem in spatial crowdsourcing which is challenging due to unknown demand, limited supply and dependent supply. These above works concentrate on online learning algorithms which determine the reward to workers based on the estimated expectation of workers. There are also many other incentive mechanisms focus on a different scenarios. For example, the auction mechanism determines the reward based on workers' submitted bids. Xiao et al. [25] consider that workers can make detours from their original travel paths to perform tasks and bid for his/her detour cost. However, these methods cannot be directly applied to our EBOTA, since they take the specific pricing strategy into account but did not consider how to allocate tasks. Moreover, the incentive mechanism determines the reward for workers to attract workers before assignment and our problem focuses on task assignment without pricing.

7 Conclusion

In this paper we propose a novel problem, i.e., Extra-Budget Aware Online Task Assignment (EBOTA), to assign the tasks to workers in real time by considering task requesters have extra budget. To solve the problem, we first propose Deadline-Exact algorithm, which always finds a global assignment for the task or worker at its deadline. However, because of the high time complexity of Deadline-Exact, it is not practical in real applications. We next propose Threshold-based algorithm which utilizes a random generated threshold to prune the pairs with small satisfaction, and Priority-based algorithm with historical threshold which learns near optimal threshold from historical data. Extensive experimental results over real dataset verify the effectiveness and efficiency of our approaches.

Acknowledgments. This work was supported by Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions, and by Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network flows (1988)
2. Banerjee, S., Freund, D., Lykouris, T.: Pricing and optimization in shared vehicle systems: an approximation framework. arXiv preprint [arXiv:1608.06819](https://arxiv.org/abs/1608.06819) (2016)
3. Chen, L., Shahabi, C.: Spatial crowdsourcing: challenges and opportunities. Bull. Tech. Committ. Data Eng. **39**(4), 14 (2016)
4. Chen, Z., Cheng, P., Chen, L., Lin, X., Shahabi, C.: Fair task assignment in spatial crowdsourcing. VLDB **13**(12), 2479–2492 (2020)

5. Chen, Z., Cheng, P., Zeng, Y., Chen, L.: Minimizing maximum delay of task assignment in spatial crowdsourcing. In: ICDE, pp. 1454–1465. IEEE (2019)
6. Cheng, P., Lian, X., Chen, L., Han, J., Zhao, J.: Task assignment on multi-skill oriented spatial crowdsourcing. TKDE **28**(8), 2201–2215 (2016)
7. Cheng, Y., Li, B., Zhou, X., Yuan, Y., Wang, G., Chen, L.: Real-time cross online matching in spatial crowdsourcing. In: ICDE, pp. 1–12. IEEE (2020)
8. Feldman, J., Mehta, A., Mirrokni, V., Muthukrishnan, S.: Online stochastic matching: beating $1-1/e$. In: FOCS, pp. 117–126. IEEE (2009)
9. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. Can. J. Math. **8**, 399–404 (1956)
10. Kazemi, L., Shahabi, C.: Geocrowd: enabling query answering with spatial crowdsourcing. In: SIGSPATIAL, pp. 189–198 (2012)
11. Liu, J.X., Xu, K.: Budget-aware online task assignment in spatial crowdsourcing. WWW **23**(1), 289–311 (2020)
12. Munkres, J.: Algorithms for the assignment and transportation problems. J. Soc. Ind. Appl. Math. **5**(1), 32–38 (1957)
13. Peng, W., Liu, A., Li, Z., Liu, G., Li, Q.: User experience-driven secure task assignment in spatial crowdsourcing. World Wide Web **23**(3), 2131–2151 (2020). <https://doi.org/10.1007/s11280-019-00728-3>
14. Song, T., et al.: Trichromatic online matching in real-time spatial crowdsourcing. In: ICDE, pp. 1009–1020. IEEE (2017)
15. Song, T., Xu, K., Li, J., Li, Y., Tong, Y.: Multi-skill aware task assignment in real-time spatial crowdsourcing. GeoInformatica **24**(1), 153–173 (2020)
16. Ting, H.F., Xiang, X.: Near optimal algorithms for online maximum edge-weighted b-matching and two-sided vertex-weighted b-matching. TCS **607**, 247–256 (2015)
17. To, H., Fan, L., Tran, L., Shahabi, C.: Real-time task assignment in hyperlocal spatial crowdsourcing under budget constraints. In: PerCom, pp. 1–8. IEEE (2016)
18. Tong, Y., Chen, L., Shahabi, C.: Spatial crowdsourcing: challenges, techniques, and applications. VLDB **10**(12), 1988–1991 (2017)
19. Tong, Y., et al.: The simpler the better: a unified approach to predicting original taxi demands based on large-scale online platforms. In: SIGKDD, pp. 1653–1662 (2017)
20. Tong, Y., She, J., Ding, B., Wang, L., Chen, L.: Online mobile micro-task allocation in spatial crowdsourcing. In: ICDE, pp. 49–60. IEEE (2016)
21. Tong, Y., Wang, L., Zhou, Z., Chen, L., Du, B., Ye, J.: Dynamic pricing in spatial crowdsourcing: a matching-based approach. In: SIGMOD, pp. 773–788 (2018)
22. Tong, Y., et al.: Flexible online task assignment in real-time spatial data. VLDB **10**(11), 1334–1345 (2017)
23. Tong, Y., Zeng, Y., Ding, B., Wang, L., Chen, L.: Two-sided online micro-task assignment in spatial crowdsourcing. TKDE **33**, 2295–2309 (2019)
24. Wan, S., Zhang, D., Liu, A., Fang, J.: Extra-budget aware task assignment in spatial crowdsourcing. In: WISE (2021)
25. Xiao, M., et al.: SRA: secure reverse auction for task assignment in spatial crowdsourcing. TKDE **32**(4), 782–796 (2019)
26. Zhao, Y., Zheng, K., Cui, Y., Su, H., Zhu, F., Zhou, X.: Predictive task assignment in spatial crowdsourcing: a data-driven approach. In: ICDE, pp. 13–24. IEEE (2020)