2017 IEEE 33rd International Conference on Data Engineering

Trichromatic Online Matching in Real-time Spatial Crowdsourcing

Tianshu Song [†], Yongxin Tong [†], Libin Wang [†], Jieying She [‡], Bin Yao [#], Lei Chen [‡], Ke Xu [†]
[†]SKLSDE Lab, School of Computer Science and Engineering and IRI, Beihang University, China
[‡]The Hong Kong University of Science and Technology, Hong Kong SAR, China
[#]Shanghai Jiao Tong University, China
[†]{songts, yxtong, lbwang, kexu}@buaa.edu.cn, [‡]{jshe, leichen}@cse.ust.hk, [#]yaobin@cs.sjtu.edu.cn

***Abstract*—The prevalence of mobile Internet techniques and Online-To-Offline (O2O) business models has led the emergence of various spatial crowdsourcing (SC) platforms in our daily life. A core issue of SC is to assign real-time tasks to suitable crowd workers. Existing approaches usually focus on the matching of two types of objects, tasks and workers, or assume the static offline scenarios, where the spatio-temporal information of all the tasks and workers is known in advance. Recently, some new emerging O2O applications incur new challenges: SC platforms need to assign three types of objects, tasks, workers and workplaces, and support dynamic real-time online scenarios, where the existing solutions cannot handle. In this paper, based on the aforementioned challenges, we formally define a novel dynamic online task assignment problem, called the *<u>t</u>richromatic <u>o</u>nline <u>m</u>atching in real-time spatial crowdsourcing* (TOM) problem, which is proven to be NP-hard. Thus, we first devise an efficient greedy online algorithm. However, the greedy algorithm can be trapped into local optimal solutions easily. We then present a threshold-based randomized algorithm that not only guarantees a tighter competitive ratio but also includes an adaptive optimization technique, which can quickly learn the optimal threshold for the randomized algorithm. Finally, we verify the effectiveness and efficiency of the proposed methods through extensive experiments on real and synthetic datasets.**

I. Introduction

The prevalence of mobile Internet and sharing economy has led to the emergence of all kinds of *spatial <u>c</u>rowdsourcing* (SC) platforms, where the *online crowd workers* are employed through their smartphones to participate in and complete *offline crowdsourcing tasks* in the physical world [1]. Many representative commercial applications of spatial crowdsourcing are ubiquitous in daily life, such as the real-time taxi-calling service, e.g. Uber [2], the product placement checking service of supermarkets, e.g. Gigwalk [3], the on-wheel meal-ordering service, e.g. GrubHub [4].

In such spatial crowdsourcing platforms, one of the most important issues is to assign real-time tasks to suitable crowd workers. Existing studies usually model this task assignment problem as a static maximum unweighted/weighted bipartite graph matching problem [5], [6], [7], [8], [9], which is to match tasks to suitable workers such that the total number/weighted value of the assigned pairs of tasks and workers is maximized. These studies only consider matching two types of objects, tasks and workers, in an offline scenario, where the full bipartite graph of all the tasks and workers is known before the matching is conducted.

However, some new emerging O2O applications, e.g. InterestingSport[10] and Nanguache[11], bring new challenges to the existing task assignment techniques in SC platforms. For example, InterestingSport needs to find suitable sports trainers and book the corresponding sports facilities for its users in real time. Imagine the following scenario. Diane is a tennis beginner and hopes to find a suitable trainer to practice tennis near her home once she has free time. However, Diane usually encounters a dilemma if she wants to practice tennis shortly. That is because she first needs to know the availability of her trainer and the nearby tennis court and it is hardly that both her trainer and the nearby tennis court are available at the same time. In fact, there may be some other available trainers and tennis courts near her home when Diane is free, and a more intelligent SC platform that can match customers, trainers and tennis courts in real time can solve Diane's dilemma. Another example is the Nanguache platform, which aims to help customers to find suitable makeup artists (or hairstylists). In particular, Nanguache would suggest a nearby 3rd-party workplace, e.g. salon, where the makeup artist and the customer would meet. Such assignment of makeup artist and 3rd-party workplace needs to be conducted in real time whenever a user makes a request.

For these new SC platforms, on one hand, three types of objects, tasks, workers and workplaces, need to be matched. On the other hand, the assignment is dynamic: tasks are usually submitted by customers in real time, workers (trainers or makeup artists) log in the platform dynamically, and the availability of facilities/workplaces is dynamically updated. Thus, the assignment for these new SC platforms should be conducted under *dynamic online scenarios*, where each task, worker and workplace may appear anywhere at anytime and requires immediate responses from the SC platforms. Notice that a recent study [12] also studies online allocation in SC, but it only considers two sides (tasks and workers) and cannot address the matching problem involving three types of objects. Therefore, the core challenge for these new emerging SC platforms is *how to model the new matching problem involving three types of dynamic objects and design a quality-guaranteed algorithm to conduct the online matching*.

As discussed above, some emerging SC platforms need to match three types of objects. Under the offline scenarios, such matching problem can be reduced to the classical 3-dimensional matching (3DM) problem [13], where tasks, workers and workplaces correspond to the three disjoint sets of nodes in a tripartite graph. Moreover, there is an edge between a task (worker) and a workplace if the workplace locates in

2375-026X/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDE.2017.147

TABLE I: Statistics for requesters, workers and workplaces

Object	Location	Arrival Time	Leaving Time
t_1	(4.50, 6.00)	8:00	8:10
p_1	(4.50, 4.75)	8:02	8:12
w_1	(5.50, 5.00)	8:05	8:15
t_2	(3.00, 4.50)	8:08	8:18
p_2	(2.50, 3.00)	8:10	8:20
w_2	(4.00, 3.25)	8:11	8:21
w_3	(3.25, 2.00)	8:13	8:23
t_3	(1.50, 3.50)	8:15	8:25
t_4	(5.50, 2.00)	8:17	8:27
p_3	(4.50, 2.00)	8:19	8:29

TABLE II: Utility of all possible matches

ID	Match	Utility Score	ID	Match	Utility Score
1	(t_1, p_1, w_1)	18	6	(t_2, p_2, w_3)	20
2	(t_1, p_1, w_2)	10	7	(t_3, p_2, w_2)	12
3	(t_2, p_1, w_1)	**90**	8	(t_3, p_2, w_3)	**48**
4	(t_2, p_1, w_2)	20	9	(t_4, p_3, w_2)	**72**
5	(t_2, p_2, w_2)	20	10	(t_4, p_3, w_3)	12

the limited range of the task requester (worker). Although there are some existing solutions for the offline 3DM problem, they cannot address the dynamic online scenarios, where the current decision can only be made based on partial tripartite graph information and needs to be conducted immediately. To further illustrate this motivation, we go through a toy example as follows.

Example 1: Suppose we have four task requesters $t_1 - t_4$, three workers $w_1 - w_3$ and three workplaces $p_1 - p_3$ on a SC platform, and their corresponding locations are shown in a 2D space (X,Y) in Fig. 1 (a). Each requester and each worker have a limited activity range, which is shown as the dotted circle and the solid circle in Fig. 1 (a), respectively. A requester (worker) can only be matched to the workplaces that are located within the range of the requester (worker). Table. II shows the utility scores of each triple, which represents the satisfaction of the matching involving the corresponding task requester, worker and workplace. The arrival and leaving time of the objects are shown in the last two columns of Table. I. Under the offline scenario, the matching problem can be modeled as a variant of the 3DM problem shown in Fig. 1 (b). Thus, the total utility of the optimal matching, which is shown in bold font in Table. II, is 210. However, under the dynamic online scenario, when a new object arrives, the platform has to process it immediately, and the decision cannot be changed once it was made. We next show a possible matching procedure in the SC platform. When w_1 arrives, the platform matches it with t_1 and p_1 or does nothing. Suppose the platform conducts the matching (t_1, p_1, w_1), and it then gains utility score of 18. Subsequently, the platform then further matches two triples, (t_3, p_2, w_3) and (t_4, p_3, w_2), and the total utility score is $18+48+72 = 138$. Notice that the total utility of the online matching cannot be greater than that of optimal total utility under the offline scenario.

Based on the aforementioned example, we formally define a new task assignment problem in real-time SC, called the *trichromatic online matching in real-time spatial crowdsourcing* (TOM) problem, where the three types of objects, task requesters, workers and workplaces, will dynamically arrive at the SC platform one by one in an online way and the platform has to make decision for a newly arrived object based on the current information before the next object arrives. In

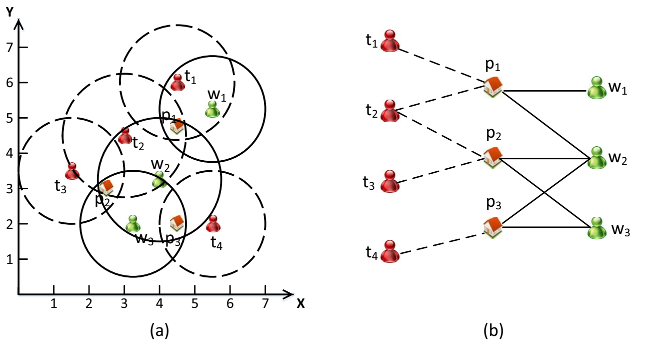


Fig. 1: Locations of requesters, workers and places.

particular, as discussed later, even the offline version of the TOM problem is NP-hard, and thus the TOM problem is very difficult to address. A close branch of existing studies is the online weighted bipartite graph matching problem [14], [15], [16], [17]. However, all of them only focus on two types of objects and cannot handle three types of objects. In summary, to solve this problem, we make the following contributions.

- Inspired by some emerging SC applications, we identify a new dynamic matching problem, called the *trichromatic online matching in real-time spatial crowdsourcing* (TOM) problem, and propose a formal definition of TOM problem. We also prove that even the offline version of TOM is NP-hard.
- We develop a greedy algorithm, which is efficient but has worse theoretical guarantee since it can be trapped into local optimal solutions easily.
- We present a threshold-based randomized algorithm framework, which not only guarantees a tighter competitive ratio, $\frac{1}{3e\lceil ln(U_{max}+1)\rceil}$, where U_{max} is the maximum utility of a triple involving task, worker and workplace, but also includes an adaptive optimization technique, which can learn the best threshold for the randomized algorithm.
- We verify the effectiveness and efficiency of the proposed online methods with extensive experiments on real and synthetic datasets.

The rest of the paper is organized as follows. We define the TOM problem in Section II. Section III presents a greedy-based online algorithm and analyzes its competitive ratio. To enhance the effectiveness of the greedy-based algorithm, a threshold-based framework and its adaptive optimization techniques with better competitive ratios are presented in Section IV. Section V discusses extensive experiment results on both synthetic and real datasets. We review related works in Section VI and conclude in Section VII.

II. Problem Statement

We first formally define the *trichromatic online matching in real-time spatial crowdsourcing* (TOM) problem, and then introduce competitive ratio, which measures the quality of an online algorithm compared with the offline optimal result. Finally, we prove that the offline version of the TOM problem is NP-hard.

A. Problem Definition

We first introduce three basic concepts, task requester, crowd worker and crowd workplace, and then formally define the online model of TOM and the utility score of a single match, i.e., a triple of a requester, a workplace and a worker. We define the TOM problem.

Definition 1 (Task Requester): A task requester ("requester" for short) is denoted by $t =< \boldsymbol{l}_t, r_t, b_t, e_t >$. $\boldsymbol{l}_t$ is the location of t in a 2D space. b_t and e_t are the arrival and leaving time of t, respectively. r_t is the radius of the limited circular range of t, whose center is $\boldsymbol{l}_t$.

Definition 2 (Crowd Worker): A Crowd worker ("worker" for short) is denoted by $w =< \boldsymbol{l}_w, r_w, b_w, e_w >$. $\boldsymbol{l}_w$, r_w, b_w and e_w represent the location, the radius, the arrival time and leaving time of w, respectively, which are similar as those of task requester.

Definition 3 (Crowd Workplace): A Crowd workplace ("workplace" for short) is denoted by $p =< \boldsymbol{l}_p, b_p, e_p >$. Particularly, a workplace p has a location $\boldsymbol{l}_p$, an arrival time b_p and a leaving time e_p.

After defining the above three basic concepts, we next introduce the online model used in the TOM problem.

Definition 4 (Online Model): A three-sided online model is adopted. Concretely, three types of objects, requesters, workers and workplaces, arrive and leave dynamically on the crowdsourcing platform. On the arrival of each object, the platform matches it to the other two types of objects actively. Alternatively, an object can wait to be matched to subsequent arrival objects passively. A match cannot be changed once it is made.

We then define the utility score of a triple.

Definition 5 (Utility Score): The utility score of a triple (t, p, w) of requester t, workplace p and worker w is defined as $U(t, p, w)$, where U can be any function derived from the profiles and spatial-temporal information of t, p and w.

Finally, we define our problem as follows.

Definition 6 (TOM Problem): Given a set of requesters T, a set of workers W, a set of workplaces P , and a utility function $U(.,.,.)$ on a SC platform, which has no object initially and allows each object to arrive and leave at any time, the TOM problem is to find a matching M among the requesters, the workers and the workplaces to maximize the total utility $MaxSum(M) = \sum_{t \in T, p \in P, w \in W} U(t, p, w)$ such that the following constraints are satisfied:

- Deadline constraint: each object should either be matched to existing objects at the arrival time, or be matched to new arrival objects before its leaving time, or otherwise remains unmatched after the deadline.
- Invariable constraint: once a triplet (t, p, w) of requester t, workplace p and worker w is matched, it cannot be changed.
- Range constraint: any workplace matched to a requester t and a worker w must locate in the restricted range of both t and w.

B. Evaluation Model

Competitive analysis [14], [15], [16] is a method for analyzing online algorithms, which compares the performance of an online algorithm to that of an optimal offline algorithm has full information in advance.

Definition 7 (Competitive Ratio): The competitive ratio of a deterministic online algorithm for the TOM problem is the following minimum ratio between the result of the online algorithm and the optimal result over all possible inputs,

$$CR = \min_{\forall G(T,W,P,U)} \frac{MaxSum(M)}{MaxSum(OPT)}$$

where $G(T, W, P, U)$ is an arbitrary input of requesters, workers, workplaces and the utility score function, $MaxSum(M)$ is the total utility score produced by the deterministic online algorithm. $MaxSum(OPT)$ is the optimal total utility score of the offline scenario.

The competitive ratio of a randomized online algorithm is

$$CR = \min_{\forall G(T,W,P,U)} \frac{E[MaxSum(M)]}{MaxSum(OPT)}$$

where $E[MaxSum(M)]$ is the expectation of the total utility scores produced by the randomized online algorithm, and the expectation comes from the coin flip in the randomized algorithm.

C. Hardness of the Offline Version of TOM

The offline version of the TOM problem is identical to the online TOM problem except that the first two constraints are excluded. In other words, the spatiotemporal information of all the requesters, workers and workplaces is known before the matching is conducted. We next analyze the hardness of the offline version of the TOM algorithm.

Theorem 1: The offline version of the TOM problem is NP-hard.

Proof: We consider the special case of the offline version of the TOM problem, where $|T| = |W| = |P|$, and the utility score of any triple that satisfies all the constraints is 1. Since the decision problem of the 3-dimensional matching (3DM) problem, which is a well-known NP-complete problem [13], is equivalent to that of the aforementioned special case, we can reduce the 3DM problem to the special case of the offline version of the TOM problem. Therefore, the offline version of the TOM problem is NP-hard. ■

III. Greedy Algorithm

In this section, we propose and analyze a straightforward solution of TOM, the Greedy algorithm.

The main idea is that when a new object arrives, Greedy finds all the triples that can be matched to the new object and chooses the one with the largest utility score with ties broken arbitrarily. Note that the greedy strategy is reflected in two aspects. First, Greedy will make a matching whenever possible, even if the corresponding utility score is low. Second, if several triples are available at the same time, Greedy always chooses the one with the largest utility score.

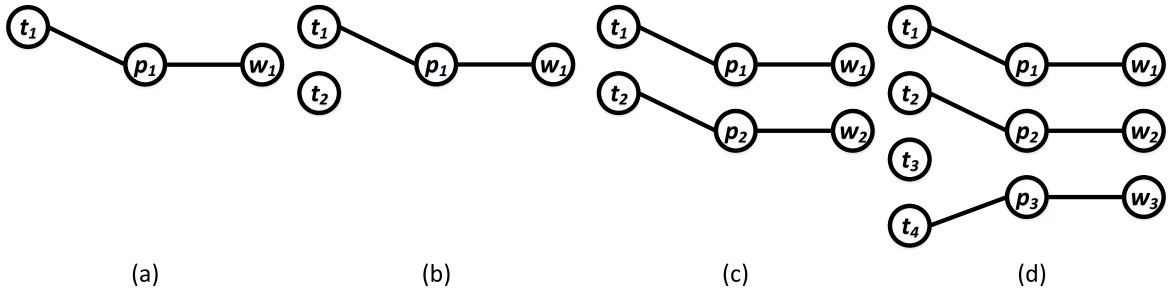


Fig. 2: Illustrated example of Greedy.

Algorithm 1: Greedy Algorithm

```
input : $T, W, P, U(.,.)$
output: A feasible matching result $M$
1 foreach new arrival object $v$ do
2     $Cand \leftarrow \{\forall u | u$ is a triple containing $v$ and satisfying all the constraints$\}$;
3     if $Cand = \emptyset$ then
4         continue;
5     else
6         $m \leftarrow$ the item in $Cand$ with the largest utility;
7         $M \leftarrow M \cup \{m\}$;
8 return $M$
```

The whole procedure of Greedy is illustrated in Algorithm 1. In lines 1-2, when a new object v arrives, Greedy puts all the triples that contain v and satisfy all the constraints into the candidate set $Cand$. In lines 3-7, if $Cand$ is not empty, Greedy chooses the triple with the largest utility from $Cand$, and adds it to the matching result M.

Example 2: Back to our running example in Example 1. According to the arrival time in Table. I, the matching procedure of Greedy is shown in Fig. 2. In Fig. 2 (a), when w_1 arrives, Greedy matches (t_1, p_1, w_1) with utility of 18. When t_2 arrives in Fig. 2 (b), the candidate set $Cand = \emptyset$ because p_1 and w_1 have been matched. After w_2 arrives, which is shown in Fig. 2 (c), Greedy matches (t_2, p_2, w_2) with utility of 20. After t_4 arrives, which is shown in Fig. 2 (d), Greedy matches (t_4, p_2, w_3) with utility of 12. Thus, the total utility is $18 + 20 + 12 = 50$.

Competitive Analysis. Next, we analyze the competitive ratio of Greedy.

Lemma 1: If M_{OPT} and M_{GRD} are the matching results of the offline optimal algorithm and Greedy respectively, we have $|M_{GRD}| \geq \frac{|M_{OPT}|}{3}$, where $|\cdot|$ represents cardinality of a set.

Proof: For each triple m in M_{OPT}, at least one object of m must be matched by Greedy. Therefore the number of objects matched by Greedy is at least $\frac{1}{3}$ of those matched by the offline optimal algorithm. Thus we have $|M_{GRD}| \geq \frac{|M_{OPT}|}{3}$. ■

Theorem 2: If the utility score of each triple is in the range of [1, U_{max}], the competitive ratio of Greedy is $\frac{1}{3U_{max}}$.

Proof:

$$\begin{aligned} MaxSum(M_{GRD}) &\geq |M_{GRD}| \cdot 1 \\ &\geq \frac{|M_{OPT}|}{3} \\ &\geq \frac{1}{3U_{max}} |M_{OPT}| \cdot U_{max} \\ &\geq \frac{MaxSum(M_{OPT})}{3U_{max}} \end{aligned}$$

Thus, the competitive ratio of Greedy is

$$CR = \frac{MaxSum(M_{GRD})}{MaxSum(M_{OPT})} = \frac{1}{3U_{max}}.$$

■

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

IV. Threshold-Based Algorithms

From Example 2, we can observe that Greedy does not perform well. In particular, when Greedy matches a triple with a small utility score, subsequent triples with large utility scores may not be matched. To address the above limitation, we propose a threshold-based framework. Basically, we filter the triples whose utility scores are less than a certain threshold so that they will not block subsequent triples that may have larger utility scores. Inspired by [17], we devise a randomized algorithm called Basic-Threshold in Section IV-A. We next enhance Basic-Threshold to Adaptive-Threshold that learns the optimal threshold in Section IV-B.

A. Basic-Threshold Algorithm

The main idea of Basic-Threshold is to first randomly choose a threshold on the utility of triples. Then for each new arrival object, an arbitrary triple containing the new object with utility no less than the chosen threshold is added to the matching result.

The whole procedure of Basic-Threshold is illustrated in Algorithm 2. In lines 1-2, Basic-Threshold randomly chooses a threshold e^k on the utility of triples according to the estimated

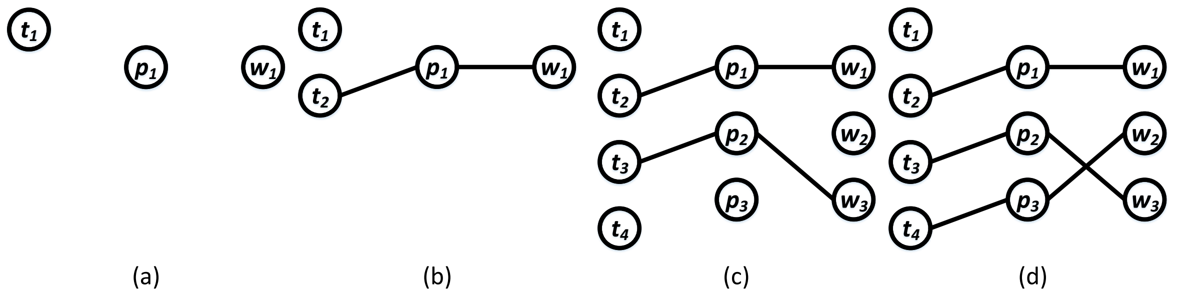


Fig. 3: Illustrated example of Basic-Threshold.

Algorithm 2: Basic-Threshold Algorithm

```
   input  : $T, W, P, U(.,.,.)$
   output: A feasible match result $M$
1  $\theta \leftarrow \lceil ln(U_{max}+1) \rceil$;
2  $k \leftarrow$ randomly choosing an integer from $\{0, \cdots, \theta-1\}$;
3  foreach new arrival requester or worker or workplace $v$ do
4      $Cand \leftarrow \{\forall u | u$ is an allocation containing $v$ and satisfying all the constraints, and the utility score of $u$ calculated by $U(.,.,.)$is no less than $e^k\}$;
5      if $Cand = \emptyset$ then
6          continue;
7      else
8          $m \leftarrow$ an arbitrary item in $Cand$;
9          $M \leftarrow M \cup \{m\}$;
10 return $M$
```

maximum utility U_{max}, which can be learned from historical records on the crowdsourcing platforms. In lines 3-4, when a new object v arrives, which may be a requester, a worker or a workplace, Basic-Threshold then adds all triples of v that have utility scores no less than e^k and satisfy all the constraints into $Cand$. In lines 5-9, if $Cand$ is not empty, Basic-Threshold chooses an arbitrary triple from $Cand$ and adds it to M.

We then explain Basic-Threshold with a running example.

Example 3: Back to our running example in Example 1. The Basic-Threshold algorithm set $\theta = \lceil ln(90+1) \rceil = 5$, so $k \in \{0, \cdots, 4\}$. If k is chosen as 3, the threshold is $e^3 \approx 20.1$. The matching procedure of Basic-Threshold is shown in Fig. 3. According to the arrival time in Table. I, when w_1 arrives, which is shown in Fig. 3 (a), the candidate set $Cand = \emptyset$ because the utility of (t_2, p_1, w_1) is less than the threshold. When t_2 arrives in Fig. 3 (b), the candidate set $Cand = \{(t_2, p_1, w_1)\}$, and the algorithm matches the triple of (t_2, p_1, w_1). Similarly, in Fig. 3 (c), when t_3 arrives, Basic-Threshold filters (t_3, p_2, w_2) with utility score of 12, and matches (t_3, p_2, w_3). When p_3 arrives in Fig. 3 (d), Basic-Threshold matches (t_4, p_3, w_2). Thus, when $k = 3$, the total utility is 210. Since Basic-Threshold is a randomized algorithm on choosing k, the expectation of the total utility for all possible k is $(50 + 50 + 50 + 210 + 162)/5 = 104.4$.

We next give the competitive ratio and the complexity of Basic-Threshold.

Theorem 3: The competitive ratio of Basic-Threshold is $\frac{1}{3e\lceil ln(U_{max}+1) \rceil}$.

Proof: We denote G as the corresponding weighted tripartite graph of the TOM input. Let $G_{[e^i, e^{i+1})}$ be a subgraph of G, which only contains the edges whose utility scores lies in $[e^i, e^{i+1})$. Let $OPT_{[e^i, e^{i+1})}$ and $M_{[e^i, e^{i+1})}$ be the optimal match and the one returned by Basic-Threshold on $G_{[e^i, e^{i+1})}$, respectively. According to Lemma 1, $|M_{[e^i, e^{i+1})}| \geq |OPT_{[e^i, e^{i+1})}|/3$. Then, we have

$$\begin{aligned} E[MaxSum(M)] &= \frac{1}{\theta} \sum_{i=0}^{\theta-1} MaxSum(M_{[e^i, e^{i+1})}) \\ &\geq \frac{1}{\theta} \sum_{i=0}^{\theta-1} e^i |M_{[e^i, e^{i+1})}| \\ &\geq \frac{1}{\theta} \sum_{i=0}^{\theta-1} e^i \frac{|OPT_{[e^i, e^{i+1})}|}{3} \\ &\geq \frac{1}{3e\theta} \sum_{i=0}^{\theta-1} MaxSum(OPT_{[e^i, e^{i+1})}) \\ &\geq \frac{1}{3e\theta} MaxSum(OPT) \end{aligned}$$

Thus, the competitive ratio of Basic-Threshold is

$$CR = \frac{E[MaxSum(M)]}{MaxSum(OPT)} = \frac{1}{3e\theta} = \frac{1}{3e\lceil ln(U_{max}+1) \rceil}$$

since $\theta = \lceil ln(U_{max}+1) \rceil$, where U_{max} is the maximum utility score of the TOM input. ∎

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

B. Adaptive-Threshold Algorithm

The performance of Basic-Threshold is not stable, because different thresholds have significant influence on the matching results, which is shown in Example 3. Thus, how to choose an appropriate threshold becomes a new challenge. To solve the problem, we devise the Adaptive-Threshold algorithm, which combines with the polynomial weights algorithm [18] and can adaptively adjust the probability distribution of choosing

Algorithm 3: Adaptive-Threshold Algorithm

```
input : $T, W, P, U(.,.,.)$
output: A feasible match result $M$
1  $\theta \leftarrow \lceil ln(U_{max}+1) \rceil$;
2  set $w_i = 1$ for $i = 0, 1, \cdots, \theta-1$;
3  set $W = \sum_{i=1}^{\theta-1} w_i$;
4  set $p_i = \frac{w_i}{W}$ for $i = 0, 1, \cdots, \theta-1$;
5  foreach new arrival object $v$ do
6      $k \leftarrow$ randomly choosing an integer from $\{0, 1, \cdots, \theta-1\}$ according to $\vec{p} = (p_0, p_1, \cdots, p_{\theta-1})$;
7      $Cand \leftarrow \{\forall u | u$ is a match containing $v$ and satisfying all the constraints, and the utility score of $u$ calculated by $U(.,.,.)$is no less than $e^k\}$;
8      if $Cand = \emptyset$ then
9          continue;
10     else
11         $m \leftarrow$ an arbitrary match in $Cand$;
12         $M \leftarrow M \cup \{m\}$;
13     Update $w_i = w_i(1 + \eta u_i)$ for $i = 0, 1, \cdots, \theta-1$, where $u_i$ is the utility score got from $v$ if using $e^i$ as the threshold from the beginning;
14     Update $W$ and $\vec{p}$;
15 return $M$
```

different thresholds according to the information that has been provided.

The main idea of Adaptive-Threshold is to combine all the fixed thresholds strategies probabilistically. In particular, once a new object arrives, we choose a threshold according to some probability distribution to filter triples with low utility scores. After processing each object, for each threshold, we calculate how well we would have done if we used the threshold from the beginning. According to the above information, we adjust the probability distribution of choosing different thresholds adaptively.

The whole procedure of the Adaptive-Threshold algorithm is illustrated in Algorithm 3. In line 1, Adaptive-Threshold first calculates the value of θ according to U_{max}, which is the same with Basic-Threshold. In lines 2-3, for each threshold e^i, we initially set the weight at 1, and the total weight $W = \sum_{i=1}^{\theta-1} w_i$. Thus, the initial probability for e^i is $p_i = w_i/W$. In lines 4-13, when a new object arrives, we choose a threshold e^k according to $\vec{p} = (p_0, p_1, \cdots, p_{\theta-1})$, and use it to filter matches, just as what Basic-Threshold does. After processing v, we update w_i following $w_i = w_i(1+\eta u_i)$ for $i = 0, 1, \cdots, \theta-1$, where u_i is the utility got from v if we use e^i as the threshold from the beginning, and the parameter η is used to control the rate of updating w_i, which will be explained later.

We next use a running example to explain how Adaptive-Threshold works.

Example 4: Back to our running example in Example 1. In this example, we use $\eta = 0.001$ to show how we update the weights of choosing different thresholds. The procedure of how w_i updates is shown in Table. III. Initially, Adaptive-Threshold sets $\theta = \lceil ln(90+1) \rceil = 5$, so $k \in \{0, \cdots, 4\}$, and $w_i = 1$, $p_i = 1/5$ for $i \in \{0, \cdots, 4\}$. After w_1 arrives, if we use $k = 0, 1, 2$ from the beginning, we can gain the utility of 18 from (t_1, p_1, w_1), and the corresponding weights will be updated to 1.02 in the third column of Table. III. After t_2 arrives, if we use $k = 3, 4$ from the beginning, utility of 90 will be gained and the corresponding weights will be updated to 1.09 in the fourth column of Table. III. After w_2 arrives, if we use $k = 0, 1, 2$ from the beginning, we can gain the utility of 20 from (t_1, p_1, w_1), and the corresponding weights will be updated to 1.04 in the fifth column of Table. III. After t_3 arrives, if we use $k = 3$ from the beginning, utility of 48 will be gained and the corresponding weights will be updated to 1.14 in the sixth column of Table. III. After p_3 arrives, if we use $k = 3, 4$ from the beginning, utility of 72 will be gained and the weights of e^3 and e^4 will be updated to 1.22 and 1.17 respectively in the last column of Table. III. After the whole procedure, it can be observed that the threshold e^3 has the largest weight, which means if there are new objects arriving, the probability of choosing e^3 to filter triples is the largest, as it performs best on the objects that have arrived, which is shown in Example 3.

TABLE III: Updating procedure of Adaptive-Threshold

Threshold	t_1	w_1	t_2	w_2	t_3	p_3
e^0	1	1.02	1.02	1.04	1.04	1.05
e^1	1	1.02	1.02	1.04	1.04	1.05
e^2	1	1.02	1.02	1.04	1.04	1.05
e^3	1	1	1.09	1.09	1.14	1.22
e^4	1	1	1.09	1.09	1.09	1.17

In Basic-Threshold, a threshold is chosen and used during the whole matching procedure. However, as mentioned, the performance of Basic-Threshold is not stable, since the total utility of Basic-Threshold using different thresholds varies significantly. We cannot know which threshold can achieve the best performance until the whole matching procedure is finished. We next prove that Adaptive-Threshold can always achieve nearly the best possible performance of Basic-Threshold with different thresholds in hindsight, which is shown in Theorem 4. Note that if we use the threshold which achieves the best performance in hindsight, a competitive ratio which is not worse than that of Basic-Threshold will be achieved.

Theorem 4: For any $\varepsilon > 0$, if we use $\eta = \varepsilon/D$, and under the assumption that $D \geq U_{max}$, Adaptive-Threshold produces an expected total utility score of at least

$$(1-\varepsilon)MaxSum(OPT) - \frac{\varepsilon(1-\varepsilon)}{2D}\sum_v (g_v^*)^2 - \frac{D(1-\varepsilon)}{\varepsilon}ln(\theta)$$

where G_{OPT} is the match result of the maximum total utility of Basic-Threshold with different thresholds in hindsight, g_v^* is the utility of processing object v using the best threshold, and D is the upper bound of the cost incurred by switching thresholds (how to estimate the value of D will be discussed later).

Proof: Suppose Adaptive-Threshold currently has weights $w_0, w_1, \cdots, w_{\theta-1}$, observes a utility score vector $(u_0, u_1, \cdots, u_{\theta-1})$, then updates the weights to $w'_0, w'_1, \cdots, w'_{\theta-1}$, where $w'_i = w_i(1+\eta u_i)$. Let $W = \sum_{i=1}^{\theta-1} w_i$ and $W' = \sum_{i=1}^{\theta-1} w'_i$. The expected gain of Adaptive-Threshold for object v is $\sum_{i=1}^{\theta-1} p_i u_i$, which we will call E_v. The expected cost due to moving from probability distribution $\vec{p}$ to distribution $\vec{p'}$ is at most

$$\begin{aligned} D \sum_{i:p'_i > p_i} (p'_i - p_i) &\le \frac{D}{W} \sum_{i:p'_i > p_i} (w'_i - w_i) \\ &\le \frac{D}{W}(W' - W) \\ &= D(\frac{W'}{W} - 1). \end{aligned}$$

According to the method of updating weights of different thresholds,

$$W' = \sum_{k=0}^{\theta-1} w_i(1+\eta u_i) = W + \sum_{k=0}^{\theta-1} \eta w_i u_i = W(1+\eta E_v).$$

Thus, the expected gain of Adaptive-Threshold is

$$MaxSum(AT) = \sum_v E_v(1-\varepsilon).$$

Let W_f be the total weight in the end, so

$$W_f = \theta \prod_v (1+\eta E_v).$$

Using the fact that $\forall x \in [0,1], ln(1+x) \le x$, we have

$$\begin{aligned} ln(W_f) &= ln(\theta) + \sum_v ln(1+\eta E_v) \\ &\le ln(\theta) + \eta \sum_v E_v. \end{aligned}$$

Suppose the gains of the optimal threshold for each object v is g^*_v, and then we have

$$\begin{aligned} W_f &\ge \prod_v (1+\eta g^*_v), \\ ln(W_f) &\ge \sum_v ln(1+\eta g^*_v). \end{aligned}$$

Using the fact that $\forall x \in [0,1], ln(1+x) \ge x - \frac{x^2}{2}$, we have

$$ln(W_f) \ge \sum_v (\eta g^*_v - \frac{(\eta g^*_v)^2}{2}).$$

Therefore,

$$\begin{aligned} \eta \sum_v E_v + ln(\theta) &\ge \sum_v g^*_v - \frac{\eta^2}{2} \sum_v (g^*_v)^2, \\ MaxSum(AT) &\ge (1-\varepsilon) MaxSum(OPT) - \\ &\frac{\varepsilon(1-\varepsilon)}{2D} \sum_v (g^*_v)^2 - \frac{D(1-\varepsilon)}{\varepsilon} ln(\theta). \end{aligned}$$

■

Discussion of D. D is the upper bound of the cost incurred by switching thresholds. If we assume that the utilities of all triples are in $[1, U_{max}]$, and the number of objects that exist on the platform at any time is at most N, then we have $D \le N \cdot U_{max}$.

Complexity Analysis. For each new arrival object, the time and space complexity of the Adaptive-Threshold algorithm are both $O(\lceil ln(U_{max}+1) \rceil max(|T||W|, |T||P|, |W||P|))$.

TABLE IV: Real Dataset

Dataset	$\|T\|$	$\|W\|$	$\|P\|$	Waiting Time
gMission	630	576	427	5,10,15,20,25
SportsCompany	856	508	179	5,10,15,20,25

TABLE V: Synthetic Dataset

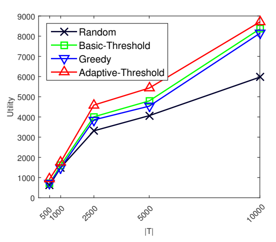
Factor	Setting
$\|T\|$	500, 1000, **2500**, 5000, 10000
$\|W\|$	500, 1000, **2500**, 5000, 25000
$\|P\|$	500, 1000, **2500**, 5000, 25000
μ of position (Normal Distribution)	10, 25, **50**, 70, 90
λ^{-1} of position (Exponential Distribution)	10, 25, **50**, 70, 90
μ of utility (Normal Distribution)	10, 25, **50**, 70, 90
λ^{-1} of utility (Exponential Distribution)	10, 25, **50**, 70, 90
r_w	5, 10, **15**, 20, 25
Waiting Time	5, 10, **15**, 20, 25
Scalability	$\|T\|$ = 10K, 20K, 30K, 40K, 50K, 100K $\|W\|$,$\|P\|$ = 7500

V. Experimental Study

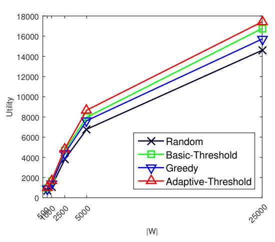
A. Experiment Setup

Datasets. We use two real datasets, the gMission dataset and the SportsCompany dataset. gMission [19] is a research-based general spatial crowdsourcing platform. In the gMission dataset, every requester has a task description, a location, a release time, a deadline and a payoff. Each worker is also associated with a location, a release time, a deadline, and a success ratio based on his/her historical records on completing tasks. Each worker place has a location, a release time and a deadline. We use the product of task's reward and worker's success ratio as the utility of a triple. SportsCompany is a crowdsourcing application where users can find sports venues and trainers. In the SportsCompany dataset, users are viewed as requesters, who have a location, a release time and a deadline. Sports venues are viewed as workplaces which have a location, a release time, a deadline and a price. Trainers are viewed as workers who have a location, a release time and a price. The utility of a triple is the sum of the price of the trainer and the sports venue. We generate the ranges of the requesters and the workers as there is not range information in the real datasets. The statistics of real data are illustrated in Table. IV. We also use a synthetic dataset for evaluation. We generate the utility scores of triples and the locations of objects following Normal and Exponential distribution respectively. The value of U_{max} is set to be 100. The statistics and configuration of synthetic data are illustrated in Table. V, where we mark our default settings in bold font.

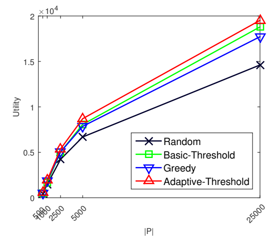
We evaluate the Random, Greedy, Basic-Threshold and Adaptive-Threshold algorithms in terms of total utility, *the average time of processing each object* and memory cost. Random is used to be the baseline algorithm. When a new object v arrives, Random finds all the triples that contain v and matches one from them randomly. We study the effect of varying parameters on the performance of the algorithms. In each experiments, we repeat 100 times and report the average results. The algorithms are implemented in Visual C++ 2015, and the experiments were conducted on a machine with Intel(R) Core(TM) i7-4710MQ 2.5GHz CPU and 8GB main memory.



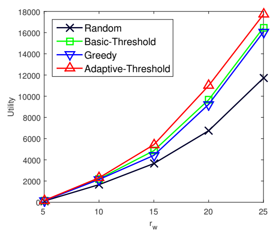
(a) Utility of varying $|T|$



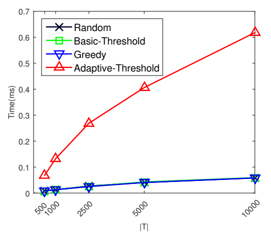
(b) Utility of varying $|W|$



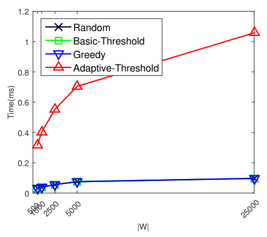
(c) Utility of varying $|P|$



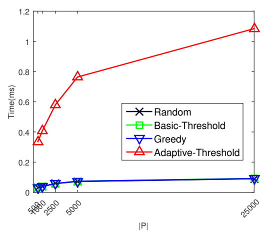
(d) Utility of varying r_w



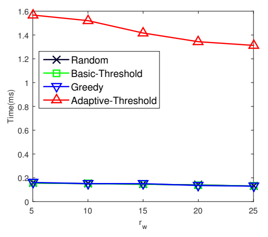
(e) Runtime of varying $|T|$



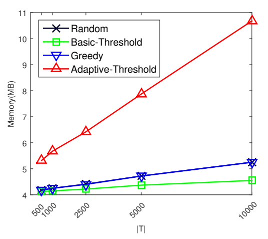
(f) Runtime of varying $|W|$



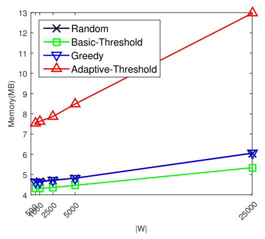
(g) Runtime of varying $|P|$



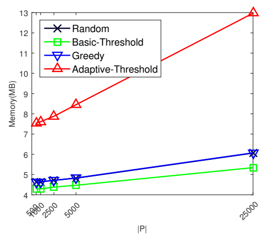
(h) Runtime of varying r_w



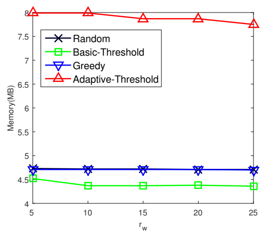
(i) Memory of varying $|T|$



(j) Memory of varying $|W|$



(k) Memory of varying $|P|$



(l) Memory of varying r_w

Fig. 4: Results on varying the cardinality objects and the range of workers.

B. *Experiment Results*

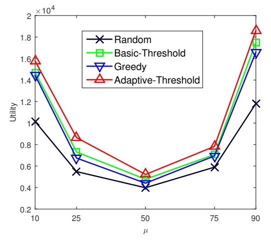
Effect of cardinality. We first study the effect of varying $|T|$. The first column of Fig. 4 presents the results when $|T|$ varies from 500 to 10000. In terms of total utility, we can first observe that the total utility scores increase as $|T|$ increases, which is natural as there are more triples that can be matched when $|T|$ increases. Second, we can observe that Adaptive-Threshold performs the best, followed by Basic-Threshold, Greedy and Random. The reasons are as follows: 1) Adaptive-Threshold can quickly learn the optimal threshold to filter triples; 2) some thresholds do not perform well and thus the expected performance of Basic-Threshold is affected; 3) the Greedy algorithm is trapped in local optimal solutions. As for the average time of processing each object and the memory cost, we can first observe that with the increment of $|T|$, the average running time increases. A possible reason is that when a new object arrives, more possible triples have to be considered. Second, we can observe that Adaptive-Threshold consumes the longest time and the most memory. The reason is that Adaptive-Threshold has to record the running history of all thresholds to learn the optimal threshold. However, we can find that Adaptive-Threshold is still efficient, as it can process an object in less than 1 millisecond on average, and the memory cost is negligible compared to the main memory.

The experiment results of varying $|W|$ and $|P|$ are presented in second and third columns of Fig. 4, which have similar patterns to those of varying $|T|$, and we omit the detailed analysis due to limited space.

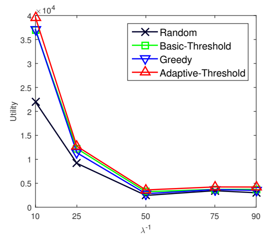
Effect of radius of limited range. We next study the effect of radius of workers. The last column of Fig. 4 shows the results when we vary r_w from 5 to 25. In terms of total utility, we can observe that the total utility scores increase as r_w increases as there are more possible triples to be matched. Second, Adaptive-Threshold still performs the best, followed by Basic-Threshold, Greedy and Random successively. Third, we can observe that the gaps between Random and the other algorithms tend to be larger. A possible reason is that when the radius of limited range is small, fewer triples can be chosen from when processing a new arrival object. As a result, different algorithms achieve the similar results. As for the time and memory cost, we can observe that Adaptive-Threshold consumes the longest time and the most memory. However, Adaptive-Threshold is still efficient enough, as each object can be processed in less than 1.5 milliseconds on average, and the memory cost is negligible.

We also study the results of varying the radius of tasks. The results have similar trending patterns, and we do not present them for brevity.

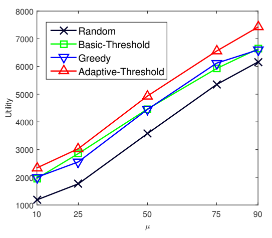
Effect of distribution of locations. We next study the effect of μ of objects' coordinates under Normal distribution.



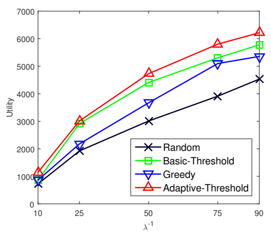
(a) Utility of varying μ of position



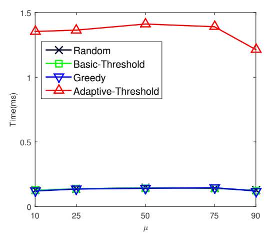
(b) Utility of varying λ^{-1} of position



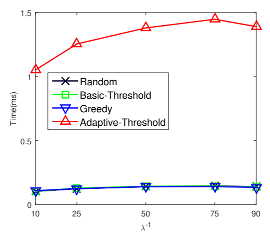
(c) Utility of varying μ of utility value



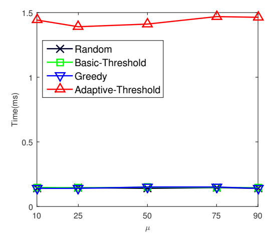
(d) Utility of varying λ^{-1} of utility score



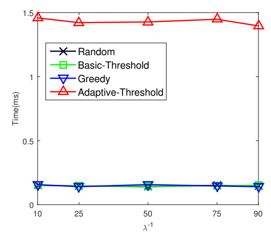
(e) Runtime of varying μ of position



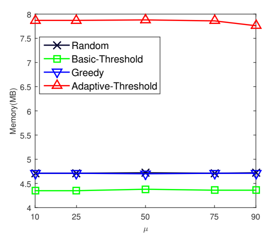
(f) Runtime of varying λ^{-1} of position



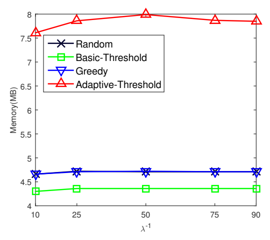
(g) Runtime of varying μ of utility score



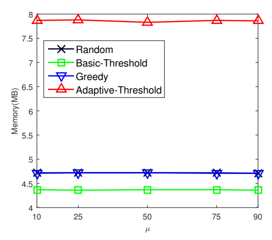
(h) Runtime of varying λ^{-1} of utility value



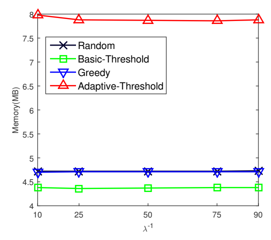
(i) Memory of varying μ of position



(j) Memory of varying λ^{-1} of position



(k) Memory of varying μ of utility score



(l) Memory of varying λ^{-1} of utility score

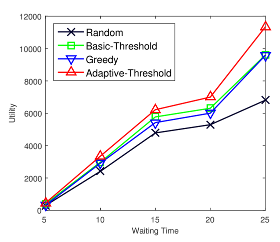
Fig. 5: Results on varying the distribution of locations of objects and utility of triples

The first column of Fig. 5 shows the results of varying μ from 10 to 90. In terms of total utility, we can observe that the total utility scores increases as μ gets farther away from 50. The reason is that the locations of all objects is in the range of $[0, 100] \times [0, 100]$, thus when μ is 50, the locations of objects are the most decentralized, leading to fewer triples that can be matched. Second, we can observe that Adaptive-Threshold still performs the best, followed by Basic-Threshold, Greedy and Random. Third, we can observe that when μ gets farther away from 50, the gaps between Random and the other three algorithms tends to be larger. The reason is that the locations of objects tend to be more centralized, thus Adaptive-Threshold, Basic-Threshold and Greedy can make a better choice from more available triples. As for the average time of processing each object and the memory cost, we can observe that although Adaptive-Threshold consumes the longest time and the most memory to record the running history of all thresholds, it is still efficient enough with negligible memory cost and time to process each object.

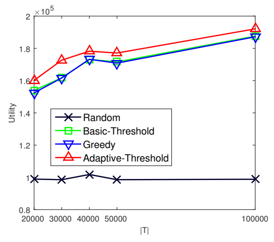
We next study the effect of λ^{-1} of objects' coordinates under Exponential distribution. The second column of Fig. 5 shows the results of varying λ^{-1} from 10 to 90. In terms of total utility, we can observe that the utility decreases as λ^{-1} increases. The reason is that the variance of Exponential distribution is λ^{-2}. Therefore, with the increment of λ^{-1}, the locations of objects are more decentralized, leading to fewer triples. Second, we can observe that Adaptive-Threshold still performs the best, followed by Basic-Threshold, Greedy and Random. Third, we can observe that when λ^{-1} decreases, the gaps between Random and the other three algorithms tends to be bigger. The reason is that locations of objects tend to be more centralized, providing Adaptive-Threshold, Basic-Threshold and Greedy more triples to choose from, while Random just matches triples randomly. As for the average time of processing each object and the memory cost, we can observe similar patterns as those of the aforementioned experiment results.

We also study the results of varying the locations of objects under power law distribution and uniform distribution. The results have similar trending patterns, and we do not present them for brevity.

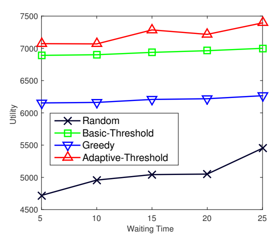
Effect of distribution of utility. We next study the effect of μ of utility under Normal distribution. The third column of Fig. 5 shows the results of varying μ from 10 to 90. In terms of total utility, we can observe that the utility increases as μ increases. The reason is that with the increment of μ,



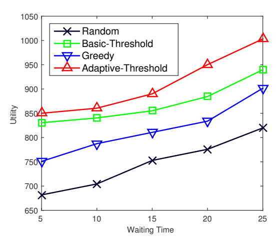
(a) Utility of varying waiting time



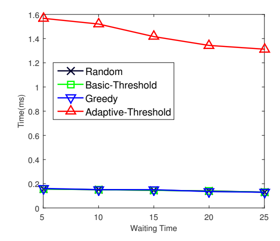
(b) Utility of scalability



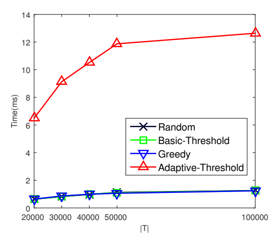
(c) Utility of gMission



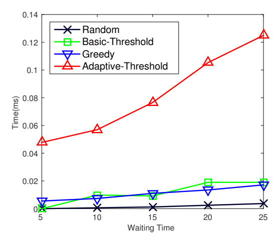
(d) Utility of SportsCompany



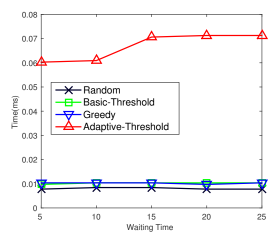
(e) Runtime of varying waiting time



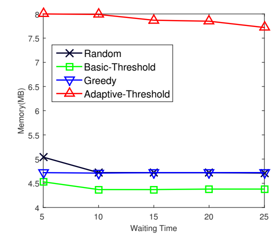
(f) Runtime of scalability



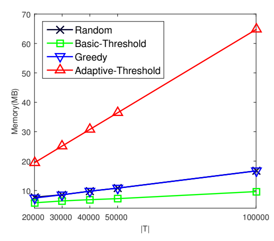
(g) Runtime of gMission



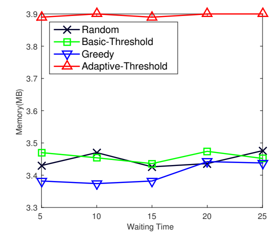
(h) Runtime of SportsCompany



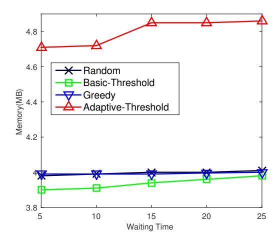
(i) Memory of varying waiting time



(j) Memory of scalability



(k) Memory of gMission



(l) Memory of SportsCompany

Fig. 6: Results on varying waiting time, scalability and real data

each triple tends to have a higher utility score. Second, we can observe that Adaptive-Threshold still performs the best. However, Basic-Threshold cannot always perform better than Greedy. A possible reason is that some thresholds used in Basic-Threshold perform bad. With respect to the average time of processing each object and the memory cost, Adaptive-Threshold consumes the longest time and the most memory because it has to record the running history of all available thresholds.

We next study the effect of λ^{-1} of utility scores under Exponential distribution. The last column of Fig. 5 shows the results of varying λ^{-1} from 10 to 90. In terms of total utility scores, we can observe that the total utility scores increase as λ^{-1} increases. The reason is that with the increment of λ^{-1}, each triple tends to have a higher utility score. Second, we can observe that Adaptive-Threshold still performs the best, followed by Basic-Threshold, Greedy and Random successively. As for the average time of processing each object and the memory cost, Adaptive-Threshold is efficient enough, which is similar with the aforementioned experiments.

We also study the results of varying the utility scores following power law distribution and uniform distribution. The results have similar trending patterns, and we do not present them for brevity.

Effect of waiting time. We next study the effect of waiting time. The first column of Fig. 6 presents the results when waiting time varies from 5 to 25. In terms of total utility scores, we can first observe that the utility scores increase with the increment of waiting time. The reason is that longer waiting time leads to more possible triples. Second, we can observe that Adaptive-Threshold performs the best, followed by Basic-Threshold, Greedy and Random. Third, the gaps between Random and the other three algorithms tend to increase, as the other three algorithms can choose from more available triples. As for the average time of processing each object and the memory cost, we can observe similar patterns to other experiment results.

Scalability. We next show the scalability of the algorithms. The experiment result is presented in the second column of Fig. 6. With respect to total utility scores, we can observe that the utility scores increase with the increment of $|T|$, which is natural as more triples can be matched. Besides, Adaptive-Threshold still performs the best, followed by Basic-Threshold, Greedy and Random successively. In terms of the average time of processing each object and the memory cost, we can observe that Adaptive-Threshold consumes the longest time and the most memory. However, Adaptive-Threshold can process each object in less than 15 milliseconds on average, and the memory cost is less than 70MB, which is quite small considering the main memory.

Real dataset. We finally present the experiment results on real dataset. The last two columns of Fig. 6 show the results on gMission and SportsCompany with varying waiting time of objects respectively. With respect to the total utility scores, it can be observed that the utility scores increase with the increment of waiting time and Adaptive-Threshold still performs the best. In terms of the average time of processing each object and the memory cost, we can observe that although Adaptive-Threshold consumes the longest time and the most memory, each object can be processed in less than 1 millisecond and the memory cost is negligible compared to the main memory.

C. Experiment Summary

In terms of total utility scores, Adaptive-Threshold can always achieve better result compared to Basic-Threshold and Greedy in both synthetic and real dataset. In most experiments, Basic-Threshold performs better than Greedy. The Random algorithm always performs the worst. In the aspect of the average time of processing each object and the memory cost, although Adaptive-Threshold performs the worst, it is efficient enough to satisfy high real-time requirements with low memory cost.

VI. Related Work

In this section, we review related works from two categories, matching problem and spatial crowdsourcing.

A. Matching Problem

Due to the static/dynamic characters, they are classified into two groups: offline matching and online matching.

1) Offline Matching: On offline matching, we review bipartite matching and 3-dimensional matching.

Bipartite Matching. The bipartite matching problem is a classical problem in the combinatorial optimization field [20]. The branch of bipartite matching problems related to the TOM problem is the maximum weighted bipartite matching (MWBM) problem. However, the MWBM problem only focuses on bipartite graphs rather than tripartite graphs and can be solved in polynomial time. Furthermore, some recent works study the bipartite matching problems in spatial data, called the spatial matching problems[21], [22], where the two sets of objects in a 2D space correspond to the two disjoint sets of vertices in the bipartite graph and the distance between two matched objects corresponds to the weight on the matched edge in the bipartite graph. Note that the above solutions of bipartite matching problems do not address the matching problem in tripartite graph and focus on the offline scenarios.

3-Dimensional Matching (3DM). As discussed in Section II, the offline version of TOM is equivalent to the optimization version of the classical 3DM problem [13], [23]. For the unweighted 3DM problems, existing studies prove that it is approximable within $3/2 + \varepsilon$ for any $\varepsilon > 0$. For the weighted 3DM problem, it is approximable within $2 + \varepsilon$ for any $\varepsilon > 0$ [24]. Although the decision problem of the offline TOM problem is equivalent to the 3DM problem, the existing solutions of 3DM problems usually adopt local search techniques, which need to know the full information of the tripartite graphs before conducting matching, and cannot be transferred to the online scenarios of the TOM problem, where the current decision can only be made based on partial tripartite graph information. In particular, to the best of our knowledge, there is no previous research about the online 3DM problem.

2) Online Matching: Another closely related issue is the online maximum weighted bipartite matching (OMWBM) problem, which is the online version of the MWBM problem[20]. A. Mehta provides a comprehensive survey regarding the OMWBM problem and its variants [16]. For the unweighted case of the OMWBM problem, the upper bound of the competitive ratio is proven to be $\frac{3}{4}$ [16]. For the OMWBM problem, [17] proposes the state-of-the-art randomized algorithm. These existing works only focus on the situations where only one side or two sides of vertices dynamically arrive. However, in our TOM problem, all three sides of vertices arrive in an online way. Since the offline version of the OMWBM problem can be solved in polynomial time, and that of the TOM problem is NP-hard, the TOM problem is harder than the OMWBM problem under the online scenario. Therefore, it is infeasible to directly extend the existing solutions of the OMWBM problem to solve the TOM problem.

To sum up, due to the high complexity and the dynamic requirement of the TOM problem, the aforementioned studies cannot be extended to solve TOM easily.

B. Spatial Crowdsourcing

Data-driven crowdsourcing has been a hot research topic. Recently, [25], [26], [27] provide comprehensive surveys for this topic. With the development of mobile Internet techniques and sharing economy, spatial crowdsourcing is attracting much attention recently [1], [5], [7], [8], [9], [19], [28], [29], [30]. In particular, the task assignment problem is a core issue in spatial crowdsourcing. [5], [6] propose the offline task assignment issue in spatial crowdsourcing, which aims to maximize the total number (or total utility) of assigned tasks. Recent studies focus on devising online algorithms to solve the dynamic task assignment problems in spatial crowdsourcing [12], [31], [32]. She et al. also address the conflict among different tasks in the task assignment problem in spatial crowdsourcing [33], [34], [35]. Moreover, different from crowdsourced data labelling and cleaning on the web [36], Hu et al. leverage spatial crowdsourcing to improve the quality of POI labelling [37]. Furthermore, [38] and [39] study the problem of protecting the location privacy of workers in spatial crowdsourcing. Although the aforementioned works study the task assignment problem in spatial crowdsourcing, they only consider two types of objects, i.e., tasks and workers.

In particular, a closely related research, called the online task assignment in spatial crowdsourcing, has been studied recently [12], [32]. They usually aim to maximize the number or the total utility of the assigned pairs of tasks and workers. There are two main differences between our work and [12], [32]. First, we study different problems. They only consider two types of objects, tasks and workers, but the TOM problem matches three types of dynamic objects. Thus, the solution space of the TOM problem is much larger than that of [12], [32]. Second, we adopt different strategies to solve the problems. For [12], the current decision can be guided by the offline optimal solution based on the current partial tripartite graph. However, the offline version of the TOM problem is already NP-hard problem. Thus, the optimal solution of the offline TOM problem is difficult to obtain in polynomial time and thus cannot be used to guide the online decision.

VII. Conclusion

In this paper, we formally define a novel dynamic task assignment problem, called the *trichromatic online matching in real-time spatial crowdsourcing* (TOM) problem. We first analyze our differences with existing task assignment studies in SC and prove that even the offline version of TOM is NP-hard. In order to solve this problem, we first devise a greedy algorithm, called Greedy, which has the competitive ratio of $\frac{1}{3U_{max}}$ but can be trapped into local optimal solutions easily. To obtain better approximation quality, a threshold-based framework is presented. Based on the framework, we first devise the Basic-Threshold algorithm, which has a better competitive ratio $\frac{1}{3e\lceil ln(U_{max}+1)\rceil}$ but has unstable approximation performance. In order to improve the stability of the Basic-Threshold algorithm, we further develop the Adaptive-Threshold algorithm, which can quickly seek optimal threshold for the Basic-Threshold algorithm. Finally, we verify the effectiveness, efficiency and scalability of the proposed methods through extensive experiments on both synthetic and real datasets.

Acknowledgment

We are grateful to anonymous reviewers for their constructive comments on this work. This work is supported in part by the National Science Foundation of China (NSFC) under Grant No. 61502021, 61232018, U1636210, National Grand Fundamental Research 973 Program of China under Grant 2014CB340300, the Hong Kong RGC Project N_HKUST637/13, NSFC Guang Dong Grant No.U1301253, the State Key Laboratory of Software Development Environment Open Program No. SKLSDE-2016ZX-13, the Scientific Innovation Act of STCSM No. 15JC1402400, and Microsoft Research Asia Gift Grant. Yongxin Tong is the corresponding author of this work.

References

[1] M. Mohamed and G. Deepak, "Labor dynamics in a mobile micro-task market," in *CHI 2013*.

[2] "Uber," *https://www.uber.com/*.

[3] "Gigwalk," *http://www.gigwalk.com*.

[4] "Grubhub," *https://www.grubhub.com/*.

[5] L. Kazemi and C. Shahabi, "Geocrowd: enabling query answering with spatial crowdsourcing," in *GIS 2012*.

[6] H. To, C. Shahabi, and L. Kazemi, "A server-assigned spatial crowdsourcing framework," *ACM Transactions on Spatial Algorithms and Systems*, 2015.

[7] L. Kazemi, C. Shahabi, and L. Chen, "Geotrucrowd: trustworthy query answering with spatial crowdsourcing," in *GIS 2013*.

[8] P. Cheng, X. Lian, Z. Chen, R. Fu, L. Chen, J. Han, and J. Zhao, "Reliable diversity-based spatial crowdsourcing by moving workers," *PVLDB*, 2015.

[9] P. Cheng, X. Lian, L. Chen, J. Han, and J. Zhao, "Task assignment on multi-skill oriented spatial crowdsourcing," *IEEE Transactions on Knowledge and Data Engineering*, 2016.

[10] "InterestingSport," *http://www.quyundong.com/*.

[11] "Nanguache," *http://www.nanguache.com/*.

[12] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen, "Online mobile micro-task allocation in spatial crowdsourcing," in *ICDE 2016*.

[13] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[14] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, "An optimal algorithm for on-line bipartite matching," in *STOC 1990*.

[15] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani, "Adwords and generalized online matching," *Journal of the ACM*, 2007.

[16] A. Mehta, "Online matching and ad allocation," *Theoretical Computer Science*, 2012.

[17] H. Ting and X. Xiang, "Near optimal algorithms for online maximum edge-weighted b-matching and two-sided vertex-weighted b-matching," *Theoretical Computer Science*, 2015.

[18] N. Nisan, T. Roughgarden, É. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*, 2007.

[19] Z. Chen, R. Fu, Z. Zhao, Z. Liu, L. Xia, L. Chen, P. Cheng, C. C. Cao, Y. Tong, and C. J. Zhang, "gMission: A general spatial crowdsourcing platform," *PVLDB*, 2014.

[20] R. E. Burkard, M. D. Amico, and S. Martello, *Assignment Problems*, 2009.

[21] R. C. Wong, Y. Tao, A. W. Fu, and X. Xiao, "On efficient spatial matching," in *VLDB 2007*.

[22] L. H. U, M. L. Yiu, K. Mouratidis, and N. Mamoulis, "Capacity constrained assignment in spatial databases," in *SIGMOD 2008*.

[23] V. Kann, "Maximum bounded 3-dimensional matching is max snp-complete," *Information Processing Letters*, 1991.

[24] E. M. Arkin and R. Hassin, "On local search for weighted k-set packing," *Mathematics of Operations Research*, 1998.

[25] G. Li, J. Wang, Y. Zheng, and M. J. Franklin, "Crowdsourced data management: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2016.

[26] A. I. Chittilappilly, L. Chen, and S. Amer-Yahia, "A survey of general-purpose crowdsourcing techniques," *IEEE Transactions on Knowledge and Data Engineering*, 2016.

[27] H. Garcia-Molina, M. Joglekar, A. Marcus, A. G. Parameswaran, and V. Verroios, "Challenges in data crowdsourcing," *IEEE Transactions on Knowledge and Data Engineering*, 2016.

[28] D. Deng, C. Shahabi, and L. Zhu, "Task matching and scheduling for multiple workers in spatial crowdsourcing," in *GIS 2015*.

[29] H. To, L. Fan, L. Tran, and C. Shahabi, "Real-time task assignment in hyperlocal spatial crowdsourcing under budget constraints," in *PerCom 2016*.

[30] D. Gao, Y. Tong, J. She, T. Song, L. Chen, and K. Xu, "Top-k team recommendation in spatial crowdsourcing," in *WAIM 2016*.

[31] Y. Tong, J. She, B. Ding, L. Chen, T. Wo, and K. Xu, "Online minimum matching in real-time spatial data: experiments and aalysis," *PVLDB*, 2016.

[32] U. U. Hassan and E. Curry, "A multi-armed bandit approach to online spatial task assignment," in *UIC 2014*.

[33] J. She, Y. Tong, L. Chen, and C. C. Cao, "Conflict-aware event-participant arrangement," in *ICDE 2015*.

[34] J. She, Y. Tong, and L. Chen, "Utility-aware social event-participant planning," in *SIGMOD 2015*.

[35] J. She, Y. Tong, L. Chen, and C. C. Cao, "Conflict-aware event-participant arrangement and its variant for online setting," *IEEE Transactions on Knowledge and Data Engineering*, 2016.

[36] Y. Tong, C. C. Cao, C. J. Zhang, Y. Li, and L. Chen, "Crowdcleaner: Data cleaning for multi-version data on the web via crowdsourcing," in *ICDE 2014*.

[37] H. Hu, Y. Zheng, Z. Bao, G. Li, J. Feng, and R. Cheng, "Crowdsourced POI labelling: Location-aware result inference and task assignment," in *ICDE 2016*.

[38] H. To, G. Ghinita, and C. Shahabi, "A framework for protecting worker location privacy in spatial crowdsourcing," *PVLDB*, 2014.

[39] L. Pournajaf, L. Xiong, V. Sunderam, and S. Goryczka, "Spatial task assignment for crowd sensing with cloaked locations," in *MDM 2014*.