Joint User- and Event- Driven Stable Social Event Organization

Xin Wang[†] Wenwu Zhu[†] Chun Chen[§] Martin Ester[‡]

[†]Department of Computer Science and Technology, Tsinghua University, China
[§]Zhejiang Provincial Key Laboratory of Service Robot, College of Computer Science, Zhejiang University, China
[‡]School of Computing Science, Simon Fraser University, Canada
[†]{xin_wang,wwzhu}@tsinghua.edu.cn [§]chenc@zju.edu.cn [‡]ester@cs.sfu.ca

ABSTRACT

The problem of social event organization (SEO) rises with the advent of online web services and plays an important role in helping users discover new offline events. Existing work on SEO only assumes that different users have different preferences towards different events, ignoring the fact that each event (its organizer) may have a separate preference towards every user. In this paper, we investigate joint user- and event- driven SEO by simultaneously considering user preferences (towards events) and event preferences (towards users). A risen challenging problem is that this joint consideration may suffer instabilities between users and events which are NP-hard to handle in SEO. Stability is a desired property that needs to be maintained in SEO, otherwise participants will incline towards changing to other events and trust less the organizer. To the best of our knowledge, we are the first to study SEO with both preferences from users to events and preferences from events to users being considered. In this work, we formulate the stable social event organization (Stable-SEO) problem and prove its NP-hardness. We then propose an efficient greedy heuristic algorithm to solve Stable-SEO, taking both user preferences and event preferences into account. The proposed approach is able to find an assignment of a group of users to a set of events for an event organization, in which there is a minimized number of users involved in user-event pairs such that both the user and event in one such pair will be better off when reassigning the user to this event. Our experiments on two real-world datasets demonstrate the strong superiority of our proposed approach over existing methods.

ACM Reference Format:
Xin Wang[†] Wenwu Zhu[†] Chun Chen[§] Martin Ester[‡] . 2018. Joint User- and Event- Driven Stable Social Event Organization. In *Proceedings of The Web Conference 2018 (WWW 2018).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186063

1 INTRODUCTION

Online web services such as Meetup and Plancast which organize group meetings and social events have come into our sights in recent years, resulting in the emergence of the problem of organizing social events for a group of people. Being capable of utilizing online user information to get people meet offline, organizing social events has become a newly rising interesting and important application. Event organization normally involves two kinds of entities / actors – the organizers planning the event and the attendees taking part in the event. In real life applications, for instance, users on Meetup can act as organizers who schedule events and other users can be attendees deciding whether to participate in the event or not. In SEO, we focus on scenarios where users can choose only one event/activity to attend at one time (i.e., users can not leave their current events/activities and switch to others). We also note that SEO aims to address problems on *organizing* events rather than *detecting* events.

As a general setting [16], there is often *cardinality constraint* in SEO which bounds the capacity of an activity by a minimum quota and maximum quota. Taking table tennis as an example activity, the minimum quota is two and the maximum quota is four – two people can play a single match, three people can play a single match in turn and four people can play a double match. Besides, a user's preference towards attending an event in SEO depends on two crucial factors: how interesting the event itself is to her and the extent to which she is going to enjoy the company of the people who will also attend the event. The former factor is referred to as the user's *innate affinity* towards an event and the later as the user's pairwise *social affinity* towards other attendees of an event. There are many possible ways to compute the *innate affinity* and *social affinity* depending on what kind of data is available, and how these two important factors are calculated is orthogonal to the problem of SEO. We remark that the existing work [9, 16] on SEO assumes that different users have different preferences towards different events and does not consider the fact that each event (its organizer) may have a separate preference towards every user.

In this paper, we explore joint user- and event- driven SEO through considering both user preferences towards events/activities and event/activity preferences towards users, which may come across the instability issues that popularly exist in the fields of combinatorics [5, 6, 10, 15] between users and events. The property of *stability* is necessary to be maintained, otherwise users will tend to switch to other activities and lose faith in the organizers. Thereby, we formulate the problem of *stable social event organization* (Stable-SEO) to deal with the instability issues in SEO. Stable-SEO is advantageous because it takes care of stabilities between users and events in SEO. The stability problem in SEO is easy to handle when we take only user preferences or event preferences into consideration. However, the joint consideration of user and event preferences will make instability issues between users and events become NP-hard to solve in SEO. In other words, solving Stable-SEO is NP-hard and

WWW 2018, April 23–27, 2018, Lyons, France
© 2018 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC BY-NC-ND 4.0 License.
ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186063

therefore very challenging (proof in Section 3). Existing literature in theoretical computer science such as Stable Marriage Problem (SMP) [4], College Admissions Problem (CAP) [2] and National Resident Matching Problem [19] can not be utilized to tackle Stable-SEO for the reason that they consider merely innate affinity and do not take social affinity into consideration, therefore is a special case of our stable social event organization problem. This being the case, we develop a polynomial time greedy algorithm that considers both user preferences and event/activity preferences to tackle the problem of Stable-SEO. Inspired by the famous Gale-Shapley algorithm [4], the proposed method solves the instability issues in SEO through finding an assignment of a group of users to a set of events, which minimizes the number of users involved in user-event pairs such that both the user and event in one such pair will be better off when reassigning the user to the event (i.e., unstable), satisfying the cardinality constraint of the event (i.e., if the capacity of the event is not yet full, just assign the user to the event; if the capacity is full and the event prefers this user to at least one of the users already assigned to it, replace the less preferred user with this user). Compared to existing methods, our approach is capable of keeping the number of unstable users in SEO as small as possible and therefore is beneficial in preventing each individual from losing trust towards the event organizer(s).

In summary, this paper makes four major contributions:

- We explore joint user- and event- driven stable social event organization through considering both user preferences and event preferences simultaneously. To the best of our knowledge, this is the first work to investigate SEO taking both user and event preferences into account.
- We formulate *stable social event organization* (Stable-SEO) as an optimization problem that requires an assignment of users to events with the number of unstable users as small as possible, while satisfying the cardinality constraint of each event.
- We prove the NP-hardness and polynomial time inapproximability of the stable social event organization problem, and then propose a heuristic greedy algorithm, *User-stable Greedy*, to solve the stable social event organization problem.
- We conduct extensive experiments on two real-world datasets, Meetup and Plancast. The experimental results demonstrate the significant improvement of our proposed approach against several existing methods.

2 RELATED WORK

The Stable-SEO problem is closely related to Social Event Organization (SEO) [9, 16], in which events are indifferent among users and the goal is to find an assignment of users to events that maximizes the total social welfare between users and events. Our Stable-SEO departs from SEO in focusing on the goal of finding an assignment that involves minimum number of unstable users when events or their organizers have different preferences over different users. First of all, Li et al.'s work does not focus on any stability issues and therefore can not solve our Stable-SEO problem. Second, we argue that the stability of an assignment is much more a crucial criteria for evaluating the "goodness" of the assignment – the property of stability becomes extraordinarily important when activities (or their organizers) have different innate affinities towards different users. This is because the maximization of overall social welfare (Li et al.'s work) *may disregard individualistic needs for certain users* who "sacrifice" themselves for the sake of total "happiness" of all users, and therefore make stability indeed a desired property to have – if an event organization does not have that, participants will tend to switch to other events and have less faith in the organizers.

Aside from the SEO problem, Stable-SEO is also related to the stable matching problem (SMP) [4] in which the objective is to find a stable matching between men and women, where all men and women have preferences over members of the opposite sex. For a stable matching, there must be no man-woman pair such that both of them have the preference to exchange each other with their currently matched partners. Gale and Shapley give a polynomial algorithm to find such a stable matching for any instance of SMP. Many variants of SMP have been widely studied [6, 11, 15, 18–20], among which the Stable Matching with Master List (SM-ML) [11] and the National Resident Matching Problem (NRMP) [19] have particular close connections with Stable-SEO. The SM-ML problem is a natural extension of the vanilla SMP in which the preferences of men, women, or both, are derived from a *master* preference list. SM-ML has real-world applications in which users are ranked according to some objective criteria or a shared common criterion of users on the opposite side. Irving et al. [11] give a formal definition that a master list of men (resp. women) is a single list containing every man (resp. woman), and each woman's (resp. man's) preference list contains her (resp. his) partners ranked precisely according to the master list. NRMP, on the other hand, is a many-to-one extension of SMP where men are regarded as residents and women as hospitals. Every hospital has a maximum quota that bounds the maximum number of residents being served in the hospital. NRMP can be reduced to SMP through replacing each quota-q hospital by q copies of quota-1 hospitals, resulting in the fact that most conclusions established for SMP can be carried over to NRMP [6]. We remark that both SM-ML and NRMP are special cases of Stable-SEO even when no user-user social affinities which act as a key component in Stable-SEO are taken into consideration:

- Stable-SEO reduces to SM-ML (with users regarded as entities in the master list and events regarded as entities on the other side) when no user-user social affinities are considered and each event has a maximum quota of 1 and a minimum quota of 1.
- Stable-SEO reduces to NRMP (with users regarded as residents and events regarded as hospitals) when no user-user social affinities are considered and each event has a minimum quota of 0, plus a master list of users.

Stable-SEO and the problem of recommending item(s) to a group of users [1] (with events treated as items) share some common properties as well when we know which group of users will attend a certain unknown event together ahead of time. However, in our problem, we do not know the sets of users who will attend an event together (i.e., membership of the users in a group) in advance, and the goal of Stable-SEO is to find such sets of users by considering not only the innate affinities but also the social affinities. Stable-SEO is also related to recommending groups to users [21] when groups of people will for sure attend some certain events and the remaining users have not yet decided which event to attend, information about both of which is unfortunately unknown and unavailable to us. Furthermore, another important difference which distinguishes Stable-SEO from recommending items to group of users and recommending groups to users is that Stable-SEO cares about the property of *stability* while neither of the two recommendation problems do.

3 JOINT USER- AND EVENT- DRIVEN STABLE SOCIAL EVENT ORGANIZATION

Joint user- and event- driven stable social event organization, with both user preferences (towards events) and event preferences (towards users) being considered, makes instabilities in SEO become NP-hard to deal with. To tackle this challenge, we concretely describe the problem of stable social event organization (Stable-SEO) in this section. We first briefly introduce some preliminary knowledge about SEO, then give a formal formulation of Stable-SEO as a discrete optimization problem, followed by the illustration of its hardness and polynomial time inapproximation.

3.1 Preliminary

Consider a setting where a company is organizing an offsite event and we want to divide employees into smaller groups and each group will participate in an activity. A similar scenario happens in a large conference or convention, where the organizers may want to assign small groups of attendees to a set of social activities, to promote networking or enhance professional relationships.

The Social Event Organization (SEO) problem posed in [16] aims to address the above mentioned scenario. This problem is defined as follows.

- U is a set of *users* who are awaiting activity assignments. We are also given a social network, represented by an undirected graph $G = (U, E)$. If user u and user v are friends, there is an edge $(u, v) \in E$.
- A is a set of *activities* to be assigned. For each activity $a \in A$, there are two cardinality constraints: First, γ_a is the lower bound, i.e., there must be at least γ_a users assigned to a. Second, δ_a is the upper bound, i.e., the number of assignees must not exceed δ_a. For example, if a is chess, then $\gamma_a = \delta_a = 2$; if a is table tennis, then $\gamma_a = 2$ (single match) and $\delta_a = 4$ (double match); if a is outdoor movie, then $\gamma_a = 0$ and $\delta_a = \infty$.
- There is a user-activity affinity function $\sigma : U \times A \rightarrow \mathbb{R}$, such that $\sigma(u, a)$ is the innate affinity u has for a.
- There is also a user-user affinity function $\omega : U \times U \rightarrow \mathbb{R}$, such that $\omega(u, v)$ is the social affinity between user u and user v. We note that social affinity is symmetric, i.e., for all $u, v \in U$, $\omega(u, v) = \omega(v, u)$.
- Let S_a denote the set of users assigned to activity a under assignment M, then the social welfare of a produced by this assignment is:
$$\mu(S_a, a) = (1-\alpha) \cdot \sum\nolimits_{u \in S_a} \sigma(u, a) + \alpha \cdot \sum\nolimits_{u, v \in S_a, u \neq v} \omega(u, v). \quad (1)$$
- The SEO problem seeks an assignment $M\colon U \rightarrow A$, such that the total social welfare of M over all activities
$$SW = \sum_{a \in A} \mu(S_a, a) \quad (2)$$
is maximized and for all $a \in A$, such that $S_a \neq \emptyset$, the cardinality constraints are satisfied: $\gamma_a \leq |S_a| \leq \delta_a$.

The solution proposed in [16] assumes the innate affinities of activities (or their organizers) towards all users to be the same, i.e., activities (or their organizers) are indifferent to different users, although in many cases activities (or their organizers) may also have preference to see certain kinds of users participate in some activities over others. For instance, it is natural to assume that activity organizers tend to prefer attendees with a higher level of activity, attendees with a broader interest and so on. Furthermore, maximizing the total social welfare has the disadvantage of ignoring certain users' individualistic needs, making them "sacrifice" themselves and end up missing those more attractive activities which also prefer them more.

3.2 Stable Social Event Organization

Stable-SEO considers a more general case where activities (or their organizers) have different innate affinities towards different users, i.e., activities (or their organizers) may have preference to see certain users participate in some activities, and by optimizing an objective related to stability.

In addition to the user-activity innate affinity and user-user social affinity, we introduce an additional affinity function for the activity-user affinity:

- The activity-user affinity function $\kappa : A \times U \rightarrow \mathbb{R}$, such that $\kappa(a, u)$ is the innate affinity a has for u.

Let S_a denote the set of users assigned to activity a, and we define the user-activity utility of user u towards activity a for assigning u to a as:
$$\varphi(u, a) = (1-\alpha) \cdot \sigma(u, a) + \alpha \cdot \sum\nolimits_{v \in S_a} \omega(u, v), \quad (3)$$
and the activity-user utility of activity a towards user u for assigning u to a as:
$$\vartheta(a, u) = \kappa(a, u). \quad (4)$$

We note that the user-activity utility $\varphi(u, a)$ is a weighted combination of the innate affinity of u towards a and the sum of u's pairwise social affinities with the other users assigned to a. The activity-user utility $\vartheta(a, u)$ equals to $\kappa(a, u)$, the innate affinity of a towards u. Consistent with [16], an assignment M is defined as a mapping from U to A, i.e., $U \rightarrow A$. We further define $M^{-1}(a)$ as the set of users assigned to a, i.e., $M^{-1}(a) = \{u \in U | M(u) = a\}$ for $a \in A$. An assignment M is said to be *feasible* if all its cardinality constraints are satisfied, i.e., $\forall a \in A$ for which $M^{-1}(a) \neq \phi$, we have $\gamma_a \leq |M^{-1}(a)| \leq \delta_a$. Now we give the formal definition of stable assignment under the stable social event organization problem as follows.

Definition 3.1. Assignment M in a stable social event organization instance is stable if there is no user-activity pair (u, a) such that

- $M(u) = \phi \quad \vee \quad \varphi(u, a) > \varphi(u, M(u))$
- $|M^{-1}(a)| < \delta_a \quad \vee \quad \exists v \in M^{-1}(a) \wedge \vartheta(a, u) > \vartheta(a, v)$.

The two conditions above can be expressed as follows in plain English:

- User u is either not assigned to any activity or prefers a to the activity that she is currently assigned to.
- Activity a either has an unfilled position or prefers u to at least one of the users assigned to it.

Definition 3.2. A user-activity pair satisfying the two conditions in Definition 3.1 is an unstable pair.

Definition 3.3. A user in an unstable pair is an unstable user.

The objective of stable social event organization is to find a feasible assignment of a set of users to a set of events, in which the number of unstable users is minimized.

	u_1	u_2	u_3	u_4	u_5	u_6
a_1	1	2	3	4	5	6
a_2	1	2	3	4	5	6

Table 1: Activity-user innate affinities

	a_1	a_2
u_1	21	10
u_2	10	1
u_3	10	1
u_4	19	10
u_5	1	10
u_6	1	11

Table 2: User-activity innate affinities

	u_1	u_2	u_3	u_4	u_5	u_6
u_1	/	1	1	1	1	1
u_2	1	/	1	1	1	1
u_3	1	1	/	1	1	1
u_4	1	1	1	/	1	1
u_5	1	1	1	1	/	1
u_6	1	1	1	1	1	/

Table 3: User-user social affinities

Lemma 3.4. *A solution to SEO is not necessarily stable.*

Proof. We prove Lemma 3.4 through giving an example. Assume the example contains six users $u_1, \cdots, u_6$ and two events/activities a_1, a_2 where $\gamma_{a_1} = \delta_{a_1} = 3, \gamma_{a_2} = \delta_{a_2} = 3$. Table 1, 2 & 3 show the activity-user innate affinities $\kappa(\cdot,\cdot)$, user-activity innate affinities $\sigma(\cdot,\cdot)$ and user-user social affinities $\omega(\cdot,\cdot)$ for users and events respectively. As the SEO problem proposed in [16] assumes the activities (or their organizers) are indifferent to different users, the social welfare of an activity defined in (1) does not contain the activity-user innate affinities $\kappa(\cdot,\cdot)$. One way to cast the activity-user innate affinity into SEO could be to add an additional term $\kappa(\cdot,\cdot)$ after each user-activity innate affinity $\sigma(\cdot,\cdot)$, resulting in a modified social welfare function $\mu(S_a, a)$,

$$\mu(S_a, a) = (1-\alpha)\cdot \sum\nolimits_{u\in S_a} \left[\sigma(u,a) + \kappa(a,u)\right] + \alpha \cdot \sum\nolimits_{u,v\in S_a, u\neq v} \omega(u,v). \quad (5)$$

However, we note that as shown in Table 1, a_1 and a_2 have exactly the same activity-user innate affinities for different users in our example, so that the values of $\kappa(a,u)$ will not affect the summation of $\mu(S_a, a)$ over a_1 and a_2, i.e., the total social welfare $\sum_{a\in A} \mu(S_a, a)$ as defined in (2), no matter how the users are assigned to a_1 and a_2. In other words, if the total social welfare with $\mu(S_a, a)$ defined according to (1) is maximized, then it will also be a maximum with $\mu(S_a, a)$ defined according to (5).

We will still adopt the original definition in (1) to compute $\mu(S_a, a)$ for the sake of succinctness in the example. First of all, we can only assign three users to each of the two activities in order to satisfy the cardinality constraints, and the assignment M which maximizes the overall social welfare SW according to (2) is as follows: $S_{a_1} = \{u_1, u_2, u_3\}$, $S_{a_2} = \{u_4, u_5, u_6\}$. On the other hand, it is easy to calculate that $\varphi(u_4, a_1) = 10.5 > \varphi(u_4, a_2) = 6$ and $\vartheta(a_1, u_4) = 4 > \vartheta(a_1, u_1) = 1$. By Definition 3.1, (u_4, a_1) is an unstable pair and thus assignment M which maximizes the overall social welfare is not stable. Now let's consider another assignment M' where $S_{a_1} = \{u_2, u_3, u_4\}$, $S_{a_2} = \{u_1, u_5, u_6\}$, which is stable by Definition 3.1, and we have $SW(M') = 38 < SW(M) = 39$ by (1) and (2).

It is therefore sufficient to conclude that an assignment maximizing the overall social welfare defined in (2) may not be stable by Definition 3.1. □

In stable social event organization problem, we use user-activity utility $\varphi(u, a)$ and activity-user utility $\vartheta(a, u)$ to simulate the extend to which user u is interested in activity a and how the organizers of activity a like user u, which follows a natural intuition. The individual utility of a user towards an activity increases when her innate affinity towards this activity, as well as the sum of her pairwise social affinities towards those who will also attend the same activity, increases. The definition of user-activity utility just simply follows this intuition. Though other different definitions can also be applied to the user-activity utility and activity-user utility, we remark that this is orthogonal to our focus in this work and can be further investigated in future work. The problem of stable social event organization is formally defined in the following.

Problem 1. *Stable Social Event Organization (Stable-SEO) Given a set of users U, a set of events A where for each $a \in A$, there is a minimum quota $\gamma_a \in \mathbb{N}$ and a maximum quota $\delta_a \in \mathbb{N}$ such that $\gamma_a \leq \delta_a$, a user-user social graph $G = (U, E)$, a user-activity affinity function $\sigma(\cdot,\cdot)$, an activity-user affinity function $\kappa(\cdot,\cdot)$ and a user-user affinity function $\omega(\cdot,\cdot)$, find a feasible assignment $M: U \rightarrow A$ with the minimum number of unstable users.*

We would like to point out that minimizing the number of unstable pairs defined in Definition 3.2 may serve as an alternative objective for the stable social event organization problem. However, consider two cases, one with ten different users involved in ten unstable pairs and the other with one user involved in ten unstable pairs. It is natural to assume that the later case is more desirable than the former one in practice because we always prefer having less unsatisfied users (e.g., one rather than ten) in real-world (user-centric) applications. Therefore, we choose to minimize the number of unstable users instead of unstable pairs in this paper.

In applications such as a company organizing an offsite gathering and a research community hosting a scientific conference where all the activities share the same group of organizers, it is reasonable to assume that all the activities have the same activity-user utilities towards the attendees, similar to the above example shown in Table 1. Moreover, it will always be helpful to have an objective criterion for activity-user utility, such as how active the user is and how many events she has taken part in already – not every individual organizer is willing to generate the activity-user utility by herself. To make the stable social event organization problem less complex and more practical for real-world applications, we assume a common list of activity-user utilities for each instance and show that even the problem of stable social event organization with one common activity-user utility list is still very hard to solve in polynomial time in 3.3.

3.3 NP-Hardness of Stable-SEO

The first question to ask is that can Stable-SEO be solved in polynomial time? Unfortunately, we have a negative conclusion that the problem of stable social event organization is NP-hard. We give a polynomial-time reduction from the Hospitals / Residents Problem with Minimum Quota (HRMQ) [7] which receives lots of attention as a hard variant of the classic Hospitals / Residents Problem (HR) [4]. In HR, there are two sets, one is a set of residents and the other is a set of hospitals. Each resident has a preference list containing a rank of all hospitals in an non-increasing order and each hospital has a similar preference list towards all residents. Every hospital has an additional maximum quota, so that it can receive up to the quota number of residents. Then the objective is to find a feasible and stable matching between the set of residents and hospitals such that the number of residents assigned to each hospital will not exceed its maximum quota. Hamada et al. [7] are the first to introduce the concept of minimum quota, followed by Huang et al. [8] and Biró et al. [2]. Although Gale and Shapley [4] show that the problem of HR with only maximum quota can be solved efficiently in polynomial time, Hamada et al. [7] prove that it is NP-hard to find a feasible and stable matching that has the minimum number of unstable residents even if all the hospitals share a common preference list of all residents (i.e., a master list as stated in the original work).

Theorem 3.5. *Stable Social Event Organization is NP-hard.*

Proof. Hamada et al. [7] show that the problem of minimizing the number of unstable residents in HRMQ with a master list of residents (denoted as Min-ML-HRMQ) is NP-hard through a polynomial-time reduction from CLIQUE, a NP-complete problem, in which given a graph $G = (V, E)$ and a positive integer $K \leq |V|$, we are asked if G has a complete subgraph with K vertices. They complete the proof by building a large gap between the "yes" instance and "no" instance of CLIQUE.

An instance of Min-ML-HRMQ can be obtained from an instance of Stable-SEO which totally ignores the social affinities between users, i.e., setting $\omega(u, v) = 0$ for all $u, v \in U$ and $\alpha = 0$. The two sets of events/activities and users are hospitals and residents, with the cardinality constraints becoming the maximum and minimum quota. A preference list of events/activities can be constructed by ranking user-activity utilities in a non-increasing order with ties broken arbitrarily for each user and the master preference list of users can be similarly formed through ranking activity-user utilities for events/activities as well. Thus Min-ML-HRMQ is isomorphic to a special case of Stable-SEO where all user-user social affinities are zero, indicating that the hardness of Min-ML-HRMQ directly carries over to Stable-SEO. □

After obtaining the hardness result of Stable-SEO, it is important to ask about the hardness of approximating Stable-SEO. Besides the NP-hardness, we are also able to show a strong evidence for the inapproximability of Stable SEO through relating the approximability of Stable SEO to the approximability of *Dense k-Subgraph Problem (DkS)*. The *Dense k-Subgraph Problem (DkS)* is that given a graph G and a positive integer k, find an induced subgraph of G with k nodes which includes as many edges as possible. DkS is a NP-hard problem whose approximability has been widely studied. However, the gap between the approximability and inapproximability of DkS still remains large, and there have been no polynomial-time approximation schemes for DkS reported so far [3, 14]. Theorem 3.6 indicates that approximating Stable-SEO is equivalent to approximating DkS.

Theorem 3.6. *If the Dense k-Subgraph Problem (DkS) does not have any polynomial-time approximation algorithm, then Stable Social Event Organization has no polynomial-time approximation.*

Proof. Again, we start from the subproblem of Stable-SEO in which $\omega(u, v) = 0$ for all $u, v \in U$ and $\alpha = 0$. The reduction of Min-ML-HRMQ from Stable-SEO is the same as in the proof of Theorem 3.5.

Lemma 3.7. *If Min-ML-HRMQ has a polynomial-time c-approximation algorithm, then DkS has a polynomial-time $(1 + \epsilon)c^4$-approximation algorithm for any positive constant ϵ.*

The above Lemma 3.7 is proved in [7], and by taking its converse-negative proposition, it is clear that the hardness of approximation carries from DkS through Min-ML-HRMQ to our Stable-SEO problem. □

As a conclusion for the hardness results in this section, it is unlikely to approximate Stable-SEO within a constant ratio in polynomial time unless the inapproximability for DkS breaks down. We remark that the hardness of Stable-SEO is proved through showing the hardness of a subproblem of Stable-SEO, which means the full problem of Stable-SEO may be even more difficult.

4 THE ALGORITHM

Given the strong evidence for the inapproximability of Stable-SEO, we propose a heuristic greedy algorithm to solve the Stable-SEO problem in this section. Before introducing various solutions in detail, we first describe some common settings shared by the algorithms in this section.

Available. An activity a is said to be *available* if the number of users assigned to a has not exceeded its maximum quota, i.e., $|S_a| < \delta_a$. A user u is regarded as *available* if she has not been assigned to any activity.

User's List and Activity's List. Without loss of generality, we generate a user-activity utility list of activities for each user by ranking this user's user-activity utilities in a non-increasing order with ties broken arbitrarily and generate an activity-user utility list of users for all activities by ranking their common activity-user utilities in the same way. In our problem setting, the activity-user utility list contains all users and every user-activity utility list contains all activities, i.e., all of the utility lists are complete. For ease of reading, user's user-activity list and activity's activity-user list are referred to user's list and activity's list respectively when there is no confusion.

We note that the user's list may be updated after each iteration in the algorithms except *Random* because the order of activities in a user's list depends not only on the user-activity innate affinities but also the user-user social affinities between this user and every user currently assigned to an activity, while the activity's list will always remain the same as no user-user social affinities are taken into consideration to determine the users' orders in it.

General Algorithmic Procedure. For ease of understanding, we give a description about the high level ideas of the algorithms before drilling down into their details. Each user will have a (possibly) different user-activity list of all activities, $\mathcal{L}_u$, given the current assignment. During each iteration of an algorithm, the user-activity list $\mathcal{L}_u$ for each user may be updated and reordered if necessary. On the other hand, the common activity-user list $\mathcal{L}^*$ will remain unchanged during the whole running of the algorithm. Suppose the common activity-user utility list of all users for all activities is in the following order, $\mathcal{L}^* : u_1, u_2, \cdots, u_{|U|} \in U$, the algorithm will then iterate through u_1 to $u_{|U|}$. In the i^{th} iteration where $1 \le i \le |U|$, if u_i's list contains at least one available activity, then u_i will be assigned to the first available activity appearing on its list. It is possible to implement $\mathcal{L}_u$ (for each u) and $\mathcal{L}^*$ by a priority queue with its elements ranked in a non-increasing order with respect to their corresponding utilities. Last but not least, there will be a post-processing phase making sure that no cardinality constraints are violated, i.e., a feasible assignment is produced, and the details of the post-processing phase can vary a lot depending on different algorithms.

4.1 Random

Random is our first algorithm, and will be compared as a baseline in the experiments. The idea is to assign users randomly to activities and satisfy the cardinality constraints at the same time. To apply this strategy for solving Stable-SEO, *Random* algorithm first shuffles the common activity's list $\mathcal{L}^*$ for every activity and then shuffles the user's list $\mathcal{L}_u$ for each user u. It then iterates over each user through traversing the activity's list $\mathcal{L}^*$, and assigns each user u to the first available activity a (i.e.,$|S_a| < \delta_a$) on her list $\mathcal{L}_u$, ignoring the minimum quota. Our definition of *available* guarantees that the number of users assigned to each activity will not exceed its maximum quota. In order to obtain a feasible assignment, we finally move users arbitrarily from activities with surplus (i.e., the number of assigned users larger than the minimum quota) to activities with deficit (i.e., the number of assigned users is smaller than the minimum quota) while not creating any new activity with deficit. Obviously, this is a naive baseline which randomly assigns users to activities without violating any cardinality constraint.

4.2 Stable Greedy

We next introduce *Stable Greedy*, an algorithm that assigns users to activities taking both user-activity innate affinities and user-user pairwise social affinities into consideration. In particular, the activity's list $\mathcal{L}^*$ with entry $\langle u, \vartheta(a^*, u)\rangle$ is ordered non-increasingly by the activity-user utility $\vartheta(\cdot,\cdot)$ defined in (4), where a^* denotes an arbitrary activity. For each user, the algorithm orders her user's list $\mathcal{L}_u$ with entry $\langle a, g(u, a|S_a)\rangle$ by a non-increasing user-activity potential utility under the current temporary assignment (possibly incomplete), which is defined as follows,

$$g(u, a|S_a) = (1-\alpha)\cdot\sigma(u, a) + \alpha\cdot\sum\nolimits_{v\in S_a}\omega(u, v). \qquad (6)$$

Procedure Greedy($U, A, M, \mathcal{L}^*, \mathcal{L}$)

```
1  for $u = \mathcal{L}^*.PopUser()$ do
       // Traverse user from top to bottom
2      if $\exists a \in A \wedge |S_a| < \delta_a$ then
3          $M(u) \leftarrow \arg\max_{a\in A} g(u, a|S_a)$ on $u$'s list
4          $S_{M(u)} \leftarrow S_{M(u)} \cup \{u\}$
5      end
6      for $v \in U \wedge v \ne u$ do
7          if $\omega(u, v) > 0$ then
8              $\mathcal{L}_v.update(\langle M(u), g(v, M(u)|S_{M(u)})\rangle)$ // Equation (6)
9          end
10     end
11 end
```

Algorithm 1: Stable Greedy

```
1  $M(u) \leftarrow null, \forall u \in U; S_a \leftarrow \emptyset, \forall a \in A$
   // $a^*$ denotes an arbitrary activity
2  foreach $u \in U$ do
3      $\mathcal{L}^*.insert(\langle u, \vartheta(a^*, u)\rangle)$ // $\vartheta(\cdot,\cdot)$ in Equation (4)
4  end
5  foreach $u \in U$ do
6      foreach $a \in A$ do
7          $g(u, a|S_a) \leftarrow (1-\alpha)\sigma(u, a)$
8          $\mathcal{L}_u.insert(\langle a, g(u, a|S_a)\rangle)$
9      end
10 end
11 Procedure Greedy ($U, A, M, \mathcal{L}^*, \mathcal{L}$)
12 foreach $a \in A$ do
13     if $|S_a| \ne \gamma_a$ then
14         if $|S_a| < \gamma_a$ then
15             $D \leftarrow D \cup \{a\}$
16         else
17             $S \leftarrow S \cup \{a\}$
18         end
19     end
20 end
21 foreach $a \in S$ do
22     arbitrarily move at most $(|S_a| - \gamma_a)$ users in $a$ to activities in $D$ until for
         each activity $a' \in D$, $|S'_a| = \gamma'_a$
23 end
24 return $M$
```

Although (6) has a very similar form to (3), it should be noticed that $\varphi(u, a)$ defined in (3) is computed after the algorithm terminates, i.e., no more users will be assigned to activity a or any other activities. On the other hand, $g(u, a|S_a)$ is calculated under the current incomplete assignment when the algorithm is still running, indicating that more users could be assigned to activity a and other activities in the future. We argue that $g(u, a|S_a)$ in (6) and $\varphi(u, a)$ in (3) have different meanings, though they look similar. Apparently, the order of the common activity's list $\mathcal{L}^*$ will remain unchanged during the whole algorithm. At the beginning of the algorithm when no users are assigned to any activity, we have $g(u, a|S_a) = g(u, a|\emptyset) = (1-\alpha)\cdot\sigma(u, a)$ for each pair of user and activity. As is shown by Procedure Greedy in line 11 of Algroithm 1, the first available user u (i.e., currently most preferred) on $\mathcal{L}^*$ will be picked at each step and assigned to activity a which is available (i.e., $|S_a| < \delta_a$) and has the maximum value of $g(u, a|S_a)$ in a greedy manner, ignoring the minimum quota. Then for every remaining available user v on the list whose social affinity with u is positive, i.e., $\omega(u, v) > 0$, her utility towards a will increase

```
Procedure Score($a, U, A, \mathcal{L}^*, \mathcal{L}$)
1 $M_s(u) \leftarrow null, \forall u \in U; A_s \leftarrow A.Copy()$
2 $\delta_{a_s} \leftarrow \infty$ where $a_s = a$, $a_s \in A_s$
3 Procedure Greedy ($U, A_s, M_s, \mathcal{L}^*, \mathcal{L}$)
4 return $|S_{a_s}|$
```

as u being assigned to a. Thus $g(v, a|S_a)$ is then recomputed and the position of tuple $\langle a, g(v, a|S_a)\rangle$ in v's list $\mathcal{L}_v$ is updated as well (line 8 of Procedure Greedy). For the implementation of this algorithm, we can use a priority queue of tuple $\langle a, g(u, a|S_a)\rangle$ in a non-increasing order with respect to $g(u, a|S_a)$ to manage list $\mathcal{L}_u$ for each u (same for list $\mathcal{L}^*$). In the end, a post-processing which moves users arbitrarily from activities with surplus to activities with deficit is conducted to guarantee the feasibility of the final assignment (line 12 to line 23).

4.3 User-stable Greedy

Before describing the idea behind *User-stable Greedy*, let's first recall the *Stable Greedy*:

(1) Obtain an assignment through applying Procedure Greedy to a given Stable-SEO instance $I = \{U, A, M, \mathcal{L}^*, \mathcal{L}\}$.
(2) Move users arbitrarily from events/activities with surplus to those with deficit while not creating new events/activities with deficit.

For ease of notation, we denote $a[\gamma_a, \delta_a]$ as an activity with minimum quota γ_a and maximum quota δ_a. Suppose during the execution of *Stable Greedy*, a user u is moved from one activity $a[\gamma_a, \delta_a]$ ($|S_a| = \delta_a$) with surplus to another activity with deficit. Thus, not only user u will have the possibility of creating unstable pairs with a (and u becomes an unstable user), but also some other users may create unstable pairs with a and become unstable users because activity a now becomes available. Therefore, to minimize the potential for bringing in new unstable users, we propose our *User-stable Greedy* algorithm.

Given an existing Stable-SEO instance, we first construct a new instance of Stable-SEO (line 2 to line 4 in Algorithm 2), denoted as *max-1 Stable-SEO* in which all activities have either minimum quota being 0 and maximum quota being 1, i.e., $a[0, 1]$ (at most one user can be assigned to a), or both minimum and maximum quota being 1, i.e., $a[1, 1]$ (one and only one user must be assigned to a). We will then run the algorithm on this max-1 Stable-SEO instance and construct the solution for our original Stable-SEO instance from that for max-1 Stable-SEO instance in the end (line 32 to line 40 in Algorithm 2).

The calculation of $g(\cdot, \cdot)$ and maintenance of list $\mathcal{L}$ in *User-stable Greedy* will be the same as in *Stable Greedy* for consistency (line 5 to line 15). First, we apply Procedure Greedy to the max-1 Stable-SEO instance (line 16), and compute the deficiency (i.e., d) of the resulting assignment which is the summation of the deficiencies over all activities in the max-1 Stable-SEO instance, i.e., $\sum_{a' \in A'} max(\gamma_{a'} - |S_{a'}|, 0)$ (line 17 to line 19). Then we choose one activity $a[0, 1]$, set its maximum quota to ∞ and apply Procedure Greedy again so that all users who have very strong utilities towards attending activity a may actually be assigned to a in very high probabilities. As such, these users will be unstable users when they are moved to other activities with deficit while the remaining users on the other hand will have far less possibilities to become unstable. In this way, the potential for creating new unstable users can be dramatically reduced. This being the case, we are able to keep the number of unstable users as small as possible compared to *Stable Greedy*. We remark that merely extending the maximum quota of one activity to ∞ may not be sufficient to obtain a feasible solution and thus it is necessary to select more activities such that sufficiently many users can be moved to other activities with deficit in order to make the assignment feasible. On the other hand, we also should keep the number of selected activities minimum to guarantee the quality of our solution. We balance this trade-off by choosing d activities with the smallest **Score** values (ties are broken arbitrarily) from those activities which are not empty after the first run of procedure Greedy

```
Algorithm 2: User-Stable Greedy
1 $M(u) \leftarrow null, M'(u) \leftarrow null, \forall u \in U; S_a \leftarrow \emptyset, \forall a \in A; A' \leftarrow \emptyset; S_{a'} \leftarrow \emptyset, \forall a' \in A'; C \leftarrow \emptyset; d \leftarrow 0$
  // Convert $a[\gamma_a, \delta_a]$ into $\gamma_a \times a[1, 1]$ and $\delta_a \times a[0, 1]$
  // $a[\gamma_a, \delta_a]$ denotes an activity with minimum quota $\gamma_a$ and maximum quota $\delta_a$
2 foreach $a \in A$ do
3     $A' \leftarrow A' \cup \{a_1[1, 1], a_2[1, 1], \cdots, a_{\gamma_a}[1, 1]\} \cup \{a_{\gamma_a+1}[0, 1], a_{\gamma_a+2}[0, 1], \cdots, a_{\delta_a}[0, 1]\}$
4 end
  // $a^*$ denotes an arbitrary activity
5 foreach $u \in U$ do
6     $\mathcal{L}^*.insert(\langle u, \vartheta(a^*, u)\rangle)$ // $\vartheta(\cdot, \cdot)$ in Equation (4)
7 end
8 foreach $u \in U$ do
9     foreach $a \in A$ do
10        for $i \leftarrow 1$ to $\delta_a$ do
11            $g(u, a_i|S_{a_i}) \leftarrow (1 - \alpha)\sigma(u, a)$
12            $\mathcal{L}_u.insert(\langle a_i, g(u, a_i|S_{a_i})\rangle)$
13        end
14    end
15 end
16 Procedure Greedy ($U, A', M', \mathcal{L}^*, \mathcal{L}$)
17 foreach $a' \in A'$ do
18     $d \leftarrow d + max(\gamma_{a'} - |S_{a'}|, 0)$
19 end
20 $C \leftarrow \{a' | a' \in A', |S_{a'}| \neq \emptyset\}$
21 foreach $a' \in C$ do
22     if Score($a', U, A', \mathcal{L}^*, \mathcal{L}$) is among the $d$ smallest then
23         $\delta_{a'} \leftarrow \infty$
24     end
25 end
26 Procedure Greedy ($U, A', M', \mathcal{L}^*, \mathcal{L}$)
27 foreach $a' \in A'$ do
28     if $\gamma_{a'} = 0$ and $|S_{a'}| > 0$ then
29         arbitrarily move users in $a'$ to empty activities until a feasible assignment is obtained (by first making every $a'[1, 1]$ activity full)
30     end
31 end
  // Construct solution for Stable-SEO
32 foreach $a \in A$ do
33     for $i \leftarrow 1$ to $\delta_a$ do
34         if $S_{a_i} \neq \emptyset$ then
35             $u \leftarrow S_{a_i}$
36             $S_a \leftarrow S_a \cup \{u\}$
37             $M(u) \leftarrow a$
38         end
39     end
40 end
41 return $M$
```

(line 20 to line 25). As is shown in Procedure Score, the **Score** value of an activity measures the number of users assigned to it after running Procedure Greedy when setting its maximum quota to ∞. Since we are considering max-1 Stable-SEO, the deficiency of an assignment (i.e., d) equals to the number of empty $[1,1]$-activities. Let A_d be the set of these selected activities, we extend the maximum quota of all activities in A_d to be ∞ and then run Procedure Greedy again (line 26). We thereafter move users who are assigned to activities in A_d arbitrarily to empty activities to make the assignment feasible. We achieve this by first making $[1,1]$-activities full and if there still exist any activities in A_d having two or more users, then send remaining users arbitrarily to empty $[0,1]$-activities, or simply make them unassigned if there is no more $[0,1]$-activity left (line 27 to line 31). Finally, we construct the solution for our original Stable-SEO instance from the resulting assignment above for max-1 Stable-SEO instance (line 32 to line 40).

5 EMPIRICAL EXPERIMENTS

In this section, we compare the performance of our proposed method with several baselines on two real-world public datasets.

5.1 Experimental Settings

Evaluation Metrics.

We evaluate different algorithms based on two metrics:

- *Number of unstable users.* This is quite a straightforward evaluation metric which is consistent with the objective of Stable-SEO problem.
- *Total social welfare (SW).* We adopt this metric defined in (2) to demonstrate that our proposed method can achieve a fairly high social welfare as well when keeping a minimum number of unstable users.

	CHI	PAR	TKO	NYC	SFO	LAX
#users	11907	18322	7828	7662	9760	2287
#events	803	852	885	1380	1198	360

Table 4: Statistics of users and events in different cities

Datasets. We conduct experiments on two real-world datasets: *Meetup* and *Plancast*, which are first released by Liu et al. [17]. Both of them are online social networking services that facilitate offline social activities/events in various localities around the world. Specifically, Meetup allows its members to find and join groups unified by common interests such as politics, books, games, movies, health, pets, careers or hobbies[1]. Due to the widely geographical distribution of users and events in these two datasets, we extract users and events from six metropolitan areas including Chicago (CHI), Paris (PAR), Tokyo (TKO) on Meetup and New York City (NYC), San Francisco (SFO), Los Angeles (LAX) on Plancast for experiments. Table 4 shows the statistics of users and events in each area. Given that the calculation of utilities is orthogonal to our methods in this work, the way of computing different utilities will rely on the information available in each dataset.

- **Meetup.** For activity-user utility, we define $\kappa(a^*, u)$ to be the number of activities u has taken part in. We note that people on Meetup can set up and join various online interest groups which in turn allow the organizations of offline activities. Therefore, it is possible to establish a heterogeneous graph of user u, group g and activity a where there are two edges (u, g) and (g, a) connecting user u and activity a if user u participates in activity a and is a member of group g organizing activity a. Besides, users on Meetup are able to choose tags that best express their interests. Thus let $T(u)$ denote the set of tags labelled on u, for user-activity utility, we follow previous work [16] by defining $\omega(u, v)$ to be the Jaccard coefficient [12] between $Tag(u)$ and $Tag(v)$ (i.e., $\omega(u,v) = \frac{|Tag(u) \cap Tag(v)|}{|Tag(u) \cup Tag(v)|}$), and $\sigma(u, a)$ to be the *Katz* distance [13] between user u and activity a in the heterogeneous graph, where $Katz(u, v) = \sum_l^\infty \beta^l \cdot |\mathcal{P}_{u,v}^l|$. Here $\mathcal{P}_{u,v}^l$ is the set of l-length paths between u and v in the graph and β is a controlling parameter to make Katz put more emphasis on short length paths. Intuitively a larger Katz score indicates a more closely connection between a user u and an event a, demonstrating a higher user-activity innate affinity. As $\omega(u, v)$ (i.e., Jaccard coefficient) lies in $[0, 1]$, we normalize $\sigma(u, a)$ (i.e., Katz Score) and $\kappa(a^*, u)$ (i.e., number of activities participated) to prevent the domination of a certain kind of affinity.
- **Plancast.** We define $\kappa(a^*, u)$ to be the number of activities u has participated in Plancast, which is same as what we do for Meetup. Since Plancast allows its users to subscribe to each other so that subscribers can receive their subscribees' updates, we set $\sigma(u, a)$ to be the number of users in activity a that have been subscribed by u and $\omega(u, v)$ to be the Katz distance between u and v. Similarly, we also normalize $\kappa(a^*, u)$, $\sigma(u, a)$ and $\omega(u, v)$ to make them fall in the range of 0 and 1.

Due to the lack of ground truth information, we follow the same preprocessing strategies in [16] and generate the maximum/minimum cardinality constraints for all events as follows. We denote $\rho = \frac{\#users}{\#events}$ as the ratio between the number of users and the number of events. Then for event $a \in A$ in all areas, we sample its maximum quota δ_a from Gaussian distribution $\mathcal{N}(2\rho, \rho)$ and sample its minimum quota γ_a from uniform distribution $\mathcal{U}(1, \delta_a)$. Moreover, α is set to 0.5 for all methods in the experiments. Finally, Figure 1 shows the distributions of social affinity and Figure 2 displays the normalized event-user affinity for all six metropolitan areas.

Methods for Comparisons. The following five algorithms including our proposed method are tested in our experiments.

- *User-Stable Greedy (USG).* Our proposed *User-Stable Greedy* algorithm, which takes both innate affinity and social affinity in to account, and aims to minimize the number of unstable users for an assignment.
- *Stable Greedy (SG).* The *Stable Greedy* algorithm introduced in Section 4 that minimizes the total number of unstable user-event pairs, taking both innate affinity and social affinity into consideration.
- *PCADG.* The *Phantom- and Community- Aware Dynamic Greedy* algorithm proposed in [16], whose objective is to maximize the total social welfare (Eq (2)) given an assignment.
- *NRMP+.* The *NRMP+* baseline proposed in [16] with activity-user innate being set to $\kappa(\cdot, \cdot)$ in our experiments. This algorithm does not take social affinity into consideration.
- *Random (RD).* The naive method that randomly assigns users to activities without violating the cardinality constraint.

[1] https://en.wikipedia.org/wiki/Meetup_(website)

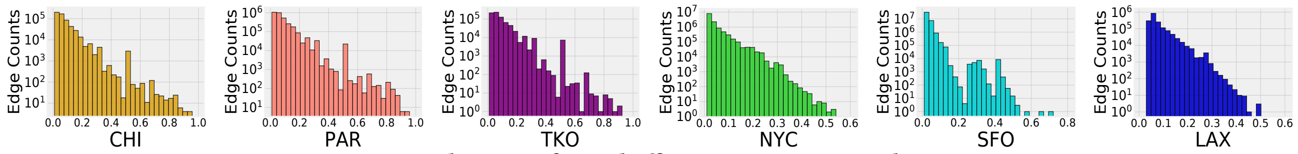

Figure 1: Distributions of social affinities in six metropolitan areas

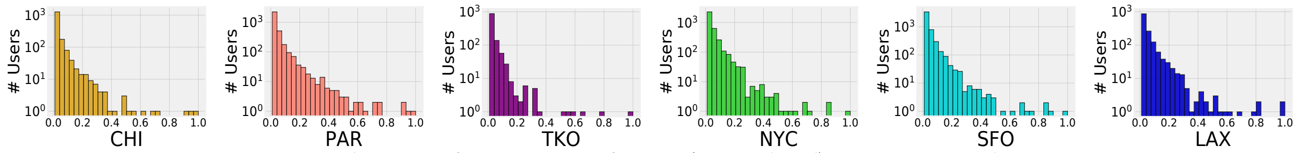

Figure 2: Distributions of event-user affinities (normalized) in six metropolitan areas

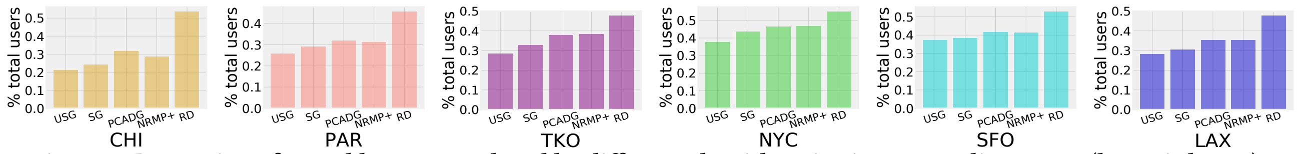

Figure 3: Proportion of unstable users produced by different algorithms in six metropolitan areas (lower is better)

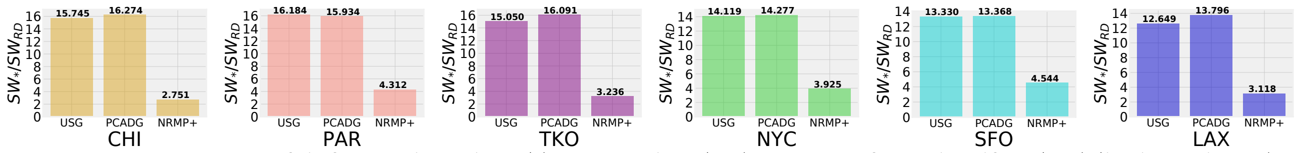

Figure 4: Improvement of different algorithms ($*$) over random (RD) in terms of social welfare (SW) (higher is better)

5.2 Results and Analysis

Number of unstable users. We present the proportion of unstable users with respect to the total number of users (the lower, the better) for our five comparison algorithms, i.e., User-Stable Greedy, Stable Greedy PCADG, NRMP+ and Random, in each area. As is shown in Figure 3, User-Stable Greedy (USG) clearly generates the least unstable users and therefore beats all other methods for all six cities, followed by Stable Greedy (SG) which tries to minimizes the number of unstable user-event pairs rather than the number of unstable users. As the most naive baseline, Random with no doubt has the worst performance. Besides, PCADG and NRMP+ have comparable performances, lying between Stable Greedy and Random, with PCADG outperforming NRMP+ for TKO, NYC, LAX and NRMP+ outperforming PCADG for CHI, PAR and SFO. This may be due to the reason that PCADG aims to maximize the total social welfare defined in (2) and NRMP+ ignores the social affinities among users. We can observe from Figure 3 that the proposed User-Stable Greedy algorithm achieves a performance boost over the best non-naive (non-random) method by a large margin (from 9.5% for SFO to 31.4% for LAX).

Total social welfare. Figure 4 shows the test of relevant improvement of User-Stable Greedy (USG), PCADG and MRMP+ over Random method (RD) in terms of social welfare (SW). We adopt the same evaluation methodology in [16] by measuring SW_*/SW_{RD}, where SW is defined in (2) and $*$ denotes different algorithms. One observation from Figure 4 is that NRMP+ performs the worst because of not considering any social affinities. We also observe that User-Stable Greedy (USG) and PCADG have comparable performances, with User-Stable Greedy (USG) slightly outperforming PCADG for PAR and PCADG slightly beating User-Stable Greedy (USG) for all the other cases.

Thereby, it is fair to conclude that our experiments demonstrate the superiority of our proposed method over other comparative approaches in producing a minimum number of unstable users and maintaining quite large social welfare at the same time. That is to say, our proposed approach is capable of reducing the number of "unhappy" individuals as much as possible and keeping the overall happiness of an event organization high simultaneously.

6 CONCLUSIONS

Social Event Organization (SEO), as a novel research topic, aims to help people gather from online to offline through organizing different social events/activities. Existing work on SEO only considers the preferences from users to events/activities (i.e., user-activity utilities) and fails to take preferences from events/activities or their organizers to users (activity-user utilities) into account. In this paper, we focus on SEO with both user preferences and event preferences being taken into consideration and solve the unstable issues brought by considering the two-side preferences. We formally formulate the Stable-SEO propblem, prove its NP-hardness and inapproximability in polynomial time. We propose an algorithm to generate the most minimum number of unstable users and achieve fairly large social welfare compared to existing methods. Extensive experiments verify the efficacy of our proposed method.

ACKNOWLEDGMENTS

This research is supported by China Postdoctoral Science Foundation No. BX201700136, National Program on Key Basic Research Project No. 2015CB352300 and National Natural Science Foundation of China Major Project No. U1611461.

REFERENCES

[1] Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawla, Gautam Das, and Cong Yu. 2009. Group Recommendation: Semantics and Efficiency. *PVLDB* 2, 1 (2009).
[2] "Péter Biró, Tamás Fleiner, Robert W. Irving, and David Manlove. 2010. The College Admissions problem with lower and common quotas. *Theor. Comput. Sci* 411, (34-36) (2010), 3136–3153.
[3] Uriel Feige. 2002. Relations between average case complexity and approximation complexity. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. ACM, 534–543.
[4] David Gale and Lloyd S Shapley. 1962. College admissions and the stability of marriage. *The American Mathematical Monthly* 69, 1 (1962), 9–15.
[5] David Gale and Marilda Sotomayor. 1985. Some remarks on the stable matching problem. *Discrete Applied Mathematics* 11, 3 (1985), 223–232.
[6] Dan Gusfield and Robert W Irving. 1989. *The stable marriage problem: structure and algorithms*. MIT press.
[7] K. Hamada, K. Iwama, and S.Miyazaki. 2008. The Hospitals/ Residents problem with quota lower bounds. In *MATCH-UP: Matching Under Preferences Workshop at ICALP*.
[8] Chien-Chung Huang. 2010. Classified stable matching. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 1235–1253.
[9] Jianbin Huang, Yu Zhou, Xiaolin Jia, and Heli Sun. 2016. A novel social event organization approach for diverse user choices. *Comput. J.* 60, 7 (2016), 1078–1095.
[10] Robert W Irving. 1985. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms* 6, 4 (1985), 577–595.
[11] Robert W Irving, David F Manlove, and Sandy Scott. 2008. The stable marriage problem with master preference lists. *Discrete Applied Mathematics* 156, 15 (2008), 2959–2977.
[12] Paul Jaccard. 1901. *Distribution de la Flore Alpine: dans le Bassin des dranses et dans quelques régions voisines*. Rouge.
[13] Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (1953), 39–43.
[14] Subhash Khot. 2006. Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM J. Comput.* 36, 4 (2006), 1025–1071.
[15] Donald Ervin Knuth. 1997. *Stable marriage and its relation to other combinatorial problems: An introduction to the mathematical analysis of algorithms*. Vol. 10. American Mathematical Soc.
[16] Keqian Li, Wei Lu, Smriti Bhagat, Laks VS Lakshmanan, and Cong Yu. 2014. On social event organization. In *KDD*. ACM, 1206–1215.
[17] Xingjie Liu, Qi He, Yuanyuan Tian, Wang-Chien Lee, John McPherson, and Jiawei Han. 2012. Event-based social networks: linking the online and offline social worlds. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1032–1040.
[18] David F Manlove, Robert W Irving, Kazuo Iwama, Shuichi Miyazaki, and Yasufumi Morita. 2002. Hard variants of stable marriage. *Theoretical Computer Science* 276, 1 (2002), 261–279.
[19] A. E. Roth. 1984. The evolution of the labor market for medical interns and residents: a case study in game theory. *Journal of Political Economy* 6, 4 (1984), 991–1016.
[20] Marilda Sotomayor. 1990. Two-sided matching: A study in game-theoretic modeling and analysis. *Econometric Society Monographs. Cambridge University Press, Cambridge* (1990).
[21] Xin Wang, Roger Donaldson, Christopher Nell, Peter Gorniak, Martin Ester, and Jiajun Bu. 2016. Recommending Groups to Users Using User-Group Engagement and Time-Dependent Matrix Factorization. In *Thirtieth AAAI Conference on Artificial Intelligence*.