World Wide Web (2020) 23:289–311
https://doi.org/10.1007/s11280-019-00696-8

Budget-aware online task assignment in spatial crowdsourcing

Jia-Xu Liu[1,2] · **Ke Xu**[1]

Received: 7 August 2018 / Revised: 1 March 2019 / Accepted: 15 May 2019 /
Published online: 25 May 2019

Abstract

The prevalence of mobile internet techniques stimulates the emergence of various spatial crowdsourcing applications. Certain of the applications serve for the requesters, budget providers, who submit a batch of tasks and a fixed budget to platform with the desire to search suitable workers to complete the tasks in maximum quantity. Platform lays stress on optimizing assignment strategies on seeking less budget-consumed worker-task pairs to meet the requesters' demands. Existing research on the task assignment with budget constraints mostly focuses on static offline scenarios, where the spatiotemporal information of all workers and tasks is known in advance. However, workers usually appear dynamically on real spatial crowdsourcing platforms, where existing solutions can hardly handle it. In this paper, we formally define a novel problem called Budget-aware Online task Assignment(BOA) in spatial crowdsourcing applications. BOA aims to maximize the number of assigned worker-task pairs under budget constraints where workers appear dynamically on platforms. To address the BOA problem, we first propose an efficient threshold-based greedy algorithm called Greedy-RT which utilizes a random generated threshold to prune the pairs with large travel cost. Greedy-RT performs well in the adversarial model when compared with simple greedy algorithm, but it is unstable in the random model for its random generated threshold may produce poor quality in matching size. We then propose a revised algorithm called Greedy-OT which could learn near optimal threshold from historical data, and consequently improves matching size significantly in both models. Finally, we verify the effectiveness and efficiency of the proposed methods through extensive experiments on real and synthetic datasets.

Keywords Budget constraint · Online task assignment · Threshold · Greedy · Spatial crowdsourcing

✉ Jia-Xu Liu
liujiaxu@buaa.edu.cn

Ke Xu
kexu@nlsde.buaa.edu.cn

[1] State Key Laboratory of Software Development Environment, Beihang University, Beijing, China

[2] College of Software, Liaoning Technical University, Huludao, China

1 Introduction

The pervasion of mobile phones has caused the rapid emergence of various of spatial crowdsourcing applications such as Uber, Gigwalk, and Waze etc. in the last decade. Part of these applications feature with budget constraints and usually fit with the occasions like data acquisition [25] and sampling survey [18]. For instance, Gigwalk provides an inspection service for international cosmetics companies to investigate how their products are displayed in supermarkets and stores. This platform recruits workers to arrive at a number of geographically dispersed cosmetics counters to check the placement of their products on main shelves, end shelves, island cabinets and ground stacks etc., take photos from various angles and upload them to the platform. Once the uploaded photos are verified as valid and timely, the workers will earn corresponding remuneration from the platform. In this real application, a batch of tasks are released by a single requester who as well provides a fixed budget for rewarding workers. The requester expects the platform could expend the fund efficiently so that more tasks are able to be assigned even if the spatiotemporal information of workers is unknown util they appear on the platform. That is, the platform is required to conduct online task assignment to maximize the number of total assigned pairs under a fixed budget.

Previous studies on the problem of online task assignment can generally be divided into two categories: non-guidance and guidance. The majority studies lay in the scope of non-guidance matching. Multiple-armed bandit [4, 25] and random-threshold greedy [22, 27] are prevalently adopted to solve the problem. However, neither of them could approach optimal solution since the global spatiotemporal information cannot be acquired in advance to guide online matching. Tong et al. proposed a guided matching solution for the problem with prior knowledge which could be estimated from the historical traces of workers and tasks [28]. Higher performance can be achieved on the condition that the real spatial-temporal distributions of workers and tasks are highly similar with their historical distributions. Once the condition is broken when emergent events occur, a great deviation will be produced between historical and real data. As a result, much more inferior matching results are generated and even worse than the non-guidance methods in performance. Unfortunately, the deviation between real-time and historical distribution in the real world is frequently non-negligible over a brief period of time, even if the spatiotemporal distribution of workers can keep periodic similarity in the magnitude of long period. Thus, many unsuitable matching pairs will emerge if fine-grained historical information is directly utilized to guid real-time task assignment. Besides the hardness in usage of historical data, another obstacle is that existing online assignment algorithms have defects in improving matching size. For example, greedy is a simple and efficient algorithm in solving online task assignment problem [26]. However, its performance is sensitive to the arrival order of workers, especially fragile in the adversarial model. We go through the following toy example to illustrate it.

Example 1 David submits six tasks to platform with the desire that platform helps him assign as many workers as possible to his tasks while the payment does not exceed the budget 10. All the tasks share the same release time 0 and expired time 10 while the appearances of workers may differ in time. Here, we deploy that any worker w_i appears at time i. Platform performs one-to-one matching between workers and tasks, where each task is completed by one worker and each worker is assigned to one task at most. The initial locations of the tasks and workers are labeled in the 2D space(X, Y) in Figure 1a. We assume the reward for workers equals to their travel cost, and the total travel cost of workers can not beyond 10. The optimal matching is shown in Figure 1b while the result of the simple greedy algorithm is shown in Figure 1c. Here, the greedy solution is $\{(w_1, t_1), (w_2, t_2)\}$

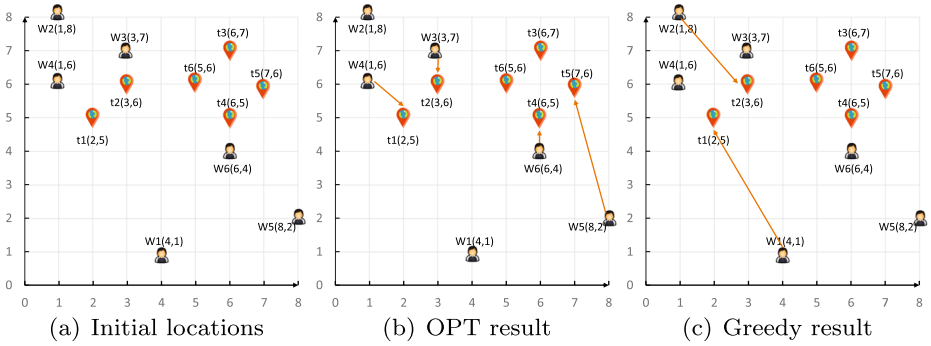


Figure 1 The matching results of the offline optimal algorithm and simple greedy algorithm

whose matching size is merely 2 and used budget is 10. There are two reasons for the unsatisfactory result: 1) bad arrival order of workers, which causes the budget is exhausted by those early arrival workers with large travel cost. 2) the greedy algorithm has no prior knowledge on the optimal matching which can be estimated from historical data and used to guid the real-time task assignment.

Motivated by the above example, we first formulate a novel problem called Budget-aware Online task Assignment(BOA). To address the BOA problem, we propose two greedy variants to amend the two defects of the simple greedy algorithm mentioned above. In order to economize budget, the first variant utilizes a random generated threshold to filter out those serious budget-expended matchings. The random thresholds cannot guarantee the quality of results since small thresholds maybe filter out those eligible matchings. Another variant ameliorates the former by learning a near optimal threshold from historical data. The main contribution of this work are summarized as follows.

- Inspired by certain emerging spatial crowdsourcing applications, we propose and formulate the BOA problem which is a problem of online task assignment in real-time spatial data with budget constraints.
- To address the BOA problem, we propose a greedy variant, called Greedy-RT, which filters out those bad matchings with large expensive cost by a random generated threshold, whose performance is superior than the simple greedy algorithm in the adversarial model.
- We then propose another variant, called Greedy-OT, which extracts a near optimal threshold from the offline optimal solution based on the spatiotemporal information of historical workers and released tasks. It can significantly improve the matching performance compared with Greedy-RT and the simple greedy algorithm both in the random model and adversarial model.
- We verify the effectiveness and efficiency of our algorithms on both synthetic dataset and large-scale Uber pickups dataset.

In the rest of this paper, we formulate the BOA problem in Section 2 and provide its offline optimal solution in Section 3. We present a greedy variant based on a random generated threshold and analyze its competitive ratio in Section 4.1, and then present a revised greedy variant based on a near optimal threshold and analyze its competitive ratio

in Section 4.2. Section 5 presents the performance evaluations, Section 6 reviews related worker and Section 7 concludes this paper.

2 Problem statement

In this section, we first introduce the basic concepts, and then formally define the Budget-aware Online task Assignment (BOA) problem.

Definition 1 (Worker) A worker is denoted by $w =< l_w, b_w, e_w, v_w >$. l_w is the location of w in a 2D space. b_w and e_w is the arrival time and departure time from platform. v_w is the velocity of w.

It is notable that b_w equals to e_w in our BOA problem, that is, when a new worker w arrives, platform will make an assignment determination for w instantly. Once w fails in matching, platform will not take him into consideration in subsequent assignments.

Definition 2 (Task) A task is denoted by $t =< l_t, r_t, d_t >$. l_t is the location of t in a 2D space. r_w and d_t is the release time and deadline of t.

Definition 3 (Task requester) A task requester is denoted by $r =< T, B >$. $T = \{t_1, ..., t_n\}$ is a batch of tasks released by a single requester. B is the budget that requester supplies to reward workers who can complete any task of T.

Definition 4 (Travel Cost) The travel cost, denoted by $cost(w, t)$, is the travel distance for w to arrive at the location of t, that is, the distance between l_w and l_t.

Definition 5 (BOA Problem) Given a task requester r, and a set of workers W where workers dynamically appear on platform one by one at any time. The goal of BOA is to find an assignment scheme M between W and $r.T$ to maximize the number of assigned pairs. That is, max $Sum(M) = \sum_{w \in W, t \in r.T} I(w, t)$, where $I(w, t) = 1$ if the pair (w, t) is matched in the assignment M, and otherwise $I(w, t) = 0$. Such the following constraints should be satisfied.

- Budget constraint. We assume the reward that platform supplies for a worker is proportion to his travel cost at the ratio of 1, then the total rewards paid for workers in M are less than or equal to B. (i.e.,$\sum_{(w,t) \in M} cost(w, t) \leq B$).
- Deadline constraint. For any worker-task pair (w, t), w must arrive at the location of t before its deadline. (i.e.,$b_w + cost(w, t)/v_w \leq d_t$).
- Invariable constraint. Once a task t is assigned to a worker w, the assignment of (w, t) cannot be revoked.

The theoretical guarantee on online algorithms is usually measured by competitive ratio, which describes the differences between the output of an online algorithm and its offline optimal result. In this paper, two different arrival orders of workers are simulated by the adversarial model and the random order model, which represent the worst case and the

general case, respectively. The corresponding competitive ratios of the two online models are defined as follows.

Definition 6 (CR in the adversarial order model) The competitive ratio of an online algorithm for the BOA problem in the adversarial model is the minimum ratio between the online algorithm and the optimal result over all possible arrival orders of workers.

$$CR = \min_{\forall G(W,T), \forall o \in O} \frac{Sum(M)}{Sum(OPT)}$$

Where $G(W, T)$ is an arbitrary input of tasks and workers, O is the set of all possible input orders, o is one order in O, $Sum(M)$ is the matching size that the online algorithm achieves, and $Sum(OPT)$ is the matching size of the offline optimal scheme.

Definition 7 (CR in the random order model) The competitive ratio of an online algorithm for the BOA problem in the random order model is

$$CR = \min_{\forall G(W,T), \forall o \in O} \frac{E[Sum(M)]}{Sum(OPT)}$$

Where $E[Sum(M)]$ is the expectation of the matching size produced by the online algorithm over all possible arrival orders.

3 Offline optimal solution

In this section, we introduce the optimal solution for the BOA problem, which can be solved in offline scenarios, where platform has acquired the entire and determined spatiotemporal information of workers and tasks before matching.

Theorem 1 *The optimal solution for the BOA problem is reducible to the minimum-cost maximum-flow problem.*

Proof Given $W = \{w_1, w_2, ...\}$ as the set of online workers, and $T = \{t_1, t_2, ...\}$as the set of tasks to be assigned. Let $G = (V, E)$ be the flow graph with V as the set of vertices, and E as the set of edges. The set V contains $|W| + |T| + 2$ vertices includeing worker vertices denoted by $V_{w_i}, i \in [1, |W|]$, task vertices marked by $V_{t_j}, j \in [0, |T|]$ and two additional vertices, source vertex V_s and destination vertex V_e respectively. The set E contains $|W| + |T| + |W| \cdot |T|$ edges, each of which has a capacity of 1. We associate the cost of edge from vertex V_{w_i} to V_{t_j} with their travel distance, and the one of other edges with 0. Thus, given a fixed budget, we may obtain the maximum matching in quantity by running the min-cost max-flow algorithm in G. □

We next describe the steps to obtain the optimal solution of the BOA problem. According to Theorem 1, we first utilize the offline spatial-temporal information of workers and tasks to build a bipartition graph, and then run traditional min-cost max-flow algorithm on

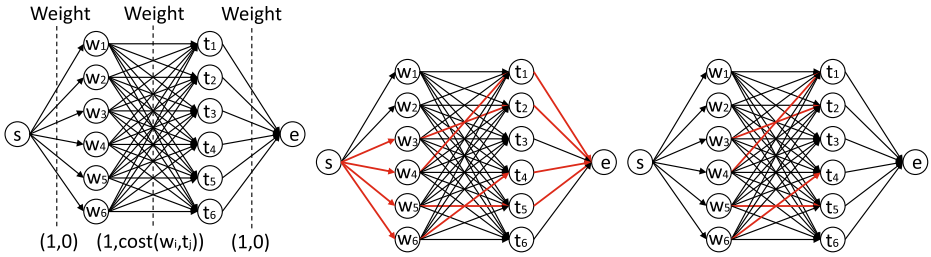
the graph to get the optimal matching $\hat{M}^*$. Finally, we sort the matching pairs in $\hat{M}^*$ in ascending order of their cost, and then choose them one by one until the total cost beyonds the budget. The whole procedure is depicted in Algorithm 1.

Algorithm 1 OPT algorithm.

Require: W, T, B
Ensure: matching scheme $\hat{M}^*$
1: $\hat{M}^* \leftarrow \emptyset, c \leftarrow 0$
2: create source vertex s, sink vertex e
3: **for all** worker node $w \in W$ **do**
4: add_edge(s, w,1,0)
5: **end for**
6: **for all** task node $t \in T$ **do**
7: add_edge(t, e,1,0)
8: **end for**
9: **for all** worker node $w \in W$ **do**
10: **for all** task node $t \in T$ **do**
11: **if** $b_w + cost(w,t)/v_w \leq d_t$ **then**
12: add_edge(w, t, 1, $cost(w,t)$);
13: **end if**
14: **end for**
15: **end for**
16: $\hat{M} \leftarrow$ Min-Cost_Max-Flow(s, e)
17: sort worker-task pairs in $\hat{M}$ by cost in ascending order
18: **for all** sorted worker-task pair $(w,t) \in \hat{M}$ **do**
19: **if** $c + cost(w,t) \leq B$ **then**
20: insert (w,t) into $\hat{M}^*$
21: $c \leftarrow c + cost(w,t)$
22: **end if**
23: **end for**
24: **return** $\hat{M}^*$

Example 2 Backing to our running example in Example 1. As shown in Figure 2, four flows are picked out by running the min-cost max-flow algorithm, which are
$f_1 : s->w_3->t_2->e$ with cost 1,
$f_2 : s->w_4->t_1->e$ with cost 2,
$f_3 : s->w_5->t_5->e$ with cost 5,
$f_4 : s->w_6->t_4->e$ with cost 1.
The total used budget is 9. The worker and task vertices are eventually extracted out from each flow, and then form the final pair set $\hat{M}^* = \{(w_3,t_2), (w_4,t_1), (w_5,t_5), (w_6,t_4)\}$.

Complexity Analysis The time complexity of the OPT algorithm is $O(V^2 * |\hat{M}|)$, where $V = (|W|+|T|+2)$. The $Min\_Cost\_Max\_Flow$ algorithm invokes the Dijkstra algorithm which expends $O(V^2)$ time cost to seek a new flow from s to e in each invocation. $|\hat{M}|$ is the total flows found by the $Min\_Cost\_Max\_Flow$ algorithm.



(a) Build graph (b) Min-cost max-flow (c) Matching pairs

Figure 2 The key steps to solve the offline optimal solution in Example 2

4 Online assignment algorithms

4.1 Greedy-RT algorithm

Greedy is a simple and efficient method for most online task matching problems, but its performance is susceptible to the order of workers' appearance. Its competitive ratio achieves the worst $\frac{1}{2^{min\{|T|,|W|\}-1}}$ [26] when workers' appearance follows the adversarial model. In order to alleviate the impact of the order, random-threshold greedy(Greedy-RT) [22] is more competent than the simple greedy method for its threshold, which is randomly generated, could feasibly filter out those extremely bad matching pairs. Specifically, Greedy-RT first produces a random threshold τ, and then the pair whose travel cost exceeds τ is denied, so that the abused budget which caused by early workers in the adversarial model is restrained. In this section, we utilize the random-threshold greedy algorithm to solve the BOA problem.

Algorithm 2 Greedy-RT algorithm.

Require: B, c_{max}
Ensure: matching scheme M
1: $M \leftarrow \emptyset, c \leftarrow 0$
2: choose κ randomly from the set $\{0,1,...,\lceil ln(c_{max}+1)\rceil\}$ with the probability $Pr(\kappa = i) = \frac{1}{\lceil\lceil ln(c_{max}+1)\rceil\rceil}$
3: $\tau \leftarrow e^{\kappa}$
4: **for all** new arrival worker w **do**
5: $T' \leftarrow \{\forall t | b_w + cost(w,t)/v_w \le d_t \wedge cost(w,t) \le \tau \wedge c + cost(w,t) \le B\}$
6: **if** $T' \neq \emptyset$ **then**
7: $t = \min_{t \in T'} cost(w,t)$
8: $M \leftarrow M \cup (w,t)$
9: $c \leftarrow c + cost(w,t)$
10: **end if**
11: **end for**
12: **return** M

The whole procedure of Greedy-RT is illustrated in Algorithm 2. In line 1, the result set M and the used budget c are assigned with initial values. In lines 2-3, Greedy-RT randomly chooses a threshold e^{κ} on travel cost according to the estimated maximum cost c_{max} which

can be learned from the scope that workers and tasks appear. In lines 4-5, when a worker w arrives, Greedy-RT filters a task subset T' where each task satisfies three conditions: 1) w could arrive before t's deadline; 2) the travel cost between w and t is not greater than the specific threshold e^{κ}; 3) the used budget c after adding the cost $d(w, t)$ cannot exceed the total budget B. In lines 6-9, if T' is not empty, Greedy-RT chooses the nearest task from T', adds it into M and updates the used budget c. In line 12, the algorithm returns the final matching scheme M when all the workers have already appeared or the budget has been used up.

Example 3 Backing to our running example in Example 1, workers and tasks appear in the 8×8 square region where the maximum travel cost does not exceed the manhattan distance 16. According to the upper bound $\lceil ln(c_{max} + 1) \rceil$, Greedy-RT divides the whole travel cost into four grades whose associated thresholds are e^0, e^1, e^2 and e^3 respectively. The matching schemes based on various thresholds are listed as follows:

e^0:{(w_3, t_2), (w_6, t_4)}, matching size:2, used budget:2
e^1:{(w_3, t_2), (w_4, t_1), (w_6, t_4)}, matching size:3, used budget:4
e^2:{(w_1, t_1), (w_2, t_2)}, matching size:2, used budget:10
e^3:{(w_1, t_1), (w_2, t_2)}, matching size:2, used budget:10.

Greedy-RT randomly chooses one threshold, so the expectation of the matching size is $E(Greedy - RT) = \frac{2}{4} + \frac{3}{4} + \frac{2}{4} + \frac{2}{4} = 2.5$, which outperforms than the simple greedy algorithm.

We next analyze the competitive ratio of Greedy-RT. Let $\mathcal{O}$ be the optimal matching with the limited budget B. Define $\mathcal{O}(e^i, e^{i+1}] = \{(w, t) \in \mathcal{O} | cost(w, t) \in (e^i, e^{i+1}]\}$ to be the subset of $\mathcal{O}$ with the cost in the interval $(e^i, e^{i+1}]$. For any $i \geq 0$, let $M_{\leq e^i}$ denote the matching scheme returned by Greedy-RT whose threshold τ equals to e^i, or equivalently, $\kappa = i$.

Lemma 1 *For any* $i \geq 0$, $|M_{\leq e^i}| \geq \begin{cases} |\mathcal{O}(e^{i-1}, e^i]| & i \geq 1 \\ |\mathcal{O}(0, e^0]| & i = 0 \end{cases}$.

Proof When $i = 0$, matching pattern $M_{\leq e^0}$ is achieved with the budget B while $\mathcal{O}(0, e^0]$ is done with budget $B_{(0,e^0]}$. Since $B_{(0,e^0]} \leq B$, then $|M_{\leq e^0}| \geq |\mathcal{O}(0, e^0]|$. When $i \geq 1$, we have $|M_{\leq e^i}| = |M^{B_0}_{(0,e^0]}| + \sum_{j=1}^{i} |M^{B_j}_{(e^{j-1},e^j]}|$, where $\sum_{j=0}^{i} B_j = B$ and B_j is determined by the Greedy-RT algorithm. Since $|M^{B_0}_{(0,e^0]}| \geq |M^{B_0}_{(e^{i-1},e^i]}|$ and $|M^{B_j}_{(e^{j-1},e^j]}| \geq |M^{B_j}_{(e^{i-1},e^i]}|$ for any $j \in [1, i]$, we have

$$|M_{\leq e^i}| \geq |M^{B_0}_{(e^{i-1},e^i]}| + \sum_{j=1}^{i} |M^{B_j}_{(e^{i-1},e^i]}| = |M^{B}_{(e^{i-1},e^i]}|. \tag{1}$$

Similarly, $\mathcal{O}(e^{i-1}, e^i]$ is achieved with the budget $B_{(e^{i-1},e^i]}$ and $B_{(e^{i-1},e^i]} \leq B$, we have

$$|M_{\leq e^i}| \geq |\mathcal{O}(e^{i-1}, e^i]|. \tag{2}$$

The lemma is proved. □

Theorem 2 *The competitive ratio of the Greedy-RT Algorithm is not less than* $\frac{1}{\lceil ln(c_{max}+1)\rceil+1}$.

Proof Let $n = \lceil ln(c_{max} + 1)\rceil$. Since the exponential part of threshold is chosen evenly from a set of integers between 0 and n, we have

$$E(|M|) = \sum_{i=0}^{n} |M_{\leq e^i}| p_i = \frac{1}{n+1} \sum_{i=0}^{n} |M_{\leq e^i}|. \tag{3}$$

According to Lemma 1,

$$E(|M|) \geq \frac{1}{n+1}(|\mathcal{O}(0, e^0]| + |\sum_{i=1}^{n} |\mathcal{O}(e^{i-1}, e^i]|) = \frac{1}{n+1}|\mathcal{O}|. \tag{4}$$

That is,

$$\frac{E(M)}{|\mathcal{O}|} \geq \frac{1}{\lceil ln(c_{max} + 1)\rceil + 1}. \tag{5}$$

The theorem follows. □

Complexity Analysis For each new arrival worker, the time complexity of the Greedy-RT algorithm is $O(|T|)$.

4.2 Greedy-OT algorithm

Greedy-RT is unstable due to its random selected threshold. Specifically, certain rational pairs will be neglected if the chosen threshold is too small, which causes its performance turns even worse than the simple greedy algorithm. It is crucial for threshold-based algorithms to choose an appropriate threshold. In this section, we propose a superior greedy variant, called Greedy-OT, which can generate a near optimal threshold.

There are periodic similarities on human daily movements and travel traces. For instance, we arbitrarily extract six-days samples from Uber dataset in May 2014, each of which comprises the detailed pickup records happened in Manhattan, New York city. Figure 3 depicts the distribution of the pickups happened between 0 o'clock and 12 o'clock with heat mapps.

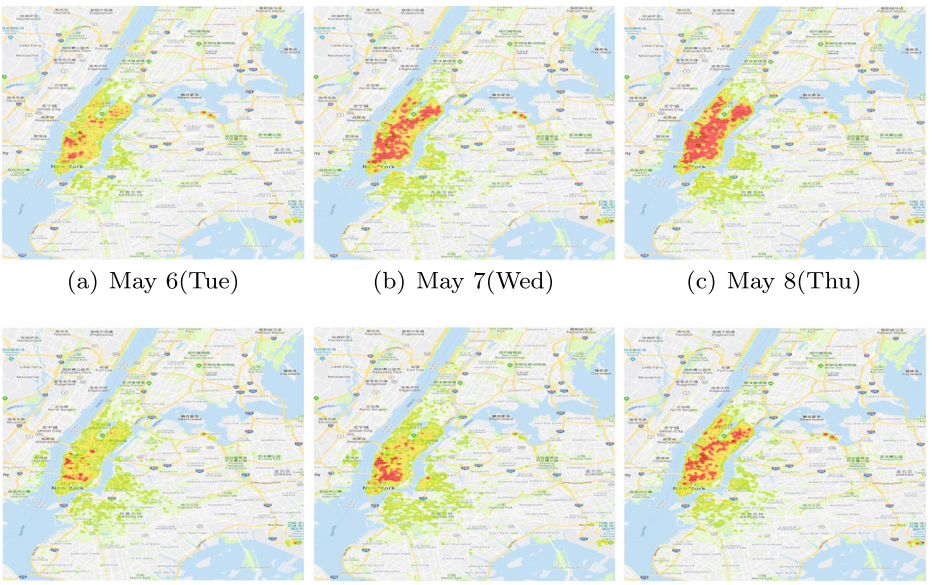
We can observe that the spatiotemporal distributions of the pickups are similar each day. Although there is a large deviation in the quantity of the pickups, the distributions of travel cost in optimal matching schemes for these days remain similar if pickups represent workers in matching with same batch of tasks. The main reason is that distribution plays crucial roles rather than quantity in budget-constraint matching problem, and superfluous workers would not produce significant impact on the final optimal matching scheme. Thus, it is feasible to utilize historical optimal matching scheme to guide the online assignment for the other days. In this section, we propose another greedy variant, called Greed-OT, which utilizes historical information to obtain proper threshold. Specifically, Greedy-OT first runs the offline optimal algorithm based on workers' historical traces, and then from the offline scheme extracts the maximum travel cost as target threshold. The detail description on Greedy-OT is listed in Algorithm 3.

Algorithm 3 Greedy-OT algorithm.

Require: B, $\hat{M}^*$
Ensure: matching scheme M
1: $c \leftarrow 0, \tau \leftarrow \max_{(w,t)\in\hat{M}^*} cost(w,t)$
2: **for all** new arrival worker w **do**
3: $T' \leftarrow \{\forall t | b_w + cost(w,t)/v_w \leq d_t \wedge cost(w,t) \leq \tau \wedge c + cost(w,t) \leq B\}$
4: **if** $T' \neq \emptyset$ **then**
5: $t = \min_{t\in T'} cost(w,t)$
6: $M \leftarrow M \cup (w,t)$
7: $c \leftarrow c + cost(w,t)$
8: **end if**
9: **end for**
10: **return** M

Example 4 Backing to our running example in Example 1, we assume historical data has the same spatiotemporal distribution with this case. Greedy-OT first utilizes maximum travel cost of the optimal matching pairs as the threshold, here, the maximum pair is (w_5, t_5) with cost 5, and then prunes bad pairs whose cost exceeds 5. In this way, we obtain the final matching solution: 5:$\{(w_2, t_1), (w_3, t_2), (w_4, t_6), (w_6, t_4)\}$, matching size:4, used budget:10. In this example, Greedy-OT and OPT achieve the same score in matching size, but OPT is more frugal in budget expenditure for only 90 percent budget is spent, while all the budget is spent by Greedy-OT.

We next deduce the competitive ratio of Greedy-OT. Suppose n_i is the number of pairs with travel cost c_i in the optimal set $\mathcal{O}$ which owns N unique cost values. That is, the size



(a) May 6(Tue) (b) May 7(Wed) (c) May 8(Thu)

(d) May 10(Sat) (e) May 11(Sun) (f) May 14(Wed)

Figure 3 Similarity on the distribution of Uber pickups for different days in May, 2014

of pairs $|\mathcal{O}| = \sum_{i=1}^{N} n_i$. The travel cost of any pair from w to t in $\mathcal{O}$ can be denoted by $c_{(w,t)_{opt}}$, where $(w,t)_{opt} \in [1,N]$. $W_{\mathcal{O}}$ is the worker set and $T_{\mathcal{O}}$ is the task set in $\mathcal{O}$. The maximum travel cost of pairs in $\mathcal{O}$ is denoted by c^*_{max}, and $c^*_{max} + \varepsilon$ is the threshold that Greedy-OT adopts to prune inferior matching pairs.

Lemma 2 *If there exists a pair(w,t) chosen by Greedy-OT with its cost little than c^*_{max}, then it must be $w \in W_{\mathcal{O}}$ or $t \in T_{\mathcal{O}}$.*

Proof Suppose there exists a pair $(w,t) \notin \mathcal{O}$ with its cost little than c^*_{max}, and its two ends, w and t, follows $w \notin W_{\mathcal{O}}$ and $t \notin T_{\mathcal{O}}$, then by replacing the maximum cost pair with (w,t), we would obtain a new optimal solution, which is contradict with the fact that $\mathcal{O}$ is the genuine optimal solution. The lemma is proved. □

Lemma 3 *If a pair p chosen by Greedy-OT is overlapped with a pair p^* in $\mathcal{O}$ at least one point (w or t), then $cost(p^*) \le cost(p) \le c^*_{max} + \varepsilon$.*

Proof Since Greedy-OT selects $c^*_{max} + \varepsilon$ as threshold, the cost of any chosen pair can not beyond $c^*_{max} + \varepsilon$. Assume the cost of p is little than p^*, then p should be added into $\mathcal{O}$ instead of p^*, which is contradict with the fact that p^* is the genuine pair in $\mathcal{O}$. Thus, the assumption that $cost(p) < cost(p^*)$ is false and alternatively $cost(p) \ge cost(p^*)$. The lemma is proved. □

Theorem 3 *The competitive ratio of Greedy-OT is not less than* $\frac{\sum_{i=1}^{N} c_i n_i}{(c^*_{max}+\varepsilon)\cdot\sum_{i=1}^{N} n_i}$.

Proof For the case $\varepsilon \ge 0$, according to Lemma 2 and 3, the cost of pair (w,t) chosen by Greedy-OT affiliates with either of the following cases:

1)$cost(w,t) \in [c_i, c^*_{max} + \varepsilon]$, where $w \in W_{\mathcal{O}}$, $cost(w,)_{opt} = c_i$ or $t \in T_{\mathcal{O}}$, $cost(,t)_{opt} = c_i$;
2)$cost(w,t) \in [c^*_{max}, c^*_{max} + \varepsilon]$, where $w \notin W_{\mathcal{O}}$ and $t \notin T_{\mathcal{O}}$.

For both of cases, any pair (w,t) that Greedy-OT chooses can be mapped to an optimal pair in $\mathcal{O}$, which inflates the cost from optimal value c to $cost(w,t) \in [c, c^*_{max} + \varepsilon]$ or $[c^*_{max}, c^*_{max} + \varepsilon]$ and consequently decreases the matched number of Greedy-OT. Specifically, for the case 1, pair (w,t) is bound to be mapped to a optimal pair whose worker entry is w or task entry is t, and for the case 2, pair (w,t) is mapped to a random optimal pair which will never be selected by the end of Greedy-OT. Suppose ξ_{ij} is the travel cost of chosen pair which is mapped to the j-th pair among n_i optimal pairs with the same cost c_i, then the lower bound of ξ_{ij} is either c_i (case 1) or c^*_{max} (case 2) while the upper bound are both $c^*_{max} + \varepsilon$. The worst expectation of Greedy-OT is

$$\begin{aligned} E(Greedy-OT) &= E(\frac{B}{cost}) = B \cdot E(\frac{1}{cost}) \\ &= \frac{B}{\sum_{i=1}^{N} n_i}(\sum_{i=1}^{N}\sum_{j=1}^{n_i}\frac{1}{\xi_{ij}}) \\ &\ge \frac{B}{\sum_{i=1}^{N} n_i}(\sum_{i=1}^{N} n_i \cdot \frac{1}{c^*_{max}+\varepsilon}) \\ &= \frac{B}{c^*_{max}+\varepsilon} \end{aligned} \tag{6}$$

Since $B \geq \sum_{i=1}^{N} c_i n_i$, we have

$$\begin{aligned} CR(Greedy-OT) &= \frac{E(Greedy-OT)}{|\mathcal{O}|} \\ &\geq \frac{\sum_{i=1}^{N} c_i n_i}{(c^*_{max}+\varepsilon)\cdot\sum_{i=1}^{N} n_i}. \end{aligned} \tag{7}$$

For the case $\varepsilon < 0$, any chosen pair (w, t) by Greedy-OT is bound to be overlapped with $\mathcal{O}$ according to Lemma 2. The travel cost of (w, t) only affiliates one case, that is

1)$cost(w,t) \in [c_i, c^*_{max}+\varepsilon]$, where $w \in W_{\mathcal{O}}$, $cost(w,)_{opt} = c_i$ or $t \in T_{\mathcal{O}}$, $cost(,t)_{opt} = c_i$.

Suppose $c_K \leq c^*_{max}+\varepsilon < c_{K+1} \leq c_N$, then we have

$$\begin{aligned} E(Greedy-OT) &= \frac{B}{\sum_{i=1}^{K} n_i}\Big(\sum_{i=1}^{K}\sum_{j=1}^{n_i}\frac{1}{\xi_{ij}}\Big) \\ &\approx \frac{B}{\sum_{i=1}^{K} n_i}\Big(\sum_{i=1}^{K} n_i \cdot E\big(\frac{1}{\xi_{ij}}\big)\Big), \end{aligned} \tag{8}$$

where $\xi_{ij} \in [c_i, c^*_{max}+\varepsilon]$. Here

$$\begin{aligned} E\big(\frac{1}{\xi_{ij}}\big) &= \int_{c_i}^{c^*_{max}+\varepsilon} \frac{1}{\xi_{ij}} \cdot \frac{1}{c^*_{max}+\varepsilon-c_i} d\xi_{ij} \\ &= \frac{\ln(c^*_{max}+\varepsilon) - \ln c_i}{c^*_{max}+\varepsilon-c_i}. \end{aligned} \tag{9}$$

Since $B \geq \sum_{i=1}^{N} c_i n_i$, we have

$$\begin{aligned} CR(Greedy-OT) &= \frac{E(Greedy-OT)}{|\mathcal{O}|} \\ &\geq \frac{(\sum_{i=1}^{N} c_i n_i)\cdot(\sum_{i=1}^{K} n_i \frac{\ln(c^*_{max}+\varepsilon)-\ln c_i}{c^*_{max}+\varepsilon-c_i})}{\sum_{i=1}^{N} n_i \cdot \sum_{i=1}^{K} n_i}. \end{aligned} \tag{10}$$

On the ground that $f(x) = \frac{\ln(c^*_{max}+\varepsilon)-\ln x}{c^*_{max}+\varepsilon-x}$ is a decreasing function on x, and $\lim_{x\to c^*_{max}+\varepsilon} f(x) = \frac{1}{c^*_{max}+\varepsilon}$, we have

$$CR(Greedy-OT) \geq \frac{\sum_{i=1}^{N} c_i n_i}{(c^*_{max}+\varepsilon)\cdot\sum_{i=1}^{N} n_i}. \tag{11}$$

The theorem follows. □

Complexity Analysis Similar to the Greedy-RT algorithm, the Greedy-OT algorithm also has a time complexity of $O(|T|)$ for each new arrival worker.

5 Experiments

5.1 Experiment setup

Synthetic Dataset We generate 10000 workers and 10000 tasks on a 500×500 2D square where all positions are generated randomly. The value of c_{max} is set to 1000, which is the maximum Manhattan distance in the specific area. The appearance of workers follows the adversarial model and random model separately, and in either of the models, the arrival time of workers and the release time of tasks scatter randomly between 0 and 99. To simulate the random order model, we sort the arrival time of all workers in ascending order to obtain their arrival sequence. For the design of the adversarial order model, we first calculate the travel cost for each worker to reach his nearest neighbor task, then sort all travel cost in ascending order to obtain the adversarial sequence, and finally exchange workers' arrival times to ensure that the arrival sequence conforms to the adversarial model. The measurement on a worker's travel cost equals to the Manhattan distance between his present and target location. For simplicity, we set all workers' velocity to 1, so that time cost is equivalent to distance cost. Platform expends budget to reward any assigned worker according to his travel cost. The statistics and configuration of synthetic data are illustrated in Table 1, where the default settings are marked in bold.

Real Dataset We choose Uber Trip Data [29] as our real dataset, which contains data on over 4.5 million Uber pickups in New York City from April to September 2014, and 14.3 million more Uber pickups from January to June 2015. Based on the raw data of the second week in May 2014, we select the rectangle area from the position(Lat:40.5998, Lon:-74.0701) to the position(Lat:40.8998, Lon:-73.7701) as investigative area whose maximum distance c_{max} is 41.7027km, which equals to the Euclidean distance of its diagonal. Any pickup appeared from 0 o'clock to 12 o'clock in the area plays as a worker and the velocity of the workers are set to the same value 40km/h. Due to the lack of task information in UTD dataset, we randomly generate 6000 tasks whose location is limited in the area, release time randomly scatters from 0:00 to 12:00 and 180 minutes survival time before deadline. We extract the optimal threshold from the pickups data in May 7(Wed), 2014, and then utilize it to guide the online matchings for the other days of this week. In our case, the requester supplies budget in amount of 300 to reward workers, which means the total travel cost (distance cost) of chosen workers can not beyond 300km.

We compare Greedy-RT, Greedy-OT with the baseline greedy and the offline optimal OPT in terms of total matching size, running time and memory cost, and then study the effect of varying parameters. In each experiment, we repeatedly test 20 different online arrival orders of workers and report the average results. All the algorithms are implemented in C++, and run in a machine with Intel(R) Core(TM) i5-2400 CPU and 4GB main memory.

Table 1 Synthetic dataset

Factor	Setting	Notes
$\|W\|$	2000,4000,6000,8000,10000	No. of workers
$\|T\|$	2000,4000,6000,8000,10000	No. of tasks
B	1000,2000,3000,4000,5000	total budget
d_t	20,40,60,80,100	Deadline of task

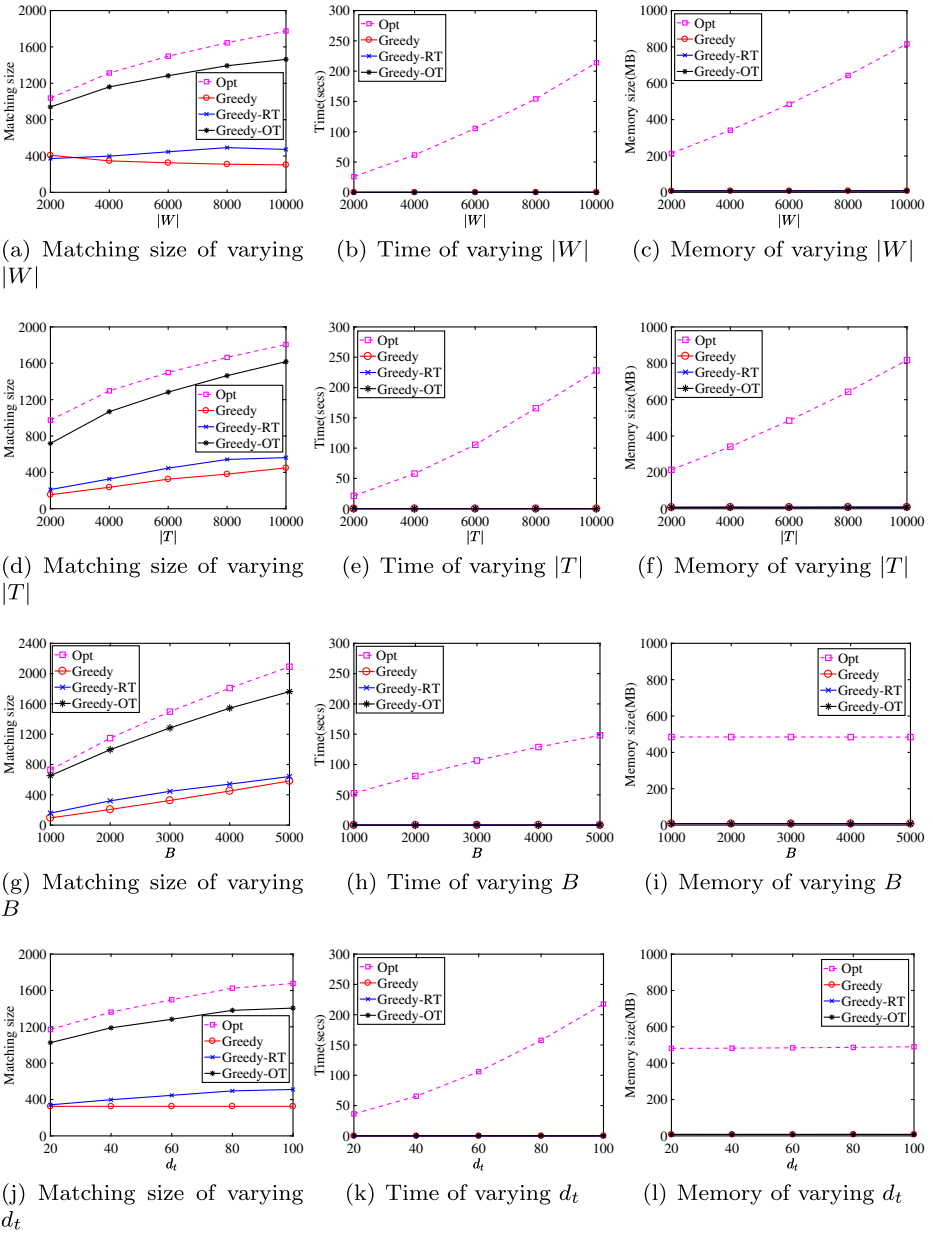
5.2 Adversarial model

Effect of $|W|$ The first row of Figure 4 shows the results when $|W|$ varies from 2000 to 10000. We can observe that for all the algorithms except the simple greedy, the quantity of successful matched pairs increases as $|W|$ increases, which is natural as there are more competent pairs that can be matched when more workers are available. Only the curve of the simple greedy algorithm declines when $|W|$ increases since the pairs with larger cost keep growing in quantity and have prior in matching in the adversarial model. Among all online algorithms, Greedy-OT performs the best, followed by Greedy-RT and simple greedy. The reasons are as follows: 1) Greedy-OT can achieve a proper threshold from a set of offline optimal pairs; 2) part of thresholds have poor performances in matching and thus the expected performance of Greedy-RT is correspondingly affected; 3) simple greedy is seriously trapped in local optimal solutions in the adversarial model. Note that Greedy-RT maybe perform worse than simple greedy when less workers are available, on the ground that large number of proper pairs are filtered out by excessively small thresholds. As for the running time and travel cost, OPT expends much than online algorithms for the reason that the min-cost max-flow algorithm needs much time and storage consumption on repetitively running the Dijkstra algorithm and keeping running states. Other algorithms are subordinate to the greedy algorithm family so that they are similar in terms of time and storage cost. Moreover, they are usually unrelated to the number of workers for the reason that the algorithms will terminate in advance when budget is drained rather than when all workers have appeared.

Effect of $|T|$ The second row of Figure 4 presents the results when we vary $|T|$ from 2000 to 10000. In terms of the quantity of matched pairs, we observe that the curves of all algorithms ascend when $|T|$ increases since more competent pairs can be matched. Greedy-OT still performs much better than Greedy-RT and simple greedy. Due to the number of workers, here is 6000, is abundant enough to prevent the decline in quantity that excessively small thresholds bring about. Thus, Greedy-RT could keep higher in quantity than simple greedy when $|T|$ increases. The results on time and memory cost have similar patterns to those of varying $|W|$, and we omit the detailed discussion.

Effect of budget The third row of Figure 4 shows the results when we vary B from 1000 to 5000. We can observe that the matching quantity rises when budget increases. The maximum value of optimal pairs grows when budget increases, which leads the threshold of Greedy-OT increases as well. Since the feature that workers with large cost appear early in the adversarial model, the ratio between Greedy-OT and OPT in matching size decreases when budget increases. The running time of OPT rises with the increase of budget. The reason is that additional budget would cause optimal pairs grow in quantity. For each pair, OPT will run the Dijkstra algorithm in a huge graph which needs expensive time cost. As for online algorithms, their running time is still tiny even can be ignored when compared with OPT. The memory cost of OPT has no related to the budget since they run on the identical graph structure with same vertices and edges.

Effect of deadline The fourth row of Figure 4 depicts the results of varying tasks' deadline from 20 to 100. All algorithms except simple greedy increase in matching size with the increase of deadline. This is because that long deadline represents more candidate tasks for



(a) Matching size of varying $|W|$ (b) Time of varying $|W|$ (c) Memory of varying $|W|$

(d) Matching size of varying $|T|$ (e) Time of varying $|T|$ (f) Memory of varying $|T|$

(g) Matching size of varying B (h) Time of varying B (i) Memory of varying B

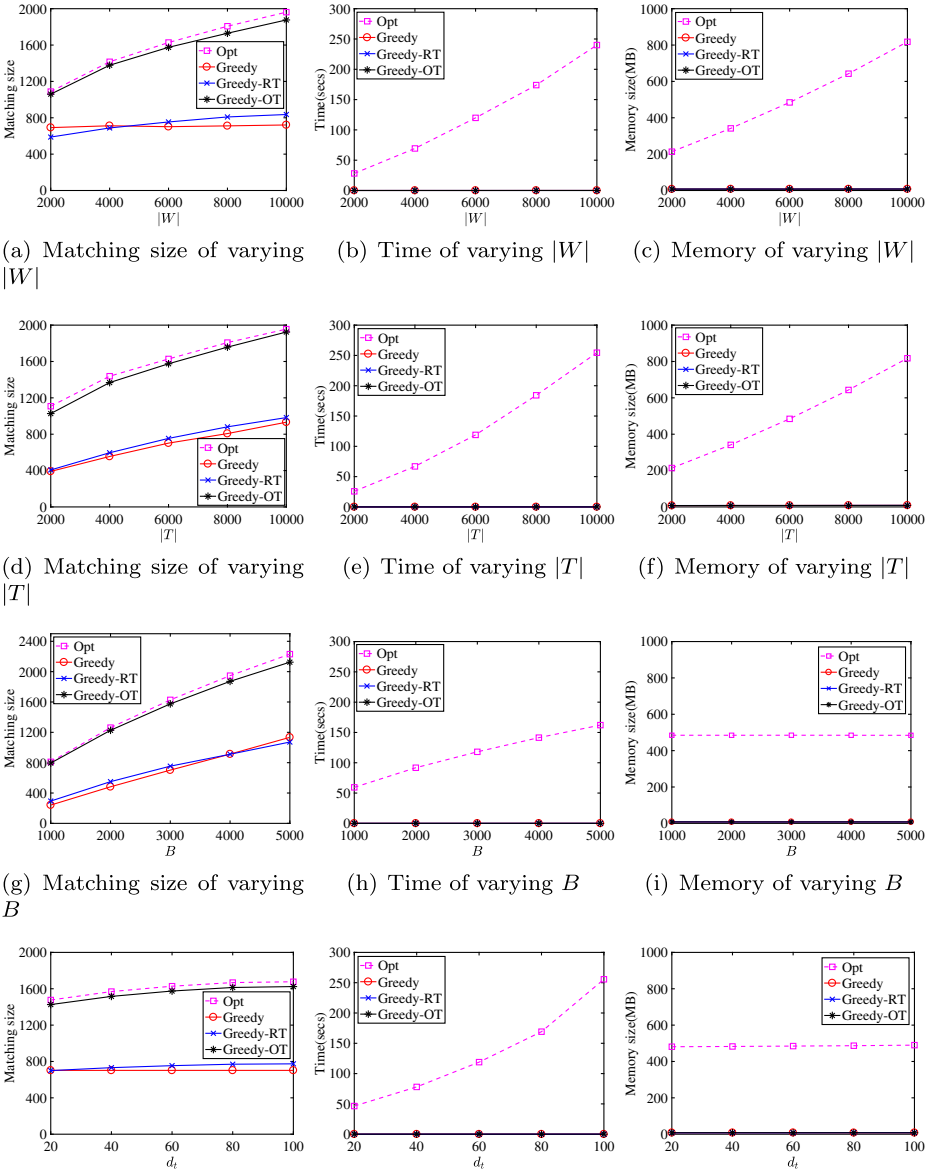
(j) Matching size of varying d_t (k) Time of varying d_t (l) Memory of varying d_t

Figure 4 Results on varying each specific parameter in the adversarial model

a new arrival worker to match with, which leads to less travel cost and correspondingly a larger matching size. The simple greedy algorithm keeps invariable in matching size for the reason that the travel cost of all matched pairs is little than 20, which means any worker can reach his nearest task before deadline. The results on time and memory cost have similar patterns to those of varying budget, and we omit the detailed discussion.

5.3 Random model

In this section, we will analyze the performance of all algorithms run in the random model. As Figure 5 depicts, all algorithms perform highly similar to the adversarial model in terms of matching size, so we merely analysis the discrepancies between Figures 4 and 5. First,



(a) Matching size of varying $|W|$ (b) Time of varying $|W|$ (c) Memory of varying $|W|$

(d) Matching size of varying $|T|$ (e) Time of varying $|T|$ (f) Memory of varying $|T|$

(g) Matching size of varying B (h) Time of varying B (i) Memory of varying B

(j) Matching size of varying d_t (k) Time of varying d_t (l) Memory of varying d_t

Figure 5 Variation of matching size on each specific parameter in the random model

we can observe that the curve of Greedy-OT in the random model is more asymptotic to the optimal curve than in the adversarial model, which means Greedy-OT performs better in the random model. Since the adversarial model has the feature that high cost workers have prior in appearance, workers Greedy-OT chooses have larger travel cost which consumes budget rapidly and consequently decreases matching size. Second, the gap of matching size between Greedy-RT and simple greedy is more narrow in the random model. The main reason is that simple greedy improves more significant than Greedy-RT in the random model in terms of matching size. Third, compared with Greedy-RT, simple greedy may perform better in certain conditions such as less available workers or sufficient budget. Specifically, as Figure 5a shown, simple greedy works well when $|W|$ equals 2000 and 4000. The reasons are as follows: 1) small thresholds Greedy-RT selects decrease the matching size; 2) simple greedy performs better in the random model than adversarial model. We also find that simple greedy outperforms than Greedy-RT when budget exceeds 4000 from Figure 5g. The reasons are as follows: 1) small thresholds still filter out many proper pairs even if the budget is sufficient; 2) those well-performance thresholds promote matching size slowly, or even reach the limit of matching size. The results on time and memory cost in the random model are similar to the adversarial model, and we omit the detailed discussion.

5.4 Real dataset

We next present the experiment results on the real dataset.

As Figure 6 shows, Greedy-OT is still the best in matching size among all online algorithms, followed by Greedy-RT and simple greedy. Greedy-OT achieves approximately 70 percent of matching sizes compared with OPT, which is at least 1.5 times than other greedy algorithms. As for the time and memory cost, OPT is still the most heavyweight algorithm. Especially in our case, the velocity of a new arrival worker is high enough that he can reach most of the tasks before their deadline. Consequently, a large number of edges is added when building flow graph and time cost increases dramatically. Compared with OPT, the other algorithms have tiny time and memory cost. Table 2 illustrates the quantity of pickups(workers) has no significant impact on the optimal threshold. The optimal thresholds during the whole week fluctuate less than 15 percent even if the number of pickups varies in a large magnitude from 9349 to 4851, which demonstrates Greedy-OT we propose is robust.

Discussion It's worth mentioning that there are no public or shared real datasets available for our BOA problem whatever in the fields of academic research or real applications, so we resort to the Uber dataset provided by a famous but another type of spatial crowdsourcing application without budget constraints. In order to make the dataset compatible with our problem, we artificially processed it by simulating workers with pickup records, manually generating task sets, and specifying workers' moving speeds.

Although Greedy-RT can improve the matching size compared with the simple greedy algorithm, its performance is still unstable due to the random generation of thresholds, especially when the generated thresholds are inaccurate, it performs even worse than the simple greedy algorithm.

Greedy-OT can greatly improve its matching performance by extracting a near optimal threshold from history records and using the appropriate threshold to filter out those bad matching pairs. Compared with the method of dividing whole region by grids and guiding online task assignment based on the predicted probability distribution of available workers in each grid and each time period, Greedy-OT has the advantages of simplicity and efficiency. In addition, the algorithm is less affected by the fluctuation of workers' historical

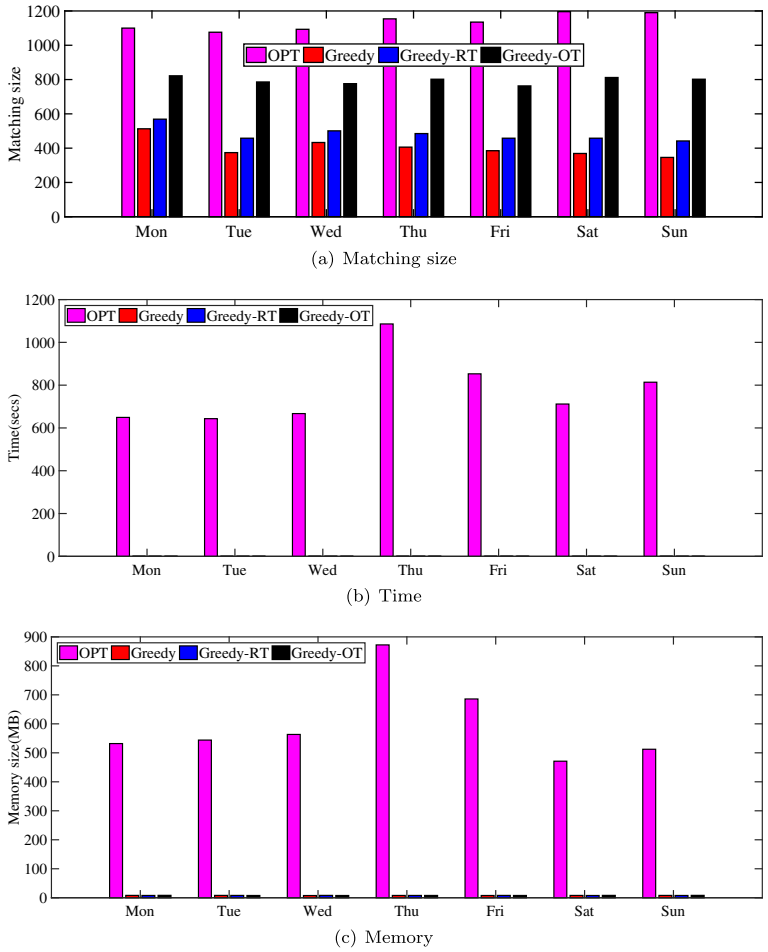


Figure 6 Performance evaluation on the Uber dataset during the second week of May 2014

distribution, so it has strong robustness. Meanwhile, for spatial crowdsourced applications with budget constraints, the improvement in matching size means that the expenditure of budget is more economical, and platform could provide requesters with better service quality.

The daily activities of workers have certain spatial and temporal patterns. It can be found from the real dataset that the travel cost distributions of workers in the optimal matching schemes generated from daily records are similar. Figure 7 shows the distribution of matching size under different travel cost from the raw optimal matching data in May 6 (the previous day), 7(current day), 8(the next day) and 14(Wed in the next week). All curves share the similar trend with the increase of travel cost, which dramatically rises up to their peaks and then gradually drops in the nadir. Further, We separate the whole travel cost into several segments, each of which spans 0.01km, and then we can find that matching size during each cost segment is similar. Therefore, we can obtain an appropriate threshold based

Table 2 Threshold Extraction on the Uber dataset during the second week of May 2014

Date	May 5 (Mon)	May 6 (Tue)	May 7 (Wed)	May 8 (Thu)	May 9 (Fri)	May 10 (Sat)	May 11(Sun)
No. Pickups (workers)	5682	5818	6065	9347	7377	4851	5276
Opt Threshold (Km)	0.7956	0.8290	0.8356	0.7848	0.7762	0.7362	0.7277

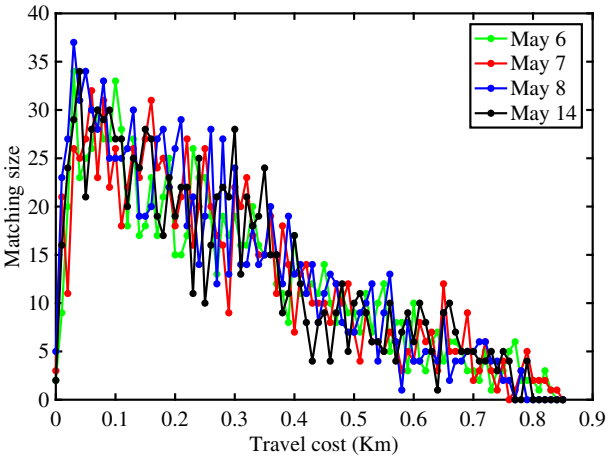


Figure 7 Similar distributions on travel cost for different days

on the four thresholds with multiple choices, such as selecting any one, mean, maximum or minimum among the thresholds to guide the online assignment for the other days.

The threshold-extraction method we propose can not only solve the BOA problem, but also be applied to some crowdsourcing applications which have no budget constraints. Taking Uber for example, we can obtain the optimal matching schemes from daily vehicle and passenger data, extract the waiting time threshold of workers and the dispatching distance threshold of vehicles for various regions and time periods according to the optimal schemes, and use them as important references to make real-time car scheduling decisions.

6 Related work

In this section, we review related work from two aspects which are task assignment with budget constraints and online task assignment.

Task assignment with budget constraints Many studies lay stress on the budget constrained assignment problem in Web crowdsourcing. Most of them take worker's reputation or reliability into consideration, and aim to achieve truthful and guaranteed answers. Karger et al. [6] takes account the confidence of answers that crowd workers submit, and aims to minimize the budget to meet certain reliability target. Li et al. [11] aims to maximize the number of labeled instances that achieve specified quality requirement under a tight budget. Khetan, Lahouti and Liu et al. [9, 10, 12] aim to find an adaptive task scheme to achieve the best trade-off between budget and accuracy of aggregated answers. Other research focuses on obtaining optimal task assignment which maximizes tasks' completion rate under budget constraint or provides maximum profit for requesters. Biswas et al. [2] improves the number of successful assignment by exploring and exploiting high-qualify workers with budgeted multi-armed bandit mechanism. Wu et al. [30] proposes linear programming based algorithms to maximize the expected profit of assignment. As for the field of spatial crowdsourcing, certain studies on the budget constrained assignment problems have also been sparked. To and Zhao [25, 32] focus on the online maximum task coverage problem which

aims to select a set of workers to maximize task coverage under budget constraints. Miao and Yu [14, 31] take workers' reputation and geometrical proximity into account, and aim to maximize the expected quality of the assignment with a limited budget. Cheng [3] formulates the maximum quality task assignment problem with the optimization objective to maximize a global assignment quality score under a traveling budget constraint. The aforementioned literatures in spatial crowdsourcing stress on the worker selection and quality assurance, which differ from our optimization goal on matching size.

Online task assignment The task assignment problem is one of core issues in spatial crowdsourcing. In offline scenarios, a large number of studies have obtained substantial achievements on fundermental assignment problem [7, 24] and complicated assignment problems which consider additional constraints such as reliability [8], spatiotemporal diversity [15], multi-skill coverage [16], conflict resolution [19] and privacy protection [17, 23]. Presently, the research on the task assignment problem has been transferred to online scenarios which are more practical for real applications. According to whether both of workers and tasks appear on platform dynamically, existing research can be divided into two categories: one-side and two-sides online assignment. Hassan, She and Tong [4, 20, 26] focus on one-side online assignment. Tong et al. [26] presents a comprehensive experimental comparison of representative algorithms [1, 5, 13] on online minimum bipartite matchings. Hassan and Curry [4] maximizes the number of successful assignments as spatial tasks arrive in an online manner. She et al. [20] extents bipartite matching to social network, and solves the event-participant arrangement problem with conflicting and capacity constraint when users arrive on platform dynamically. Song, Tong et al. [21, 27, 28] focus on two-sides online assignment. Tong et al. [27] devotes to allocate micro-tasks to suitable crowd workers in online scenarios, where all the spatiotemporal information of micro tasks and crowd workers are unkown. Tong et al. [28] guides workers' movements based on the prediction of distribution of workers and tasks to optimize the online task assignment. Besides the traditional bipartite online matching based on workers and tasks, Song et al. [21] presents the trichromatic online matching in real-time spatial crowdsourcing which comprise three entities of worker, tasks and workplace. Our solution for BOA attaches to one-side online task assignment problem since platform acquires tasks' spatiotemporal information in advance while be unaware of workers' until they appear. Different from aforementioned one-side assignment, we consider budget constraint and receive the guidance provided by historical data.

7 Conclusion

In this paper, we formally define a dynamic task assignment problem, called budget-aware online task assignment (BOA) in real-time spatial crowdsourcing. We first prove the optimal solution of BOA can be solved with min-cost max-flow algorithm, and then propose two greedy variants to solve the approximate solutions. The first variant named Greedy-RT, which has the competitive ratio of $\frac{1}{\lceil ln c_{max}+1 \rceil+1}$, generates a random threshold to abandon those large cost pairs to reduce the abuse of budget. In order to improve the stability of Greedy-RT, we further propose another variant called Greedy-OT, which learns a near optimal threshold from historical spatiotemporal information of workers and achieves the competitive ratio of $\frac{\sum_{i=1}^{N} c_i n_i}{(c^*_{max}+\varepsilon)\cdot\sum_{i=1}^{N} n_i}$. Finally, we verify the effectiveness and efficiency of the proposed methods through extensive experiments on both synthetic and real datasets.

References

1. Bansal, N., Buchbinder, N., Gupta, A., Naor, J.: A randomized o($log^2 k$)-competitive algorithm for metric bipartite matching. Algorithmica **68**(2), 390–403 (2014)
2. Biswas, A., Jain, S., Mandal, D., Narahari, Y.: A Truthful Budget Feasible Multi-Armed Bandit Mechanism for Crowdsourcing Time Critical Tasks. In: International Conference on Autonomous Agents and Multiagent Systems, pp. 1101–1109 (2015)
3. Cheng, P., Lian, X., Chen, L., Shahabi, C.: Prediction-Based Task Assignment in Spatial Crowdsourcing. In: IEEE International Conference on Data Engineering, pp. 997–1008 (2017)
4. Hassan, U.U., Curry, E.: A multi-armed bandit approach to online spatial task assignment. In: Proceedings of the 11th IEEE International Conference onUbiquitous Intelligence and Computing, pp. 212–219 (2014)
5. Kalyanasundaram, B., Pruhs, K.: On-Line Weighted Matching. In: Acm-Siam Symposium on Discrete Algorithms, pp. 234–240 (1991)
6. Karger, D.R., Oh, S., Shah, D.: Budget-optimal task allocation for reliable crowdsourcing systems. Oper. Res. **62**(1), 1–24 (2013)
7. Kazemi, L., Shahabi, C.: Geocrowd: Enabling Query Answering with Spatial Crowdsourcing. In: International Conference on Advances in Geographic Information Systems, pp. 189–198 (2012)
8. Kazemi, L., Shahabi, C., Lei, C.: Geotrucrowd: Trustworthy Query Answering with Spatial Crowdsourcing. In: Acm Sigspatial International Conference on Advances in Geographic Information Systems, pp. 314–323 (2013)
9. Khetan, A., Oh, S.: Achieving Budget-Optimality with Adaptive Schemes in Crowdsourcing. In: Advances in Neural Information Processing Systems 29, pp. 4844–4852 (2016)
10. Lahouti, F., Hassibi, B.: Fundamental Limits of Budget-Fidelity Trade-Off in Label Crowdsourcing. In: Thirtieth Annual Conference on Neural Information Processing Systems, pp. 5059–5067 (2016)
11. Li, Q., Ma, F., Gao, J., Su, L., Quinn, C.J.: Crowdsourcing High Quality Labels with a Tight Budget. In: ACM International Conference on Web Search and Data Mining, pp. 237–246 (2016)
12. Liu, X., He, H., Baras, J.S.: Trust-Aware Optimal Crowdsourcing with Budget Constraint. In: IEEE International Conference on Communications, pp. 1176–1181 (2015)
13. Meyerson, A., Nanavati, A., Poplawski, L.: Randomized Online Algorithms for Minimum Metric Bipartite Matching. In: Seventeenth Acm-Siam Symposium on Discrete Algorithm, pp. 954–959 (2006)
14. Miao, C., Yu, H., Shen, Z., Leung, C.: Balancing quality and budget considerations in mobile crowdsourcing. Decis. Support. Syst. **90**(2), 56–64 (2016)
15. Peng, C., Xiang, L., Zhao, C., Lei, C., Han, J., Zhao, J.: Reliable diversity-based spatial crowdsourcing by moving workers. Proc. Vldb Endowment **8**(10), 1022–1033 (2015)
16. Peng, C., Xiang, L., Lei, C., Han, J., Zhao, J.: Task assignment on multi-skill oriented spatial crowdsourcing. IEEE Trans. Knowl. Data Eng. **28**(8), 2201–2215 (2016)
17. Pournajaf, L., Li, X., Sunderam, V., Goryczka, S.: Spatial Task Assignment for Crowd Sensing with Cloaked Locations. In: IEEE International Conference on Mobile Data Management, pp. 73–82 (2014)
18. Restuccia, F., Das, S.K.: Fides: a Trust-Based Framework for Secure User Incentivization in Participatory Sensing. In: World of Wireless, Mobile and Multimedia Networks, pp. 1–10 (2014)
19. She, J., Tong, Y., Chen, L., Cao, C.C.: Conflict-Aware Event-Participant Arrangement. In: IEEE International Conference on Data Engineering, pp. 1629–1643 (2015)
20. She, J., Tong, Y., Chen, L., Cao, C.C.: Conflict-aware event-participant arrangement and its variant for online setting. IEEE Trans. Knowl. Data Eng. **28**(9), 2281–2295 (2016)
21. Song, T., Tong, Y., Wang, L., She, J., Yao, B., Chen, L., Xu, K.: Trichromatic Online Matching in Real-Time Spatial Crowdsourcing. In: IEEE International Conference on Data Engineering, pp. 1009–1020 (2017)
22. Ting, H.F., Xiang, X.: Near optimal algorithms for online maximum edge-weighted b -matching and two-sided vertex-weighted b -matching. Theor. Comput. Sci. **607**(P2), 247–256 (2015)
23. To, H., Ghinita, G., Shahabi, C.: A framework for protecting worker location privacy in spatial crowdsourcing. Proc. Vldb Endowment **7**(10), 919–930 (2014)
24. To, H., Shahabi, C., Kazemi, L.: A server-assigned spatial crowdsourcing framework. Acm Trans. Spatial Algorithm. Syst. **1**(1), 1–28 (2015)
25. To, H., Fan, L., Luan, T., Shahabi, C.: Real-Time Task Assignment in Hyperlocal Spatial Crowdsourcing under Budget Constraints. In: IEEE International Conference on Pervasive Computing and Communications, pp. 1–8 (2016)
26. Tong, Y., She, J., Ding, B., Chen, L., Wo, T., Xu, K.: Online minimum matching in real-time spatial data: experiments and analysis. Proc. Vldb Endowment **9**(12), 1053–1064 (2016)

27. Tong, Y., She, J., Ding, B., Wang, L., Chen, L.: Online Mobile Micro-Task Allocation in Spatial Crowdsourcing. In: IEEE International Conference on Data Engineering, pp. 49–60 (2016)
28. Tong, Y., Wang, L., Zhou, Z., Ding, B., Chen, L., Ye, J., Xu, K.: Flexible online task assignment in real-time spatial data. Proc. Vldb Endowment **10**(11), 1334–1345 (2017)
29. Uber Trip Data: https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city
30. Wu, K.L., Wu, K.L., Wu, K.L., Wu, K.L.: Budgeted Online Assignment in Crowdsourcing Markets: Theory and Practice. In: Conference on Autonomous Agents and Multiagent Systems, pp. 1763–1765 (2017)
31. Yu, H., Miao, C., Shen, Z., Leung, C.: Quality and budget aware task allocation for spatial crowdsourcing. In: Proceedings of International Conference on Autonomous Agents and Multiagent Systems, pp. 1689–1690 (2015)
32. Zhao, D., Li, X.Y., Ma, H.: Budget-feasible online incentive mechanisms for crowdsourcing tasks truthfully. IEEE/ACM Trans. Netw. **24**(2), 647–661 (2016)