IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 4, 2019, accepted September 25, 2019, date of publication October 4, 2019,
date of current version October 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2945585

Complex Dynamic Event Participant in an Event-Based Social Network: A Three-Dimensional Matching

YUAN LIANG
State Key Laboratory of Software Development Environment, School of Computer Science, Beihang University, Beijing 100191, China
e-mail: lianyuan120@buaa.edu.cn

ABSTRACT In recent years, event-based social networks (EBSNs), such as Meetup, Eventbrite and Douban, have emerged as a popular new type of social network on which online users can organize and register for offline social events. Existing approaches focus mainly on arrangement strategies that include users and events on an EBSN platform under an offline scenario, where all information is known in advance. However, these strategies ignore the importance of the organizer, and the offline scenarios can be impractical since all elements appear dynamically in reality. Therefore, we consider not only the user and event in the EBSN platform under the online scenario but also the importance of the organizer. In this paper, we study the complex dynamic event participant in an EBSN that considers three elements: users, events and organizers. We first formalize the event arrangement problem in the online scenario through event arrangement through three-dimensional matching (EATDM). Then, we propose a straightforward solution of the EATDM problem, a greedy algorithm with a competitive ratio, and further develop a basis solution, a random algorithm. We also propose a threshold-based algorithm and a weighted-threshold algorithm with a tighter competitive ratio. Finally, we verify the validity and practicability of the proposed algorithms on synthetic and real datasets, and we find that the weighted-threshold algorithm performs better than the random algorithm, the greedy algorithm and the threshold-based algorithm in terms of the total utility value.

INDEX TERMS Event-based social network, three-dimensional matching, event arrangement, competitive ratio.

I. INTRODUCTION

In recent years, event-based social networks (EBSNs) [1], such as Meetup,[1] Eventbrite [2] and Douban,[3] have emerged as a popular new type of online-to-offline social media; on EBSNs, *online* users can organize and register for *offline* social events. On the Meetup platform, which has attracted more than 18 million users with more than 320 thousand events held each month, users can create events that combine groups and organize events at a specific location. On the Plancast and Eventbrite[4] platforms, events in a community are organized, and users and organizers are recruited for the event [1]–[4].

Unfortunately, existing approaches consider only users and events, the two elements in EBSNs, and find a global arrangement strategy among the events and users to satisfy most users' interests [2], [5]–[8]. Reference [9] introduces the social event organization (SEO) problem, which assigns users to events such that users' satisfaction can be maximized, and the satisfaction includes the similarity between users and events and the social affinities of each user. Reference [10] considers that the location information and social friendship will influence the event arrangement, and they minimize the sum of the distance between users and events and the social cost, which is the weight of each partition. Reference [11] consider three influential factors, i.e., social friendship, similarity of attributes and location information, in the social event arrangement strategy. These studies did not consider the real-time aspects of users and events or the importance of the organizer.

In particular, in the EBSN platform, there are three elements: users, events and organizers. If the event is to be held

[1]https://www.meetup.com/
[2]https://www.eventbrite.com/
[3]https://www.douban.com/
[4]https://www.eventbrite.com/

The associate editor coordinating the review of this manuscript and approving it for publication was Zhan Bu.

successfully, the user must have a high degree of interest in the event, and the organizer must have a clear understanding of the whole process and related details of the event. In addition, the user's experience will affect the success of the next event. Therefore, to have a good user experience and to have a certain influence on the organizer, these three factors must be considered to play a role in activity arrangement.

In addition, [12] introduces the idea that event organizers are essential to the overall success of social events in online EBSNs, and organizers can attract a number of users who not only have the relevant expertise or skills required for an event but also consider the number of participants. For example, to organize a multidisciplinary conference on sustainability, we may want to choose a small group of program chairs (e.g., three or four) whose expertise covers the topics of the conference and who are able to influence the largest number of users in the community to contribute and participate in the conference. Obviously, selection of influential organizers is critical to the success of an event; thus, it is very necessary to consider the role of the organizer.

Therefore, in a real situation, users will consider the type of event, the venue and the organizer, and the organizer of the event will consider the users and the location of the event. These considerations will introduce new research topics to the area of event arrangement on EBSN platforms. For example, there is an outdoor sports event on the EBSN platform; Bob is the organizer of this event and has the right to choose where the event will take place; he also has the right to choose users. In addition, users have the right to choose in which events to participate. Therefore, when we assign users to activities, we should consider the role of three objects: users, events and organizers. These three objects need to be considered in real-world event arrangement tasks.

For such circumstances, studies can be conducted in the static offline setting, where all user and event information is known in advance. In fact, users, events and organizers have dynamic and real-time attributes. Users and organizers log onto the EBSN platform dynamically, and the invalid events are updated dynamically from time to time. Thus, the arrangement of the three elements of an EBSN platform should be conducted under dynamic online scenarios, in which users and events may appear anytime and anywhere. However, the three types of elements need to be matched, and previous studies cannot solve the current problem involving the three elements that need to be matched in the EBSN platform. Therefore, there are new technical challenges of this new application regarding *the modeling of the complex event arrangement task of the EBSN platform while considering the three elements and the development of a quality-assured and efficient algorithm for handling online scenarios*. For further elaboration, we provide a minor example as follows.

Example 1: In an EBSN platform, suppose that we have four users, $u_1 - -u_4$, three events, $e_1 - -e_3$, and three organizers, $o_1 - -o_3$, and that each user has an arrival time and leaving time, that each event has an arrival and leaving time, and that each organizer has an arrival and leaving time. There are also two restrictions: 1) the event organizer must arrive before the start of the event, and 2) the event organizer must leave after the end of the event. The arrival times and leaving times of users, events and organizers are shown in Table 1, and the arrival order is $u_1, u_2, o_1, e_1, o_2, u_3, e_2, o_3, u_4, e_3$. The capacities of u_1, u_2, u_3, u_4 are 2, 2, 2, and 3, and these values represent the maximum number of events in which users can participate. The capacities of e_1, e_2, e_3 are 3, 2, and 2, and these values represent the maximum numbers of users that an event can accommodate. Table 2 shows the utilities of all possible triples, and these values represent the satisfaction of the arrangement of users, events and organizers.

TABLE 1. Arrival time and leaving time of users, events and organizers.

Elements	Arrival Time	Leaving Time
u_1	8:00	10:00
u_2	8:10	15:00
o_1	8:20	10:30
e_1	8:30	10:00
o_2	12:30	15:30
u_3	13:00	15:00
e_2	13:00	15:00
o_3	13:45	16:10
u_4	13:55	16:02
e_3	14:00	16:00

TABLE 2. Utility of all possible triples.

Triples	Utility Value
(u_1, o_1, e_1)	28
(u_1, o_1, e_2)	10
(u_2, o_2, e_1)	20
(u_2, o_2, e_2)	38
(u_3, o_2, e_2)	50
(u_4, o_3, e_3)	76

Under the offline setting, the arrangement results are $\{(u_1, o_1, e_1), (u_2, o_2, e_2), (u_3, o_2, e_2), (u_4, o_3, e_3)\}$. Thus, the total utility of arrangement is 192. However, under the dynamic online scenario, when users exhibit dynamic arrival, the EBSN platform will immediately assign activities to them, and the decision cannot change once it is made. We can construct an arrangement of triples according to the arrival order; we can also conduct the arrangements according to $\{(u_1, o_1, e_1), (u_2, o_2, e_1), (u_3, o_2, e_2), (u_4, o_3, e_3)\}$, and the total utility is 184. We can see that the total utility of the offline setting is greater than that in the online dynamic scenario.

Based on this motivation, we formally define the event arrangement problem in the online scenario, called event arrangement through the three-dimensional matching (EATDM) problem, where users, events and organizers need to be arranged when they dynamically arrive at the platform one-by-one. When the decision can be made according to partial information, the arrangement cannot change. Previous work considered neither the preferences of organizers nor their roles in event arrangement. In summary, this is the first study of the EATDM problem considering the three elements

under the online scenario. Therefore, it is crucial to design an efficient and effective online algorithm for the EATDM problem. This paper has several contributions:

a) We formally define the event arrangement problem in the online scenario, called event arrangement through three-dimensional matching (EATDM), and it includes users, events, and organizers. We present that the EATDM problem is an NP-hard problem under the offline scenario, and we introduce an offline solution, i.e., local search.

b) We propose a straightforward solution of the EATDM problem: a greedy algorithm with a competitive ratio $\frac{1}{3F_{max}}$. Moreover, we develop a basis solution: a random algorithm with no competition. To filter the low utility value, we propose a threshold-based algorithm with a competitive ratio $\frac{p_{min}}{3e}$, where $p_{min} = \{p_0, p_1, \ldots, p_{\theta-1}\}$. Then, to improve the efficiency of the threshold-based algorithm, we develop a weighted-threshold algorithm with a tight competitive ratio that adaptively adjusts the probability distribution of choosing different thresholds.

c) We verify the validity and practicability of the proposed algorithms on synthetic and real datasets, and in terms of the total utility value, the weighted-threshold algorithm performs better than the random algorithm, the greedy algorithm and the threshold-based algorithm. Although the weighted-threshold algorithm requires a longer running time and more memory as it records all available thresholds, it processes a new arrival element in less than one millisecond and consumes less than 60 MB to process an element in most cases.

The paper is organized as follows. Section 2 overviews the related work. Section 3 states the problem in mathematical form. Section 4 proposes the algorithms of the EATDM problem under the offline and online settings, together with the theoretical analysis of several algorithms. Section 5 conducts experiments and analyzes the results. We conclude this paper in Section 6.

II. RELATED WORK

In this section, we summarize related work from two categories: event-based social networks (EBSNs) and three-dimensional matching.

A. EVENT-BASED SOCIAL NETWORK

In recent years, there have been some studies about EBSNs, such as [1], [4], [7], [10], [11], [13], [14]. Reference [1], which is the first study to analyze EBSNs, reports that, on the platform, offline organizers will post information about the event and users can use the platform to participate in offline activities. Reference [1] state that the distance between users and events influences the arrangement of users and events. Reference [15] focus on event recommendation on EBSNs by utilizing a learning model of EBSN data and recommending the right activities for the user. These works focus mainly on the benefits of one particular event and recommend events to users that satisfy most activities of organizers and users; however, these works do not consider that users dynamically arrive and do not consider the importance of the organizer.

There are some studies about arrangement strategies, for example [6], [8]–[11], [16]–[23]. Reference [9] maximize the sum of similarity and social friendship, and they mainly focus on the existence of two features that influence the user of the arranged event on the EBSN platform. Reference [10] studies a graph partition problem based on a game theoretic approach, which can arrange events for users. These works do not consider that users dynamically arrive, and their solutions are not suitable for users who dynamically arrive at the platform. Reference [8] consider several features that influence the arrangement of users to events, and this work does not consider the organizer. References [11], [20] studies the users' dynamic arrival setting. However, they do not consider the importance of the organizer on the EBSN platform. These issues require consideration of two elements, namely, users and events, and thus, these issues are different from the EATDM problem. In addition, the solution that includes two elements is not suitable for the EATDM problem, which considers three elements of the EBSN platform.

B. THREE-DIMENSIONAL MATCHING

In recent years, some studies on three-dimensional matching [24] have proven that the maximum three-dimensional matching problem is NP-hard, and [25] have proven that the three-dimensional matching problem is MAX SNP-hard. In the study of approximation algorithms for the three-dimensional matching problem, for any $\epsilon > 0$, [26], [27] prove that a solution is approximable within $\frac{2}{3} + \epsilon$, and [28] reports that a solution is approximable within $\frac{1}{2} - \epsilon$ for any $\epsilon > 0$. In these works, the solution of the unweighted three-dimensional matching problem and the maximum three-dimensional matching problem is local search, wherein an attempt is made to remove edges from the existing match and edges are added that do not conflict with the remaining match (i.e., no common points) to determine whether a better match can be obtained. These studies mainly consider the offline scenario, in which all information of the three elements is known in advance.

In addition, there are some studies about the three-dimensional matching problem under the online scenario. Reference [29] studied three-dimensional matching under two-element and one-element dynamic arrival. In our work, there are three elements, and the three elements all dynamically arrive on the EBSN platform. Reference [30] studies trichromatic online matching under the online scenario on the spatial crowdsourcing platform. We mainly consider an EBSN platform in a classical offline-to-online scenario where offline events do not need to be arranged immediately. Reference [31] studies three-dimensional matching of user, event and organizer; however, they mainly consider that each user can be arranged only in one triple, each event can only be arranged with one organizer, and each organizer can be arranged only with one user. In our work, we consider that the user, event and organizer have an associated capacity, which is the difference between our approach and that in [31].

III. PROBLEM STATEMENT

We first present the problem definition of the event arrangement through the three-dimensional matching (EATDM) problem and then provide the hardness of the offline scenario. Finally, we give the definition of the competitive ratio.

A. PROBLEM DEFINITION

This subsection first gives several definitions of users, events and organizers and then presents several concepts and gives the utility function of the EATDM problem under the online dynamic scenario.

Definition 1 (Users): A user is denoted by $u =< l_u, c_u, p_u, s_u, e_u >$, where l_u denotes the location of user u, c_u represents the capacity of user u, which denotes the maximum number of events in which user u can participate, p_u denotes the probability of attending an event, which can be obtained from historical data, and s_u and e_u denote the arrival time and leaving time for user u, respectively.

Definition 2 (Events): An event is denoted by $e =< l_e, c_e, s_e, e_e >$, where l_e represents the location of event e, c_e denotes the maximum number of users that can be accommodated in event e, and s_e and e_e denote the arrival time and leaving time for event e, respectively.

Definition 3 (Organizers): An organizer is denoted by $o =< l_o, u_o, c_o, s_o, e_o >$, where l_o represents the location of organizer o, u_o denotes the utility of organizer o for organizing an event, c_o denotes the maximum number of events that can be organized by organizer o, and s_o and e_o indicate the arrival time and leaving time for organizer o, respectively.

After defining the three elements of the EBSN platform, we introduce the EATDM problem under the online dynamic scenario and the utility function.

Definition 4 (Online EATDM): There are a set of users, a set of events, and a set of organizers, which have a location, starting time and leaving time. Upon the arrival of an element, the platform immediately arranges the element according to the other two elements, and once the arrangement is formed, it cannot be changed.

Definition 5 (Utility Function): Consider a set of users U, a set of events E, a set of organizers O, and a function $F(.,.,.)$ that denotes the utility of users, events and organizers in accordance with triples. The EATDM problem mainly finds an arrangement A among users, events and organizers to maximize the utility function $MaxSum(A) = \sum_{u\in U, o\in O, e\in E} F(u, o, e)$, which satisfies the following two constraints:

1) Online Constraint: once a triplet (u, o, e) of user t, organizer o and event e is arranged, the arrangement irreversible.

2) Time Constraint: The arrival time of the organizer must be before the arrival time of users and events, and the leaving time of the organizer must be after the leaving time of users and events. The arrival times of users are before the arrival times of events, and the leaving times of users are after the leaving times of events. In short, $s_o \le s_u \le s_e$ and $e_e \le e_u \le e_o$.

B. HARDNESS OF THE OFFLINE SCENARIO UNDER EATDM

In this section, we mainly study the hardness of the EATDM problem in the offline setting. In the offline setting, all user, event, and organizer information is known in advance. We study the hardness of the EATDM problem under the offline scenario in Theorem 1 as follows.

Theorem 1: The EATDM problem is NP-hard under the offline scenario.

Proof: We consider the special example of the EATDM problem such that the numbers of users, events and organizers are equal, i.e., the number of users $U = \{u_1, u_2, \ldots, u_n\}$, the number of events $E = \{e_1, e_2, \ldots, e_n\}$ and the number of organizers $O = \{o_1, o_2, \ldots, o_n\}$. Suppose the utility of an organizer is 1, the probability of users participating in events is 1, and the capacity of users, events and organizers is also 1. Note that the classical problem of three-dimensional matching is a well-known NP-complete problem [24]. The above example is a special case of three-dimensional matching, and we can reduce three-dimensional matching to the special case of the EATDM problem under the offline example. Therefore, the offline scenario of the EATDM problem is NP-hard.

However, in this paper, we mainly consider the online scenario of EATDM, in which the information of users, events and organizers is not known in advance, and one of the elements of the triplets dynamically arrives such that the solution to the EATDM problem under the online scenario becomes more challenging.

C. EVALUATION MODEL

In online algorithm research, the competitive ratio is usually used to measure the performance of the algorithm [32]–[34] by comparing the performance of the online algorithm to the performance of the offline optimal algorithm, where all information can be known in advance.

Definition 4 (Competitive Ratio): In the deterministic online algorithm, the competitive ratio is defined as

$$CR = \min_{\forall I(U,O,E,F)} \frac{A(I)}{OPT(I)}$$

where $I(U, O, E, F)$ denotes an arbitrary input of users, organizers, events and the utility function, and $A(I)$ denotes the total utility value conducted by the online deterministic algorithm. $OPT(I)$ denotes the optimal utility value of the event arrangement under the offline setting. In the EATDM problem, CR minimizes a ratio between the result the optimal result over all possible inputs and the result of online deterministic algorithm.

In the randomized online algorithm, each input I is sampled independently from a distribution D. Depending on the specific settings, the distribution may or may not be known. Under this model, the competitive ratio is defined as

$$CR = \min_{\forall I(U,O,E,F)} \frac{E[I]}{E[OPT(I)]}$$

where $E[OPT(I)]$ represents the expectation of the total utility value conducted by the randomized online algorithm.

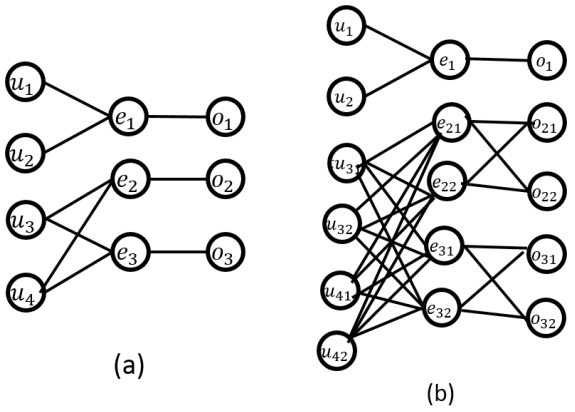


FIGURE 1. An example of an offline scenario under the EATDM problem.

IV. SOLUTION

In this section, we first introduce a solution under the offline scenario; then, we propose a straightforward solution, a basis algorithm and two algorithms of the EATDM problem under the online scenario.

A. OFFLINE ALGORITHM

Theorem 1 has proved that the EATDM problem under the offline scenario is NP-hard, and we first provide a special case such that the numbers of users, events and organizers are all equal to m. The goal of the EATDM problem is to maximize the utility function $F(u, o, e)$, and this function is divided into three parts: a paired user and event of an arrangement with a corresponding utility value $f(u, e)$, a paired user and organizer of an arrangement with a corresponding utility score $f(u, o)$, and a paired event and organizer of an arrangement with a corresponding utility score $f(e, o)$. Finally, the functions are defined as $F(u, o, e) = f(u, e) + f(u, o) + f(e, o)$ [35].

Then, we introduce a local search algorithm to solve the EATDM problem under the offline scenario. Local search mainly tries to remove edges from the existing arrangement and then join the edges (i.e., each edge has an associated utility score f) that satisfy the remaining arrangements (i.e., have no common nodes) and determine if a better arrangement can be obtained. To further illustrate this algorithm, we discuss an example.

Example 2: Recall Example 1, in which there are four users, $u_1 - -u_4$, three events, $e_1 - -e_3$, and three organizers, $o_1 - -o_3$. Suppose the capacity values of users are 1, 1, 2, and 2. The capacity values of events are 1, 2, and 2. The capacity values of organizers are 1, 2, and 2. If the capacity of an element is larger than 1, we can take the form of copying; Figure. 1 shows the capacity values of users, events or organizers that are larger than 1. Figure. 1(a) shows three-dimensional matching in accordance with Table 2, and we find that the utility values of (u_1, o_1, e_2) and (u_2, o_2, e_2) are 10 and 38, respectively. For event e_2, we use the form of copying by rewriting e_2 as e_{21} and e_{22}. After copying, Figure. 1(b) shows the result of Figure. 1(a).

The offline algorithm first adds the arrangement edges (i.e., arrange users to events, arrange organizers to events, arrange users to organizers) to the temporary result sets (i.e., Temp). In Figure. 1(a), the algorithm will use a top-down order to add edges into the temporary sets that satisfy all constraints. First, we add (u_1, o_1, e_1) to Temp, and then we consider (u_2, o_1, e_1); we find that it does not satisfy the capacity, and so (u_2, o_1, e_1) cannot be added into Temp. For each edge in the collection, try to delete the edge each time and add a new edge, leaving the new edge that satisfies the maximum utility score and the capacity constraints. Repeat this process, and the result will change according to a local search. In summary, the offline scenario needs to know all information in advance, and such a local search is not conducted in the online dynamic scenario.

B. GREEDY ALGORITHM

In this subsection, we propose a straightforward solution of the EATDM problem called the greedy algorithm.

The main idea of this algorithm is that a new element dynamically arrives, and the greedy algorithm finds the two other elements that construct triples that complete the user, event and organizer arrangement with the largest utility function $\max F(u, o, e)$. Note that if there are several arrangements that satisfy all constraints, then the greedy algorithm will choose the largest utility value that includes the user, event and organizer. Furthermore, the greedy algorithm will obtain an arrangement whenever possible, even if the utility value of that arrangement is very low.

Algorithm 1 Greedy Algorithm

input: $U, O, E, F(.,.,.)$
output: A feasible arrangement A

```
1: for new arrival user u do
2:     S ← {∀e|e is a triple containing user u and organizer
   o and satisfying all constraints};
3:     if S ≠ ∅ then
4:         a ← the element in S with the largest utility
   value;
5:         A ← A ∪ {a};
6:     end if
7: end for
8: return the final arrangement A.
```

Algorithm 1 illustrates the procedure. In lines 1–2, when a new element arrives, the greedy algorithm finds a triple containing the other two elements that satisfy all constraints and then puts triples into S. In lines 3–8, if S is the $\emptyset$, then the greedy algorithm continues; if $S \neq \emptyset$, the element in S with the largest utility value is selected and added into the arrangement. In line 10, the greedy algorithm returns the final arrangement A.

Example 3: Recall Example 1. We obtain the arrival time and leaving of users, events and organizers according to Table. 1. When user u_1 arrives, $S = \emptyset$. After $u_2, o_1, and e_1$

arrive, the greedy algorithm arranges the triple (u_1, o_1, e_1) and puts this triple into S. After o_2 $and e_2$ arrive, the greedy algorithm arranges (u_2, o_2, e_2) with utility value 38. Finally, other elements arrive, and the greedy algorithm returns an arrangement such that $(u_1, o_1, e_1), (u_2, o_2, e_2), (u_4, o_3, e_3)$. Thus, the total utility score is $28 + 38 + 76 = 142$.

1) COMPETITIVE RATIO

Then, we analyze the competitive ratio of the greedy algorithm.

Lemma 1: Let A_{OPT} and A_{Greedy} denote the arrangement results of the optimal solution under the offline scenario and the greedy algorithm. We have $|A_{Greedy}| \geq \frac{|A_{OPT}|}{3}$, where $|A_{Greedy}|$ and $|A_{OPT}|$ are the number of triples of the offline optimal result and the greedy algorithm result, respectively.

Proof: For each triple a in A_{OPT}, which includes three arrangements (i.e., arrangement of users to an event, arrangement of organizers to events and arrangement of users to organizers), and there is at least one of the arrangements arranged by the greedy algorithm; thus, we have $|A_{Greedy}| \geq \frac{|A_{OPT}|}{3}$.

Theorem 1: Suppose that the utility value of each triple is $[1, F_{max}]$ and that the competitive ratio of the greedy algorithm is $\frac{1}{3F_{max}}$.

Proof: Suppose the utility value of each arrangement of a triple is $[1, F_{max}]$, and we have

$$\begin{aligned} MaxSum(A_{Greedy}) &\geq |A_{Greedy}| \cdot 1 \geq \frac{|A_{OPT}|}{3} \cdot 1 \\ &\geq \frac{1}{3F_{max}} |A_{OPT}| \cdot F_{max} \\ &\geq \frac{MaxSum(A_{OPT})}{3F_{max}} \end{aligned} \tag{1}$$

Thus, the competitive ratio of the greedy algorithm is

$$CR = \frac{MaxSum(A_{greedy})}{MaxSum(A_{OPT})} = \frac{1}{3F_{max}}.$$

2) COMPLEXITY ANALYSIS

There are new arrival users U, events E and organizers O, and the time complexity and space complexity of the greedy algorithm are both $O(\max(|U||E|, |U||O|, |E||O|))$.

C. RANDOM ALGORITHM

In this section, we propose a basis solution to the EATDM problem called the random algorithm.

The main idea of the random algorithm is that, when a new element dynamically arrives, the random algorithm will select the two other elements to be matched randomly to satisfy all constraints. Additional details of the random algorithm are shown in Algorithm 2.

Algorithm 2 illustrates the random procedure. The input to the random algorithm is the arrival users U, events E, organizers O and a utility function $F(.,.,.)$, and the output of the random algorithm is a feasible arrangement A. In lines 1–2, when a new element arrives, the random algorithm finds a triple containing the other two elements that satisfy all constraints and then puts triples into S. In lines 3–6, if $S \neq \emptyset$, select the element in S randomly and add the element into the arrangement. In line 8, the random algorithm returns the final arrangement A.

Algorithm 2 Random Algorithm

input: $U, O, E, F(.,.,.)$
output: A feasible arrangement A
1: **for** new arrival element v **do**
2: $S \leftarrow \{\forall v | v$ in a triple including element v and satisfying all constraints$\}$;
3: **if** $S \neq \emptyset$ **then**
4: $a \leftarrow$ Select an arrangement from S randomly;
5: $A \leftarrow A \cup \{a\}$;
6: **end if**
7: **end for**
8: **return** the final arrangement A.

Example 4: Recall Example 1. We obtain the arrival time and leaving time of users, events and organizers according to Table. 1, where the arrival order is $u_1, u_2, o_1, e_1, o_2, u_3, e_2, o_3, u_4, e_3$. Let the capacities of $u_1, u_2, u_3, and u_4$ be 2, 2, 2, and 3; the capacities of $e_1, e_2, and e_3$ be 3, 2, and 2; and the capacities of $o_1, o_2, and o_3$ be 1, 1, and 1. The first arrival of the four elements becomes a triple (u_1, o_1, e_1), and the random algorithm will arrange user u_1 to event e_1 and organizer $o_!$ and obtain a utility value of 28. As such, the capacity of organizer o_1 is 0. Next, $o_2, u_3, and e_2$ arrive, and the random algorithm randomly arranges (u_2, o_2, e_2) as a triple; the utility value is 38. After $o_3, u_4, and e_3$ arrive, the random algorithm randomly arranges (u_3, o_2, e_2) and satisfies all constraints. Finally, the total utility value is $28 + 38 + 50 = 108$. From this example, the random algorithm obtains a lower total utility since, when the lower-utility arrangement conflicts with the higher-utility arrangement, the relevant elements of the lower-efficiency arrangement will arrive first, such that the random algorithm will lose the most efficient arrangement.

Complexity Analysis: There are new arrival users U, events E and organizers O, and the time complexity and space complexity of the random algorithm are both $O(\max(|U||E|, |U||O|, |E||O|))$.

D. THRESHOLD-BASED ALGORITHM

The random algorithm will obtain a lower total utility since, when the lower-utility arrangement conflicts with the higher-utility arrangement, such that when the relevant elements of the lower-efficiency arrangement arrive first, the random algorithm will lose the most efficient arrangement. To solve this problem, we extend the greedy-RT algorithm [36] and develop an online algorithm called the threshold-based algorithm. The threshold algorithm is used only for tasks with two elements and allows only one type of element to arrive dynamically. In this section, we extend the threshold-based

algorithm to the pattern in which three types of elements arrive dynamically.

The main idea of the threshold-based algorithm is that, when first selecting a utility threshold when the platform makes an interactive arrangement, the threshold-based algorithm chooses only the arrangement in which the utility is greater than the threshold. Although this strategy may lose some less-effective arrangements, the larger utility can be selected in conflicting arrangements to avoid the worst case. For the threshold-based algorithm, when an element v arrives, if the capacity of the newly arrived element c_v is larger than 1, let c_v be multiple elements with a capacity of 1. Additional details are shown in Algorithm 3.

Algorithm 3 Threshold-Based Algorithm

input: $U, O, E, F(.,.,.)$
output: A feasible arrangement A
1: $\theta \leftarrow \lceil \ln(F_{max}+1) \rceil$
2: Taking an integer from $k \in \{0, 1, \ldots, \theta-1\}$ as the threshold with probability $p_0, p_1, \ldots, p_{\theta-1}$
3: **for** new arrival element v **do**
4: $S \leftarrow \{\forall v | v$ in a triple including element v and satisfying all constraints, and $F(.,.,.) \geq e^k\}$;
5: **if** $S \neq \emptyset$ **then**
6: $a \leftarrow$ Select a triple from S arbitrarily;
7: $A \leftarrow A \cup \{a\}$;
8: **end if**
9: **end for**
10: **return** the final arrangement A.

Algorithm 3 illustrates the threshold-based algorithm procedure. The input to the threshold-based algorithm is the arrival of users U, events E, organizers O and a utility function $F(.,.,.)$; the output of the threshold-based algorithm is a feasible arrangement A. In lines 1–2, the threshold-based algorithm chooses an integer from $\{0, 2, \ldots, \theta-1\}$ as the threshold with probability $p_0, p_1, \ldots, p_{\theta-1}$ and sets $\theta \leftarrow \lceil \ln(F_{max}+1) \rceil$, where F_{max} is a maximum utility value according to historical data. In lines 3–4, each new arrival element v is put into a triple that includes element v that satisfies all constraints; the utility value of this triple is computed by $F(.,.,.)$ and is no less than e^k. In lines 5–10, if S is the empty set, then the threshold-based algorithm continues; if S is not the empty set, then the threshold-based algorithm selects a triple from S arbitrarily and puts this triple into A. Finally, the threshold-based algorithm returns a total feasible arrangement A.

Example 5: Recall Example 1. We obtain the arrival time and leaving time of users, events and organizers according to Table. 1, where the maximum utility value is 76, $\theta = \lceil \ln(F_{max}+1) \rceil = 4$, and let k obtain a value from $\{0, 1, 2, 3\}$. Suppose $k = 3$; the threshold is approximately 20. Suppose that the elements' arrival order is $u_1, u_2, o_1, e_1, o_2, u_3, e_2, o_3, u_4, e_3$ and that the triple $(u_1, o_1, e_1) > 20$; therefore, the threshold-based algorithm arranges a triple that is (u_1, o_1, e_1). After the other elements arrive, each time a threshold is judged and $(u_1, o_1, e_2) < 20$, the threshold-based algorithm will not arrange these elements. Then, the threshold-based algorithm will put (u_1, o_1, e_1) into S. The final arrangement of the threshold-based algorithm is $(u_1, o_1, e_1), (u_3, o_2, e_2), (u_4, o_3, e_3)$. Thus, when $k = 4$, the final utility value is 154. This algorithm is a random algorithm for choosing k, and the expectation of the total utility value for all possible k is $\frac{154+142+114+76}{4} = 121.5$.

1) COMPETITIVE RATIO ANALYSIS

Then, we analyze the competitive ratio of the threshold-based algorithm.

Theorem 2: The competitive ratio of the threshold-based algorithm is $\frac{p_{min}}{3e}$, where $p_{min} = \{p_0, p_1, \ldots, p_{\theta-1}\}$.

Proof: Suppose that G is the optimal solution under the offline scenario and that $G_{[e^i, e^{i+1})}$ is a subgraph of G that contains only edges whose utility values lie in $[e^i, e^{i+1})$. Let $OPT_{[e^i, e^{i+1})}$ be the optimal solution in $G_{[e^i, e^{i+1})}$, and $A_{[e^i, e^{i+1})}$ is the arrangement result of the threshold-based algorithm in $G_{[e^i, e^{i+1})}$. According to Lemma 1, we have $|A_{[e^i, e^{i+1})}| \geq \frac{|OPT_{[e^i, e^{i+1})}|}{3}$; thus,

$$\begin{aligned}
E[MaxSum(A)] &= \sum_{i=0}^{\theta-1} p_i MaxSum(A_{[e^i,\infty)}) \\
&\geq p_{min} \sum_{i=0}^{\theta-1} MaxSum(A_{[e^i, e^{i+1})}) \\
&\geq p_{min} \sum_{i=0}^{\theta-1} e^i |A_{[e^i, e^{i+1})}| \\
&\geq p_{min} \sum_{i=0}^{\theta-1} e^i \frac{|OPT_{[e^i, e^{i+1})}|}{3} \\
&\geq \frac{p_{min}}{3} \sum_{i=0}^{\theta-1} MaxSum(OPT_{[e^i, e^{i+1})}) \\
&\geq \frac{p_{min}}{3e} MaxSum(OPT). \qquad (2)
\end{aligned}$$

In short, the competitive ratio of the threshold-based algorithm is $\frac{p_{min}}{3e}$, where $p_{min} = \{p_0, p_1, \ldots, p_{\theta-1}\}$.

2) COMPLEXITY ANALYSIS

There are new arrival users U, events E and organizers O, and the time complexity and space complexity of the threshold-based algorithm are both $O(\max(|U||E|, |U||O|, |E||O|))$.

E. WEIGHTED-THRESHOLD ALGORITHM

The threshold-based algorithm is better than the random algorithm in terms of utility value since the threshold-based algorithm can choose the larger utility value from the conflict arrangement. However, the shortcoming of the threshold-based algorithm is that different thresholds are selected, and the total utility obtained by the threshold-based algorithm is greatly affected by k. Therefore, we introduce a

weighted-threshold algorithm that is combined with the randomized weighted majority, which is used in the metrical task system (MTS) problem [37].

The weighted-threshold algorithm allows dynamic adjustment in choosing k with probability p_k, increases the probability of k for which a better effect is selected, and thus improves the performance of the algorithm. The weighted-threshold algorithm first sets the weight of $0, 1, .., \theta-1$ as w_i; additionally, $W = \sum_i w_i$, $p_i = \frac{w_i}{W}$. If no element arrives, k is randomly selected from $0, 1, .., \theta-1$. More details are shown in Algorithm 4.

Algorithm 4 Weighted-Threshold Algorithm

input: $U, O, E, F(.,.,.)$
output: A feasible arrangement A
1: $\theta \leftarrow \lceil \ln(F_{max}+1) \rceil$;
2: $w_i = 1$ for $i = 0, 1, 2, \ldots, \theta-1$;
3: Set $p_i = \frac{w_i}{W}$ for $i = 0, 1, 2, \ldots, \theta-1$ and $W = \sum_{i=0}^{\theta-1} w_i$;
4: **for** new arrival element v **do**
5: $k \leftarrow$ Taking an integer from $\{0, 1, \ldots, \theta-1\}$ with probability $p_0, p_1, \ldots, p_{\theta-1}$
6: $S \leftarrow \{\forall v | v$ in a triple including element v and satisfying all constraints, and $F(.,.,.) \geq e^k\}$;
7: **if** $S = \emptyset$ **then**
8: continue;
9: **else**
10: $a \leftarrow$ Select a triple from S arbitrarily;
11: $A \leftarrow A \cup \{a\}$;
12: **end if**
13: Suppose $w_k = w_k(1+\beta)^{\frac{F_k}{F_{max}}}$, where $\beta \in (0,1)$, and F_k denotes that the utility of the threshold-based algorithm lies in the threshold e^k;
14: Update W, p_i;
15: **end for**
16: **return** the final arrangement A.

Algorithm 4 illustrates the weighted-threshold algorithm procedure. The input to the weighted-threshold algorithm is the arrival of users U, events E, organizers O and a utility function $F(.,.,.)$, and the output of the weighted-threshold algorithm is a feasible arrangement A. In lines 1–3, set $\theta \leftarrow \lceil \ln(F_{max}+1) \rceil$, let $w_i = 1$ for $i = 0, 1, 2, \ldots, \theta-1$, and set $p_i = \frac{w_i}{W}$ for $i = 0, 1, 2, \ldots, \theta-1$ and $W = \sum_{i=0}^{\theta-1} w_i$. In lines 4–15, for new arrival element v, k is a value from $\{0, 1, \ldots, \theta-1\}$ with probability $p_0, p_1, \ldots, p_{\theta-1}$. In line 6, the weighted-threshold algorithm will put new arrival element v into a triple that includes element v and satisfies all constraints, the utility value of this triple is computed by $F(.,.,.)$, which is no less than e^k, and these are a set of triples S. In lines 7–12, if S is empty, then the weighted-threshold algorithm will continue; if S is not empty, then the weighted-threshold algorithm will select a triple from S arbitrarily and put this triple into A. In lines 13–14, suppose $w_k = w_k(1+\beta)^{\frac{F_k}{F_{max}}}$, where $\beta \in (0,1)$, and F_k denotes that the utility of the threshold-based algorithm lies in the threshold e^k; then, update W, p_i. The weighted-threshold algorithm returns the final arrangement A.

Example 6: Recall Example 1. We obtain the arrival time and leaving time of users, events and organizers according to Table. 1. In this example, we use $\beta = 0.001$ to update the w_i. In the weighted-threshold algorithm, let $\theta = \lceil \ln(78+1) \rceil = 4$, and $k = \{0, 1, 2, 3\}$, $w_i = 1$, $p_i = \frac{1}{4}$. When u_1 arrives, we use $k = 0, 1, 2$, and we obtain the utility of 28 for triple (u_1, o_1, e_1) and update the weight from 1 to 1.02. Then, o_2 arrives, and we use $k = 3$ at the start, and the utility 20 will be gained, whereby the weights will be updated to 1.04. After u_3 arrives, we use $k = 2, 3$ at the start, and the utility 50 be gained, such that the weights e^2 *and*e^3 will be updated to 1.22 and 1.17, respectively. Next, a utility of 76 is gained. We observe that e^3 has the largest utility, which means that, if there are other elements that arrive, e^3 is the largest probability and is chosen since threshold e^3 performs the best on the elements dynamically arriving.

1) COMPETITIVE RATIO ANALYSIS

Then, we analyze the competitive ratio of the weighted-threshold algorithm.

Theorem 3: For any $\epsilon > 0$, the expected utility of the weighted-threshold algorithm is at least $(1-\epsilon)MaxSum(OPT)$, where $MaxSum(OPT)$ is the optimal result under the offline scenario.

Proof: Suppose the weighted-threshold algorithm currently has weights $w_0, w_1, \ldots, w_{\theta-1}$; utility $F_0, F_1, \ldots, F_{\theta-1}$ is obtained with different values of k, and after updating weights $w'_0, w'_1, \ldots, w'_{\theta-1}$, where $w_k = w_k(1+\beta)^{\frac{f_k}{F_{max}}}$, $\beta \in (0,1)$, f_k denotes that the utility of the threshold-based algorithm lies in the threshold e^k.

Set $W = \sum_i w_i$, $W' = \sum_i w'_i$, $p_i = \frac{w_i}{W}$, $p'_i = \frac{w'_i}{W'}$. When element v arrives, the weighted-threshold algorithm arranges element v so that it will obtain the expected utility of $E_v = \sum_i p_i F_i$, and the loss for the probability distribution p_i to the probability distribution p'_i is at most $\sum_{i:p'>p_i}(p'_i - p_i) \leq \frac{\sum_{i:p'>p_i}(w'_i - w_i)}{W} \leq \frac{W'}{W} - 1$.

For any $\alpha \leq 1$, $(1+x))^\alpha \leq 1+\alpha x$, we have

$$W' \leq \sum_i w_i(1+\frac{\beta F_i}{F_{max}}) = W\sum_i p_i(1+\frac{\beta F_i}{F_{max}}) = W(1+\frac{\beta E_v}{F_{max}}).$$

Considering $(\frac{W'}{W} - 1) \leq \frac{\beta E_v}{F_{max}}$, the weighted-threshold algorithm will obtain utility $E_v(1-\frac{\beta}{F_{max}})$ from element v. Suppose W_f is the weight at which the weighted-threshold algorithm stops, and we have $(1_\beta)^{\frac{OPT}{F_{max}}} \leq W_f \geq \prod_v (1+\frac{\beta E_v}{F_{max}})$.

For any $x \in [0,1]$, $\ln(1+x) \in [\frac{x-x^2}{2}, x]$, we have

$$\frac{OPT}{F_{max}}(\beta - \frac{\beta^2}{2}) \leq \frac{OPT(\ln(1+\beta))}{F_{max}} \leq \sum_v \ln(1+\frac{\beta E_v}{F_{max}}) \leq \frac{\beta}{F_{max}}\sum_v E_v.$$

Therefore, $\sum_v E_v \geq (1-\frac{\beta}{2})OPT$, and the total utility is $F \geq (1-\frac{\beta}{2})(1-\frac{\epsilon}{2}) \geq (1-\epsilon)OPT$.

2) COMPLEXITY ANALYSIS

There are new arrivals of users U, events E and organizers O, and the time complexity and space complexity of the weighted-threshold algorithm are both $O(\max(\lceil(F_{max}+1)\rceil|U||E|, \lceil(F_{max}+1)\rceil|U||O|, \lceil(F_{max}+1)\rceil|E||O|))$.

V. EXPERIMENTAL EVALUATION

In this section, we first introduce the experimental environment, where real datasets from Meetup and synthetic datasets are used to test the scalability of the greedy algorithm, random algorithm, threshold-based algorithm and weighted-threshold algorithm. Then, the experimental results are analyzed. Finally, we summarize the results of the experiment.

A. EXPERIMENTAL ENVIRONMENT AND DATASETS

1) EXPERIMENT ENVIRONMENT

All experiments are performed using the Linux Ubuntu operating system. The experiments are conducted on a computer with an Intel Xeon E5620 with a 2.40 GHz 16-core CPU and 16 GB of memory, and all algorithms are written in C++.

2) REAL DATASETS

We used the Meetup dataset as the real dataset [1], [13], [38]. Each user is associated with a set of tags, a location, a release time and a deadline time, where the tag denotes the interest of a user, the location records the latitude and longitude of a user, the release time represents a user's arriving time and the deadline time represents a user's leaving time. In addition, there is a set of groups in the EBSN platform, and we see each group as an event, where each event is associated with a set of tags, location, arriving time and leaving time. Several users of each group are considered as organizers, and each organizer has a tag, location, leaving time and arriving time. We consider N_g as the number of users in a group, N_o as the number of organizers in a group, and we have $N_g \leq N_o$. The leaving time minus the arriving time is the waiting time, $w_t = s_t - e_t$, and we use waiting to denote the effective time of a triple. During the experiment, by simply using the tags of users, groups and the organizers of the group, we can calculate the utility of a triple (i.e., user, event and organizer). The capacities of users are generated by the normal distribution, where the mean of c_u is 3, the capacities of events are generated by the normal distribution, with the mean of c_e at 50, and the capacities of organizers are generated by the normal distribution, where the mean of c_o is 2. The parameters of the real dataset are summarized in Table 3.

TABLE 3. Real dataset.

City	$\|U\|$	$\|E\|$	$\|O\|$	Mean of c_u	Mean of c_e	Mean of c_o
Vancouver	2012	225	250	3	50	2
Singapore	1500	87	100	3	50	2
Auckland	569	37	50	3	50	2
Beijing	113	16	20	3	50	2

TABLE 4. Synthetic dataset.

Factor	Setting
$\|E\|$	100, 300, **500**, 700, 1000
$\|U\|$	1000, 3000, **5000**, 7000, 10000
$\|O\|$	120, 360, **600**, 840, 1200
c_u, c_o, c_e	10 to 200 with 10 as a step

3) SYNTHETIC DATASETS

To test the scalability of our algorithms, we use some synthetic datasets. Synthetic datasets are generated by the locations of users, events and organizers by following the normal distribution, and the maximum utility score F_{max} is set to 100. The arriving time and leaving time are generated by the normal distribution, the capacity of a triple is set from 10 to 200 with 10 as a step. Table 4 shows the statistics and configuration of synthetic datasets, and default settings are marked in bold.

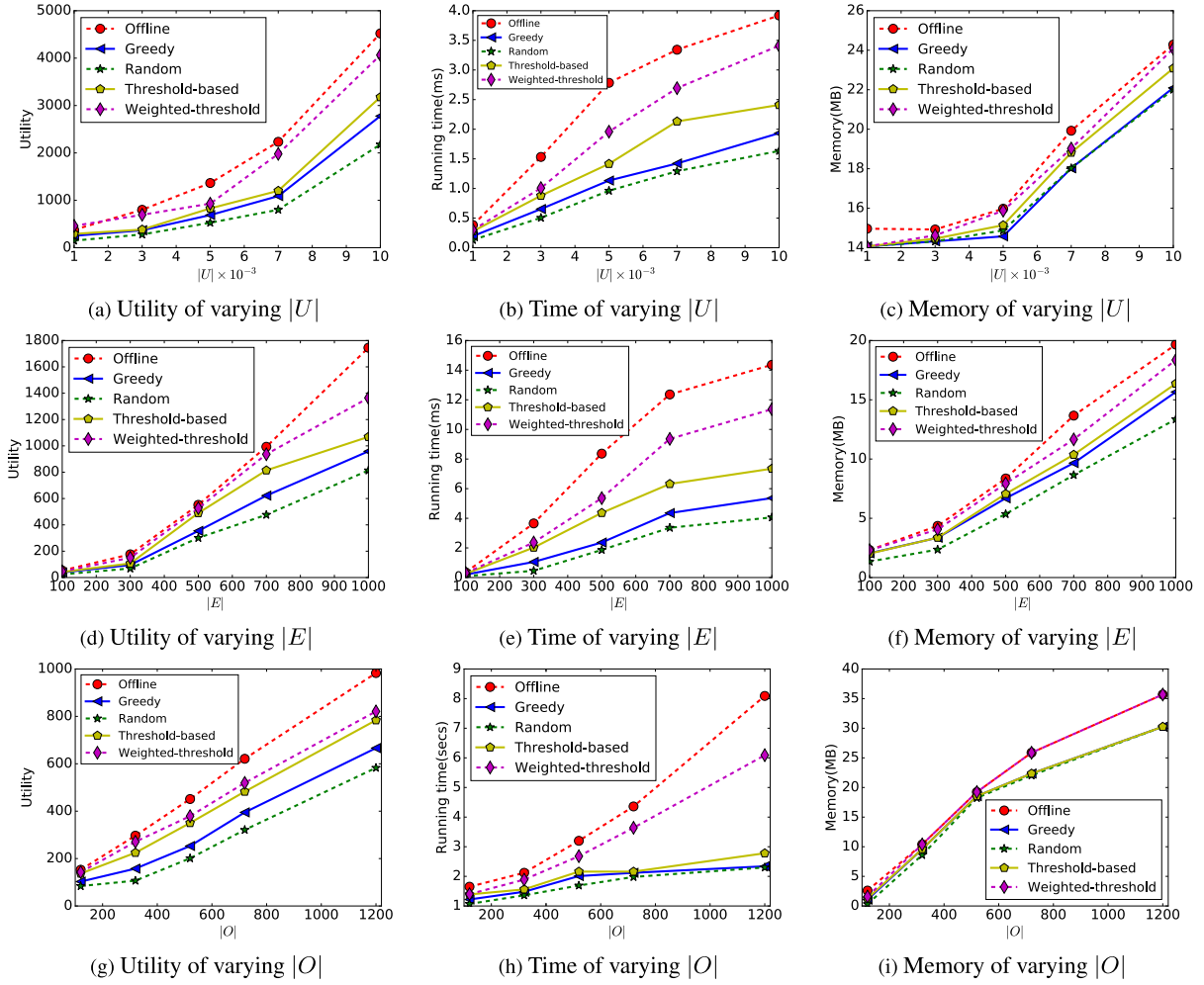
In this experiment, we compare the following algorithms: the optimal solution, i.e., the offline algorithm; a straightforward solution, i.e., the greedy algorithm; a basis solution, i.e., the random algorithm; the threshold-based algorithm; and the weighted-threshold algorithm. We evaluate all proposed algorithms in terms of total utility, running time and memory cost and study the effects of varying the parameters on the performance of the proposed algorithms.

B. EXPERIMENTAL RESULTS

In this subsection, we first evaluate the offline, greedy, random, threshold-based, and weighted-threshold algorithms in terms of utility value, memory usage and running time. Then, we study the effects of the following parameters: the size of $|U|$, the size of $|E|$, the size of $|O|$, the capacity of users c_u, the capacity of events c_e, the capacity of organizers c_o, and waiting time of each element w_t.

1) EFFECT OF $|U|$

We first study the effects of varying $|U|$, where $|U|$ is set to $\{1000, 3000, 5000, 7000, 10000\}$, and the number of events is 500, the number of organizers is 600, and the waiting time of each element is 50 minutes. The capacities of users are generated following the normal distribution $N[3, 5]$. The capacities of events are generated following the normal distribution $N[50, 75]$. The capacities of organizers are generated following the normal distribution $N[2, 5]$. The experimental results in terms of the total utility value, running time and memory cost are shown in Figs. 2a to 2b. We report the results as follows. First, the weighted-threshold algorithm performs better than the threshold-based algorithm, greedy algorithm and random algorithm in terms of utility, since the weighted-threshold algorithm can provide a dynamic adjustment for choosing k with probability p_k and increase the probability of k for which a better effect is selected. Second, when $|U|$ increases, the total utility value, memory usage and running time are increased because, when a new element arrives,



(a) Utility of varying $|U|$ (b) Time of varying $|U|$ (c) Memory of varying $|U|$

(d) Utility of varying $|E|$ (e) Time of varying $|E|$ (f) Memory of varying $|E|$

(g) Utility of varying $|O|$ (h) Time of varying $|O|$ (i) Memory of varying $|O|$

FIGURE 2. Results of varying $|U|$, $|E|$, and $|O|$.

more triples can be arranged. Third, the memory and running time increase as $|U|$ increases, and the weighted-threshold, compared to other algorithms, will consume more memory and more time, since the weighted-threshold algorithm has to record all thresholds to choose the best threshold.

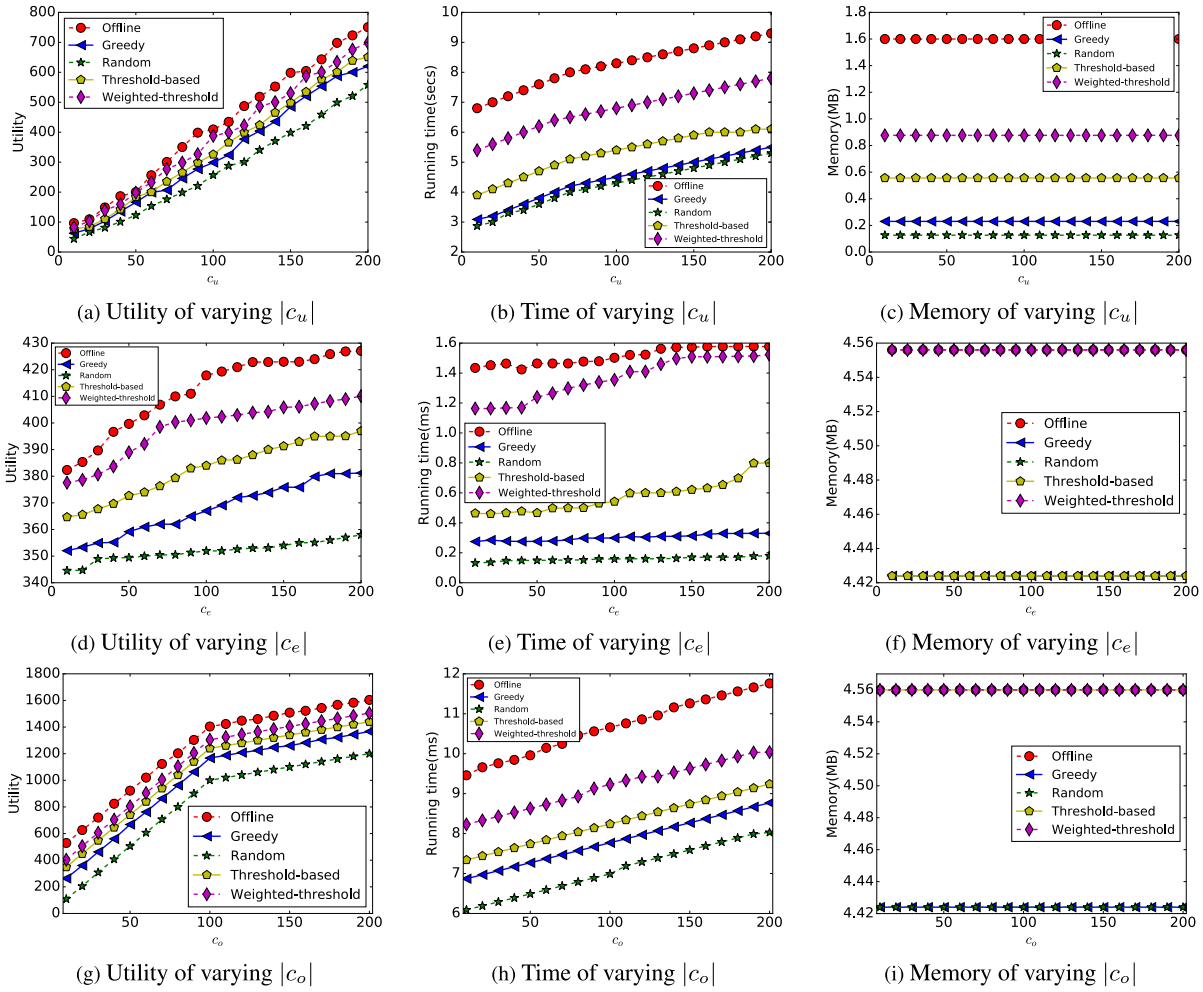
2) EFFECT OF $|E|$

Next, we study the effect of varying the number of events as {100, 300, 500, 700, 1000 }, where the number of organizers is set to 600, and the number of users is set to 5000. Capacity and waiting time settings are shown by default numbers. The experiments of all proposed algorithms are shown in Figs. 2d to 2f. We report the following observations. First, the total utility value increases when $|E|$ increases because more triples will be arranged when new elements arrive. Second, the memory usage and the running time increase as the number of events $|E|$ increases, which is natural, as there are more events that can be arranged for users and organizers when E increases. Third, we observe that the weighted-threshold algorithm performs better than other algorithms under the online scenario, and this algorithm consumes more running time and memory usage. However, we find that the weighted-threshold algorithm is still efficient, and it can process an element that dynamically arrives in less than one millisecond.

3) EFFECT OF $|O|$

Next, we present the results of setting the number organizers to {120, 360, 600, 840, 1200}, while the other parameters are shown by default numbers. Figs. 2g to 2i present the result of varying the number of organizers in terms of total utility, running time and memory usage. We report several observations as follows. First, the total utility, running time and memory usage increase when the size of organizers $|O|$ increases; it is natural as there are more triples that can be arranged. Second, we can find that the weighted-threshold algorithm performs better than the greedy algorithm, random algorithm and threshold-based algorithm. Third, the running time and memory usage are mainly related to the scale of organizers.

Next, we conduct experiments by varying the capacity of users, events and organizers, where c_u, c_o, c_e are set from 10 to 200 with 10 as a step, while the number of users is fixed at 5000, the number of events is fixed at 500, and the number of organizers is fixed at 600. Moreover, for the



(a) Utility of varying $|c_u|$ (b) Time of varying $|c_u|$ (c) Memory of varying $|c_u|$

(d) Utility of varying $|c_e|$ (e) Time of varying $|c_e|$ (f) Memory of varying $|c_e|$

(g) Utility of varying $|c_o|$ (h) Time of varying $|c_o|$ (i) Memory of varying $|c_o|$

FIGURE 3. Results of varying $|c_u|$, $|c_e|$, and $|c_o|$.

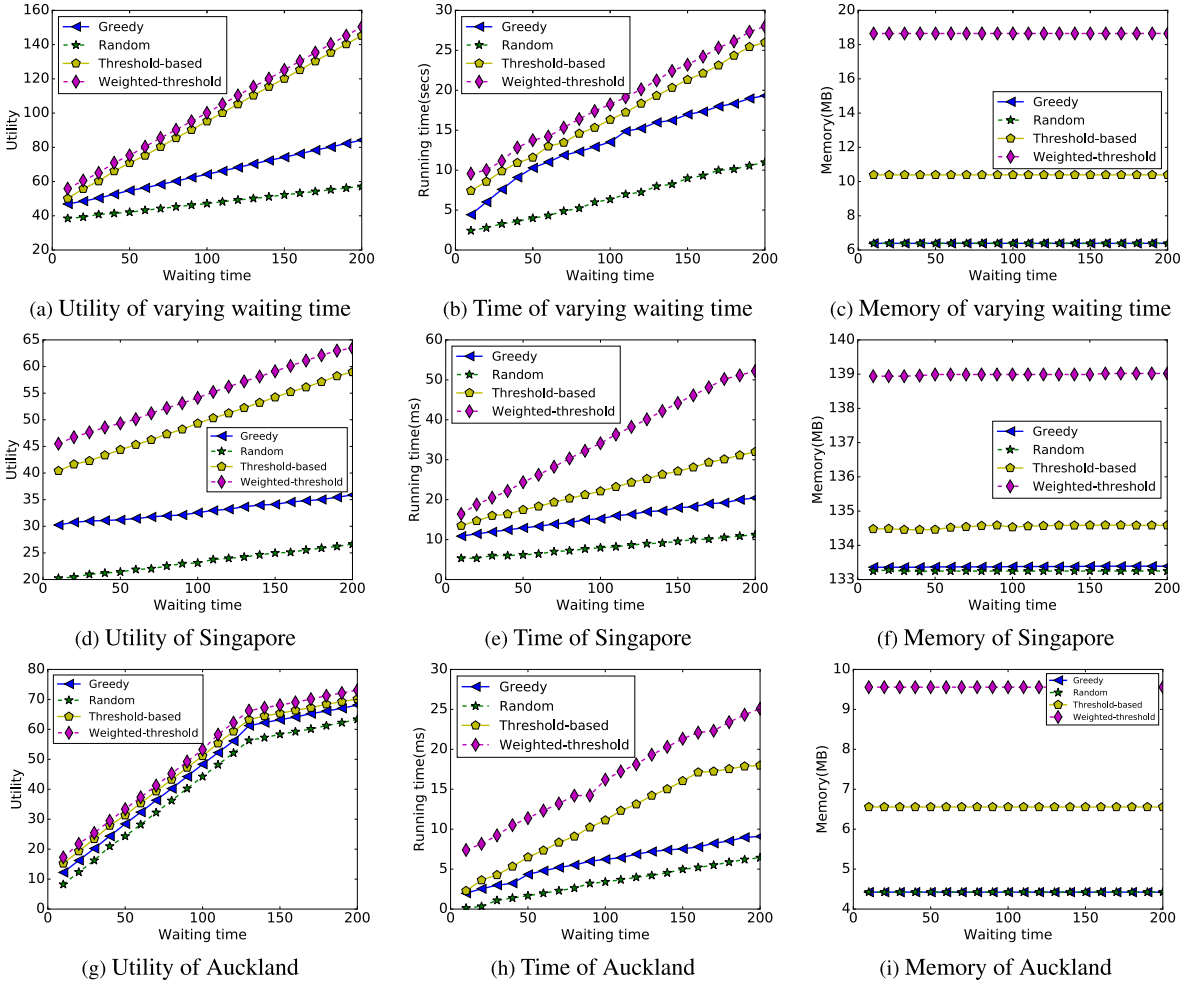
random algorithm, threshold-based algorithm and weighted-threshold algorithm, together with the settings of each parameter, these algorithms are conducted for 100 rounds, and the average results of these 100 rounds are evaluated. The greedy algorithm is a deterministic algorithm; this algorithm can obtain only one result when all parameters are the same.

4) EFFECT OF VARYING c_u

We study the effect of varying c_u from 10 to 200 with 10 as a step in the Vancouver dataset, where the number of users is 2012 and the number of events is 225; we set the number of organizers as 250, and the waiting time of the arriving element is 50 minutes. The mean of c_o is 50, and the mean of events c_e is 50. Figs. 3a to 3c present the total utility, running time and memory usage when varying the mean of c_u. We report several observations as follows. First, the total utility increases when varying the mean of c_u because, as the mean of c_u increases, the capacity of a user increases, leading to more triples that can be arranged. Second, the running time and the memory usage are not changed much since the sizes of users, events, and organizers are not changed. Third, the weighted-threshold algorithm still performs better than other proposed algorithms for the online EATDM problem, and this algorithm consumes the most memory and requires the longest running time to record the history thresholds; however, it is still efficient for the EATDM problem to process each dynamic arrival element.

5) EFFECT OF VARYING c_e

We next study the effect of varying c_e in the Vancouver dataset; the number of users is 2012, the number of events is 225, the number of organizers is 250, and the waiting time of the arrival element is 50 minutes. Let the mean of c_e vary from 10 to 200 with 10 as a step, while the mean of c_u is 50 and the mean of events c_o is 50. Figs. 3d to 3f present the total utility, running time and memory usage while varying the mean of c_u. We report several observations as follows. First, the weighted-threshold algorithm still performs better than the other algorithms under the online scenario, achieving a solution that is very close to the optimal solution, and the threshold-based algorithm performs better than the greedy algorithm. Second, we can see that the total utility value increases when $|c_e|$ is high. The reason is that, with the increased mean of c_e, the event can accommodate more



(a) Utility of varying waiting time (b) Time of varying waiting time (c) Memory of varying waiting time

(d) Utility of Singapore (e) Time of Singapore (f) Memory of Singapore

(g) Utility of Auckland (h) Time of Auckland (i) Memory of Auckland

FIGURE 4. Results of varying waiting time, Singapore and Auckland.

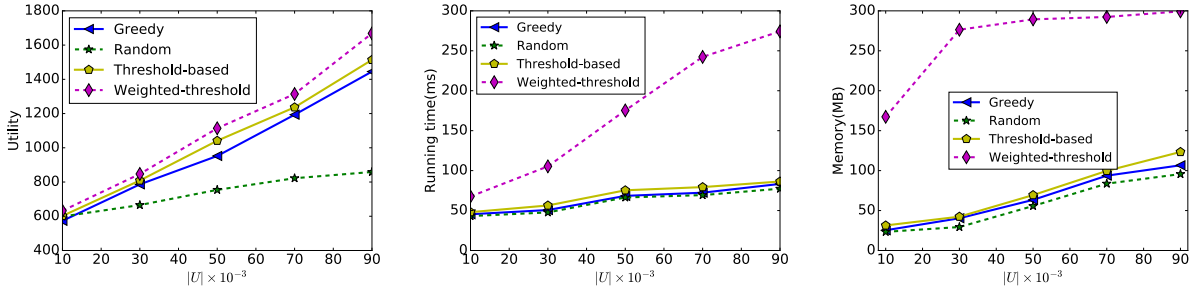
users, such that each triple tends toward a higher utility value. Third, the weighted-threshold algorithm requires more running time and memory usage than the other algorithms since it has to record the historical thresholds while processing the algorithm.

6) EFFECT OF VARYING c_o

Next, we analyze the effect of the mean of c_o on the Vancouver dataset, where the number of users is 2012, the number of events is 225, the number of organizers is 250, and the waiting time of the arrival element is 50 minutes. Let the mean of c_o vary from 10 to 200 with 10 as a step, while the mean of c_u is 50 and the mean of events c_e is 50. Figs. 3g to 3i present the total utility, running time and memory usage while varying the mean of c_o. We report several observations as follows. First, the total utility tends to increase when varying the mean of c_o as the number of organizers increases, and more triples can be arranged. We also observe that the weighted-threshold algorithm performs the best, and the threshold-based algorithm performs better than the greedy algorithm and random algorithm. However, the weighted-threshold algorithm requires a longer running time and more memory than the other algorithms under the online scenario. Second, the running time and memory usage still show no large changes because the numbers of users, events and organizers have not changed. Third, when a new element arrives, the weighted-threshold requires more time and memory to record the available thresholds, such that it consumes more time and memory usage than the random algorithm and greedy algorithm.

7) EFFECT OF w_t

We study the effect of waiting time (i.e., staying time) on the Vancouver dataset, and the waiting time is also varied from 10 to 200 with 10 as a step while other parameters are set by default. Figs. 4a to 4c present the result of varying the waiting time in terms of total utility, running time and memory usage. We report several observations as follows. First, we can observe that the total utility value increases as the waiting varies from 10 to 200 since the longer staying time has more triples that can be arranged. Second, the weighted-threshold algorithm still performs the best, and the threshold-based algorithm performs better than the random algorithm and greedy algorithm.



(a) Utility of scalability (b) Time of scalability (c) Memory of scalability

FIGURE 5. Results on scalability.

Third, for the running time and memory usage of processing each dynamically arriving element, the weighted-threshold algorithm requires more time and memory to record the history thresholds, and it needs constant adjustment of the probability of selecting the best threshold.

8) REAL DATASET

Next, we study the proposed method's effect on the real datasets of Singapore and Auckland; in these two datasets, we vary the waiting time from 10 to 200 with 10 as a step. Figs. 4d to 4f show the result on the Singapore dataset in terms of total utility, running time and memory usage, and Figs. 4g to 4i present the result on the Auckland dataset in terms of total utility, running time and memory usage. We first observed that the weighted-threshold algorithm performs the best in terms of utility, and the threshold-based algorithm performs better than the random and greedy algorithms since the threshold-based algorithm filters some low-utility triples. We also observed that running time and memory usage are related only to the numbers of users, events and organizers. Finally, we observed that the weighted-threshold algorithm requires more memory and a longer running time, as it records the available thresholds. However, the weighted-threshold algorithm processes each dynamically arriving element in less than one millisecond.

9) SCALABILITY

We finally study the scalability of the proposed algorithm, greedy algorithm, random algorithm, threshold-based algorithm and weighted-threshold algorithm. Here, we set the number of users as {10000, 30000, 50000, 70000, 90000}, and Figs.5a to 5a illustrate the experimental results of scalability in terms of total utility, running time and memory usage. We report the following observations. The total utility increases as the number of users U increases, which is natural, as more triples can be arranged. Moreover, the weighted-threshold algorithm performs the best, the threshold-based algorithm is second, followed by the greedy algorithm, and the random algorithm is the worst. We also observe that the weighted-threshold algorithm still requires more memory and a longer running time. However, the weighted-threshold algorithm processes a new dynamically arriving element and takes less than one millisecond. In addition, its memory usage consumption is less than 60 MB, and this memory consumption is still relatively small compared to the main memory.

C. EXPERIMENTAL SUMMARY

In terms of total utility value, the weighted-threshold algorithm performs better than the random algorithm, the greedy algorithm and the threshold-based algorithm. Although the weighted-threshold algorithm requires a longer running time and more memory to record all available thresholds, this algorithm processes a newly arrived element in less than one millisecond and consumes less than 60 MB to process an element in most case. In addition, the weighted-threshold algorithm performs the best, the threshold-based algorithm is second, followed by the greedy algorithm, and the random algorithm is the worst.

VI. CONCLUSION

In this paper, we formally defined the event arrangement problem in an online scenario, called event arrangement through three-dimensional matching (EATDM). We first showed that the EATDM problem is NP-hard under the offline scenario and introduced an offline solution called local search. Next, we proposed a straightforward solution to the EATDM problem, i.e., the greedy algorithm, and developed a basis solution, i.e., the random algorithm. To filter the low utility value, we proposed a threshold-based algorithm. Then, to improve the performance of the algorithm, we proposed a weighted-threshold algorithm that can dynamically adjust to choose different k with probability p_k, increasing the probability of k for which a better effect is selected. Finally, we verified the validity and practicability of the proposed algorithms on synthetic and real datasets.

In reality, the event-based social networks mentioned in this article are actually multiple networks, and we will discuss the effect of groups/communities in future work. There are some search methods related to groups and multiplex networks, such as [39]–[41], where users, events, and organizers are seen as a group and each social group reflects a social relationship. Therefore, we will consider group models in event-based social networks and analyze social group behaviors.

REFERENCES

[1] X. Liu, Q. He, Y. Tian, W.-C. Lee, J. McPherson, and J. Han, "Event-based social networks: Linking the online and offline social worlds," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2012, pp. 1032–1040.

[2] J. She, Y. Tong, and L. Chen, "Utility-aware social event-participant planning," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, New York, NY, USA, 2015, pp. 1629–1643.

[3] B. Li, Y. Cheng, G. Wang, and Y. Sun, "Incremental bilateral preference stable planning over event based social networks," *Complexity*, vol. 2019, Apr. 2019, Art. no. 1532013.

[4] H. Yin, L. Zou, Q. V. H. Nguyen, Z. Huang, and X. Zhou, "Joint event-partner recommendation in event-based social networks," in *Proc. IEEE 34th Int. Conf. Data Eng. (ICDE)*, Apr. 2018, pp. 929–940.

[5] A. M. Ahmed, T. Qiu, F. Xia, B. Jedari, and S. Abolfazli, "Event-based mobile social networks: Services, technologies, and applications," *IEEE Access*, vol. 2, pp. 500–513, Apr. 2014.

[6] Z. Qiao, P. Zhang, C. Zhou, Y. Cao, L. Guo, and Y. Zhang, "Event recommendation in event-based social networks," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 3130–3131.

[7] H. Khrouf and R. Troncy, "Topical community detection in event-based social network," 2018, *arXiv:1803.04354*. [Online]. Available: https://arxiv.org/abs/1803.04354

[8] J. Cao, Z. Zhu, L. Shi, B. Liu, and Z. Ma, "Multi-feature based event recommendation in event-based social network," *Int. J. Comput. Intell. Syst.*, vol. 11, no. 1, pp. 618–633, 2018.

[9] K. Li, W. Lu, S. Bhagat, L. V. Lakshmanan, and C. Yu, "On social event organization," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2014, pp. 1206–1215.

[10] N. Armenatzoglou, H. Pham, V. Ntranos, D. Papadias, and C. Shahabi, "Real-time multi-criteria social graph partitioning: A game theoretic approach," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, New York, NY, USA, 2015, pp. 1617–1628.

[11] Y. Tong, J. She, and R. Meng, "Bottleneck-aware arrangement over event-based social networks: The max-min approach," *World Wide Web*, vol. 19, no. 6, pp. 1151–1177, 2016.

[12] K. Feng, G. Cong, S. S. Bhowmick, and S. Ma, "In search of influential event organizers in online social networks," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, New York, NY, USA, 2014, pp. 63–74.

[13] J. She, Y. Tong, L. Chen, and C. C. Cao, "Conflict-aware event-participant arrangement," in *Proc. IEEE 31st Int. Conf. Data Eng.*, Apr. 2015, pp. 735–746.

[14] C. Chen, L. Zheng, V. Srinivasan, A. Thomo, K. Wu, and A. Sukow, "Conflict-aware weighted bipartite B-matching and its application to E-commerce," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1475–1488, Jun. 2016.

[15] W. Zhang and J. Wang, "A collective Bayesian Poisson factorization model for cold-start local event recommendation," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2015, pp. 1455–1464.

[16] Z. Wang, Y. Zhang, Y. Li, Q. Wang, and F. Xia, "Exploiting social influence for context-aware event recommendation in event-based social networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[17] G. Li, Y. Liu, B. Ribeiro, and H. Ding, "On group popularity prediction in event-based social networks," *IEEE Trans. Netw. Sci. Eng.*, to be published.

[18] F. Kou, Z. Zhou, H. Cheng, J. Du, Y. Shi, and P. Xu, "Interaction-aware arrangement for event-based social networks," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 1638–1641.

[19] T.-A. N. Pham, X. Li, G. Cong, and Z. Zhang, "A general graph-based model for recommendation in event-based social networks," in *Proc. IEEE 31st Int. Conf. Data Eng.*, Apr. 2015, pp. 567–578.

[20] J. She, Y. Tong, L. Chen, and C. C. Cao, "Conflict-aware event-participant arrangement and its variant for online setting," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 9, pp. 2281–2295, Sep. 2016.

[21] H. Liu, Y. Tong, P. Zhang, X. Lu, J. Duan, and H. Xiong, "Hydra: A personalized and context-aware multi-modal transportation recommendation system," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2019, pp. 2314–2324.

[22] J. She, Y. Tong, L. Chen, and T. Song, "Feedback-aware social event-participant arrangement," in *Proc. ACM Int. Conf. Manage. Data (SIGMOD)*, New York, NY, USA, 2017, pp. 851–865.

[23] A. Q. Macedo, L. B. Marinho, and R. L. T. Santos, "Context-aware event recommendation in event-based social networks," in *Proc. 9th ACM Conf. Recommender Syst. (RecSys)*, New York, NY, USA, 2015, pp. 123–130.

[24] D. S. Johnson and M. R. Garey, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, vol. 1. San Francisco, CA, USA: W.H. Freeman, 1979.

[25] C. H. Papadimitriou and M. Yannakakis, "Optimization, approximation, and complexity classes," *J. Comput. Syst. Sci.*, vol. 43, no. 3, pp. 425–440, 1991.

[26] C. A. J. Hurkens and A. Schrijver, "On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems," *SIAM J. Discrete Math.*, vol. 2, no. 1, pp. 68–72, 1989.

[27] M. Cygan, "Improved approximation for 3-dimensional matching via bounded pathwidth local search," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci.*, Oct. 2013, pp. 509–518.

[28] E. M. Arkin and R. Hassin, "On local search for weighted k-set packing," *Math. Oper. Res.*, vol. 23, no. 3, pp. 640–648, 1998.

[29] U. U. Hassan and E. Curry, "A multi-armed bandit approach to online spatial task assignment," in *Proc. IEEE 11th Int. Conf. Ubiquitous Intell. Comput. IEEE 11th Int. Conf. Autonomic Trusted Comput., IEEE 14th Int. Conf Scalable Comput. Commun. Assoc. Workshops (UIC-ATC-ScalCom)*. Washington, DC, USA: IEEE Computer Society, Dec. 2014, pp. 212–219.

[30] T. Song, Y. Tong, L. Wang, J. She, B. Yao, L. Chen, and K. Xu, "Trichromatic online matching in real-time spatial crowdsourcing," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 1009–1020.

[31] B. Li, Y. Cheng, Y. Yuan, G. Wang, and L. Chen, "Three-dimensional stable matching problem for spatial crowdsourcing platforms," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 1643–1653.

[32] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, "An optimal algorithm for on-line bipartite matching," in *Proc. 22nd Annu. ACM Symp. Theory Comput.*, 1990, pp. 352–358.

[33] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani, "Adwords and generalized on-line matching," in *Proc. 46th Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*, Oct. 2005, pp. 264–273.

[34] A. Mehta, "Online matching and ad allocation," *Found. Trends Theor. Comput. Sci.*, vol. 8, no. 4, pp. 265–368, 2013.

[35] S. Polyakovskiy, F. C. R. Spieksma, and G. J. Woeginger, "The three-dimensional matching problem in Kalmanson matrices," *J. Combinat. Optim.*, vol. 26, no. 1, pp. 1–9, 2013.

[36] H. F. Ting and X. Xiang, "Near optimal algorithms for online maximum edge-weighted b-matching and two-sided vertex-weighted b-matching," *Theor. Comput. Sci.*, vol. 607, pp. 247–256, Nov. 2015.

[37] A. Blum and C. Burch, "On-line learning and the metrical task system problem," *Mach. Learn.*, vol. 39, no. 1, pp. 35–58, 2000.

[38] Y. Cheng, Y. Yuan, L. Chen, C. Giraud-Carrier, and G. Wang, "Complex event-participant planning and its incremental variant," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 859–870.

[39] J. Jiang, B. An, Y. Jiang, C. Zhang, Z. Bu, and J. Cao, "Group-oriented task allocation for crowdsourcing in social networks," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.

[40] Z. Li, F. Yan, and Y. Jiang, "Cross-layers cascade in multiplex networks," *Auton. Agents Multi-Agent Syst.*, vol. 29, no. 6, pp. 1186–1215, Nov. 2015.

[41] W. Zhang, J. Wang, and W. Feng, "Combining latent factor model with location features for event-based group recommendation," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2013, pp. 910–918.

YUAN LIANG is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Beihang University. Her major interests are data mining, event-based social networks, crowdsourcing, and spatiotemporal data processing and analysis.