On Social Event Organization

Keqian Li† Wei Lu† Smriti Bhagat‡ Laks V.S. Lakshmanan† Cong Yu॥

†University of British Columbia
Vancouver, B.C., Canada
{keqianli,welu,laks}@cs.ubc.ca

‡Technicolor Research
Palo Alto, CA, USA
smriti.bhagat@technicolor.com

॥Google Research
New York, NY, USA
congyu@google.com

ABSTRACT

Online platforms, such as Meetup and Plancast, have recently become popular for planning gatherings and event organization. However, there is a surprising lack of studies on how to effectively and efficiently organize social events for a large group of people through such platforms. In this paper, we study the key computational problem involved in organization of social events, to our best knowledge, for the first time.

We propose the Social Event Organization (SEO) problem as one of assigning a set of events for a group of users to attend, where the users are socially connected with each other and have innate levels of interest in those events. As a first step toward Social Event Organization, we introduce a formal definition of a restricted version of the problem and show that it is NP-hard and is hard to approximate. We propose efficient heuristic algorithms that improve upon simple greedy algorithms by incorporating the notion of phantom events and by using look-ahead estimation. Using synthetic datasets and three real datasets including those from the platforms Meetup and Plancast, we experimentally demonstrate that our greedy heuristics are scalable and furthermore outperform the baseline algorithms significantly in terms of achieving superior social welfare.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining

Keywords

Social Networks; Event Organization; Assignment Problems

1. INTRODUCTION

The problem of organizing social events for a large group of people can resonate with the academic community: we are all familiar with the "canned" social events offered by conference organizers, which are often a simple list of activities to choose from and attendance at each event is organically decided by people at the conference based on who they know and how interested they are in the events.

KDD'14, August 24–27, 2014, New York, NY, USA.

http://dx.doi.org/10.1145/2623330.2623724.

More broadly, event organization is one of the most important social activities online, with many companies, established giants and startups alike, such as Meetup[1] and Plancast[2], offering platforms for their users to plan and organize events. The social networks data and the data indicating the interests of the users on these platforms offer a rich setting in which events can be organized effectively and efficiently. Despite its importance, there are few academic studies on this topic: most of the existing research is focused on event detection through time series analysis (e.g., burst detection). Our goal in this paper is to provide a first principled definition of Social Event Organization (SEO) and a platform of algorithmic solutions for addressing this problem. To the best of our knowledge, this is the first paper on *organizing* events, instead of *detecting* events.

The tasks of organizing various social events share many common characteristics. First, each user often has his or her unique preference for the events being offered, a numeric measure which we call a user's *innate affinity* towards an event. Innate affinities can be stated explicitly: e.g., a user can mark his preference for a game of chess as 9 out of 10. Or they can be categorically stated by the users and computed by the organizer: e.g., a user can list outdoor activities in her profile and the organizer can deduce that she will likely prefer hiking to playing chess. Finally, they can be predicted by a recommender system based on the past events that the user has participated in. Regardless of how innate affinities are computed, they are one of the two key ingredients for users' happiness, for purposes of successful organization of social events.

The other key ingredient is the social connections among users – individuals often enjoy an activity more if they attend it along with their friends or others whom they would like to be around with. This connection is often defined between a pair of users and we call this pairwise *social affinity*. Similar to innate affinities, social affinities can be stated explicitly by the users. For example, users on social networks explicitly provide their friendship connections. They can also be deduced based on users' interests or past activities. For instance, two users who share lots of interests and past activities are more likely to enjoy each other's company. Thus, similar to innate affinity, social affinity is represented by a numerical measure.

Importantly, in real world situations, events typically come with natural *cardinality constraints*. For example, most sports activities need a certain number of participants: two or four for tennis, two for chess, and two to nine for poker games.

We illustrate aforementioned characteristics more concretely through the following motivating example. Academic conferences and business conventions often provide networking opportunities for attendees by offering social events. Planning them involves gathering individuals with similar interests to facilitate interactions

[1]http://www.meetup.com
[2]http://www.plancast.com

as well as ensuring that they have a good time. The current practice is to offer these events as a "canned" list of options: e.g., everybody selects from a small set of event choices based on their interests, perhaps uses ad hoc conversations among friends for coordination. Being able to automatically assign people to events from a large list of options, in a *centralized* manner, taking into account both users' interests and friendships will likely lead to people enjoying those events more.

The above example illustrates the core technical challenge inherent in event organization, viz., the *assignment problem*, where the system has to *assign* a given set of users to events, with the goal of making users happy. Other examples in this category include organization of activities at a reunion party, corporate activity day, community fund raising, or volunteering gatherings.

In this first paper towards Social Event Organization, we focus on the assignment problem illustrated above. Specifically, we are given a set of users, who are socially connected and have varying inherent interests in the events being offered. We need to assign users to events so as to maximize the "overall happiness" of the users, while respecting the event cardinality constraints (min and max). In event organization over a wide area, in addition to cardinality constraints, time and location also play a role. E.g., user's proximity to the event's location may affect the user's innate affinity for the event. A similar remark holds for the time at which the event is organized. In order to allow us to focus on the key technical problem of assignment, we assume that users' proximity to the location of events as well as availability of users for the events are factored in while determining the innate affinity of a user to an event. Consequently, *our focus is squarely on finding good assignments of users to events as opposed to scheduling of events.*

One critical aspect of the solution is to define the overall happiness as a combination innate and social affinities. We provide an intuitive definition of *social welfare*, as a linear combination of (i) the total innate affinities enjoyed by the users to the events they are assigned, and (ii) the aggregate social affinities enjoyed by pairs of users assigned to the same event (exact definition in §3.1). This focus allows us to cast the Social Event Organization problem as a constrained discrete optimization problem, for which we can explore algorithmic solutions.

Our event organization problem has close connections with several major bodies of work – generalized assignment problem (GAP) [20, 7, 6, 12], a family of two-sided matching problems including the National Resident Matching Problem (NRMP) [19], and community-search problems [21]. We depart from them by taking both innate and social affinities into account and solving the event organization problem under a unified framework. A detailed comparison with these related works appears in §2.

Specifically, we make the following contributions.

- We formally define social event organization as a constrained discrete optimization problem that asks for an assignment with maximum social welfare while respecting cardinality constraints of events (§3.1).
- The problem is shown to be NP-hard. Given this, we analyze its approximability: we establish close connections between restricted versions of our problem and the SUBSET-SUM problem and DENSE k-SUBGRAPH problem. We exploit these connections to offer strong evidence that the problem is hard to approximate (§3.2).
- We then develop several heuristics including a greedy hill-climbing strategy and several improvements. Our improvements are based on a notion of "phantom events" which refer to tentatively scheduled events which have yet to meet their lower cardinality constraint, and a look-ahead based technique (§4).
- We conduct a comprehensive experimental study on both synthetic and real data sets from Plancast, Meetup and SIGCOMM 2009. Our results show that our methods with look ahead estimation far exceed the baseline methods in terms of the social welfare achieved and the regret attained. Furthermore, our most sophisticated algorithm completes event assignments on a dataset consisting of 100,000 users and 500 events in under 17 minutes (§5).

Finally, we conclude the paper and summarize interesting directions for future work in §6.

2. RELATED WORK

Our problem has close connections to the generalized assignment problem (GAP) [20, 7, 6, 12]: there are multiple bins with capacity constraints, multiple objects with a size and a profit for placing an object in a bin; the problem is to find an assignment of objects to bins respecting capacity such that the total profit is maximized. In a special case, all objects have the same size and capacity is replaced by (maximum) cardinality. Differences between GAP and event organization are two-fold. First, events typically have lower bound constraints on cardinality: e.g., a game of Mahjong cannot be played by fewer than four people. Both lower and upper cardinality constraints must be respected by the solutions we propose. Very recently, lower bound constraints have been studied in the context of GAP [12], and the precise relationship between event organization and [12] will be made clear in §3.2. Second, social affinity is not considered in GAP, whereas it is a key ingredient in social event organization.

Our problem is also related to the National Resident Matching Problem (NRMP) and other two-sided matching problems such as college admissions [19, 9, 8, 5]. In NRMP, medical school graduates are matched with residency programs offered by hospitals throughout the U.S. The differences with event organization are three-fold. First, graduates choose residency programs as individuals, and social affinity is almost never a concern, except in rare cases where two graduates are a couple. Second, in NRMP, every graduate provides an ordered list of programs she'd like to join *and* every program has its own preference list over graduates. *Stability* is an important concern. A matching is stable if there is no pair of a graduate and a program so that they both prefer each other over the current assignment they get. While stability is similar in spirit to avoiding regrets (i.e., assigning a user to an event she does not enjoy) in event organization, in event organization stability is not important as events treat all users alike! Third, in contrast to the objectives in NRMP, our goal is to maximize social welfare (defined in §3.1) of the assignment, taking both innate and social affinities into account, while respecting the cardinality constraints of events. In §4, we adapt the NRMP solution to solve our problem, and empirically compare it in §5.

In a sense, event organization is also similar to group recommendations: so long as we know which set of users will attend the same event, i.e., group memberships, but not which one, we can, in principle, determine that event using group recommendations [2]. However, we do not know these groups and part of the challenge is to find them. Besides, our assignment of users (and hence of groups) to events is more "holistic" in considering not only innate, but also social affinity: it is their combination that drives our assignment. Thus, group recommendations cannot solve our problem.

Team formation problems in social networks [13, 3] assume for each user a set of skills and for each pair of users a compatibility measure. The problem is to put together teams that satisfy the skill requirements of a given task while minimizing the communication overhead of the team, e.g., the diameter or weight of the minimum spanning tree of the graph of the formed team. The problem is NP-hard and many of the works resort to approximation algorithms. Superficially, the team formation problem seems similar to ours. However, while social affinity has a counterpart in compatibility, there is no direct innate affinity between users and tasks, except indirectly via skills. More importantly, cardinality constraints in event organization make the problem significantly more challenging. Finally, a user may be assigned to more than one task whereas users are assigned to at most one event in our problem.

Also relevant to our work is the community search problem [21], where given a set of query nodes, the task is to find a community containing the query nodes that maximizes the overall social affinity, subject to a maximum cardinality constraint. However, there is no notion of events, nor users' innate affinities towards events in [21]. In contrast, our problem involves a combination of social and innate affinities and thus a *holistic approach* is needed which at once pays attention to the innate aspect (matching quality) and the social aspect. There have also been studies on geo-social query processing [4]. Though both our problem and theirs consider social effects, the SEO problem is inherently an assignment problem and cannot be solved by issuing queries.

3. SOCIAL EVENT ORGANIZATION

We now formulate Social Event Organization (SEO) as an assignment problem and study its hardness and approximation.

3.1 Problem Definition

Consider a setting where organizers of a large gathering (e.g., an international conference or a company-wide retreat) are planning social activities for the conference attendees. Or a scenario in which or an event planning platform (e.g., Meetup) is being used to plan social events for its users over a wide local area such as a city. In both cases, there is a set of users U who must be assigned to events or activities from a given set of possible events A. We assume the presence of a social network, through which the friendships among those users can be obtained. Let $G = (U, E)$ be the friendship graph induced on U. Each event $a \in A$ is associated with a *minimum cardinality bound* γ_a and a *maximum cardinality bound* δ_a, as motivated in §1.

There is a function $\sigma : U \times A \to \mathbb{R}_+$ such that for each user u and event a, $\sigma(u, a)$ models the user's *innate affinity* for the event. We often denote $\sigma(u, a)$ as $\sigma_{u,a}$ for simplicity. As mentioned in the introduction, we assume that the innate affinity $\sigma_{u,a}$ factors in u's proximity to a's location and u's availability w.r.t. the time at which a is held. In addition to their innate affinity to events, users have pair-wise social affinity with each other, and we capture this by a function $w : U \times U \to \mathbb{R}_+$, such that $w(u, v)$ is the social affinity between u and v. Both innate affinity to events and social affinity to other users attending the same event contribute to a user's sense of "utility" for attending an event. Note that as discussed in §1, social affinity is important because an individual may enjoy an event more when she is joined by her friends. We assume in this paper that social affinity is non-negative and that social affinity values are symmetric, i.e., $w(u, v) = w(v, u)$, for all $u, v \in U$.

Suppose a set of users $S \subset U$ is assigned to an event $a \in A$. We define the *utility* of this assignment as:

$$\mu(S, a) = (1 - \alpha) \sum_{u \in S} \sigma_{u,a} + \alpha \sum\nolimits_{u,v \in S, u \neq v} w(u, v), \quad (1)$$

where $\alpha \in [0, 1]$ can be chosen by the user of the SEO system, i.e., the organizer, to adjust the relative importance of the two affinities.

That is, the overall utility of the set of participants of an event is a linear combination of the total innate affinity of the participants for the event and the total social affinity of all pairs of participants. As a special case, we define $\mu(\emptyset, a) = 0$: when nobody is assigned to an event, it results in zero utility. As mentioned in the introduction, both innate affinity $\sigma_{u,a}$ and social affinity $w(u, v)$ can be provided by the users explicitly through stated interests and friendships. However, in case such explicit information is not available, those values can be estimated from a user's implicit interests and past activities. For example, social affinity can be computed using well-studied social distance measures (e.g., graph distance, Katz, or hitting time) [14]. We defer the details to §5.

An *assignment* is a (possibly partial) function $M : U \to A$. For $a \in A$, we denote by $M^{-1}(a)$ the (possibly empty) subset of users assigned to a, i.e., $M^{-1}(a) = \{u \in U \mid M(u) = a\}$. An assignment M is said to be *feasible* provided all cardinality constraints are satisfied, i.e., $\forall a \in A$ for which $M^{-1}(a) \neq \emptyset$, we have $\gamma_a \leq |M^{-1}(a)| \leq \delta_a$, in words, for all events to which at least one user is assigned, the number of users assigned to that event lies within its lower and upper bounds. Notice that owing to the cardinality constraints and the number of users and events in an instance of the assignment problem, it may not be possible to find a feasible assignment that covers all users: e.g., we may have 10 users and 1 event such that the event can accommodate at most 5 people. This is the reason we allow partial functions above. Similarly, not all events may be scheduled by an assignment, i.e., M need not be onto. We define the *social welfare* of a feasible assignment M as

$$\omega(M) = \sum\nolimits_{a \in A} \mu(M^{-1}(a), a). \quad (2)$$

The overall social welfare of an assignment is thus determined by the sum of utilities of the assignments made to each event. Our objective is to find a feasible assignment that maximizes the overall social welfare. Our definition of overall social welfare reflects the following intuition. A user's personal utility for an assignment increases with her innate affinity for the event, as well as the social affinity she has toward other users attending the same event. Besides, the more such fellow attendees, the higher her utility. The definition of overall social welfare is a simple extension of this intuition. Note that other definitions of social welfare, using other aggregate functions, are possible: we prefer to use a simple definition here that captures the intuition above. We formally define the Social Event Organization (SEO) problem as follows.

PROBLEM 1 (SOCIAL EVENT ORGANIZATION (SEO)). *Given a set U of users with a social graph $G = (U, E)$, a set A of events where each $a \in A$ has a minimum and maximum cardinality bound, denoted $\gamma_a \in \mathbb{N}$ and $\delta_a \in \mathbb{N}$ respectively ($\gamma_a \leq \delta_a$), innnate affinity function $\sigma(\cdot, \cdot)$, and social affinity function $w(\cdot, \cdot)$, produce a feasible assignment $M : U \to A$ that has the maximum overall social welfare $\omega(M)$, i.e., find*

$$M^* = \arg\max\{\omega(M) \mid M \textit{ is feasible}\}.$$

The number of scheduled events (i.e., those have at least one user assigned and cardinality constraints satisfied) is not fixed in advance, but rather determined by the solution, in which some events may receive zero users. However, the cardinality constraints for each event that has non-zero participants assigned must be respected. It is possible to have multiple groups performing the same event, e.g., there may be four groups of two participants each that are suggested the event of playing chess. This is easily handled by technically treating different instances of a given event as different events. We can consider weighting the two terms in Eq. (1), corresponding to the innate and social contributions to social welfare, differently. Our results in the next section show that the problem remains hard regardless of the weights chosen.

3.2 Hardness Results

Not surprisingly, our first result is that Social Event Organization is NP-hard. Our reduction is from the recently studied Seminar Assignment Problem (SAP). SAP is obtained by adding lower bound constraints to a restricted version of GAP where all objects have the same size, and is formally stated as follows. Given a set of n students and and a set of m seminars with maximum cardinality $B_1, B_2, \ldots, B_m \in \mathbb{N}$ and minimum cardinality $q_1, q_2, \ldots, q_m \in \mathbb{N}$, $q_i \leq B_i$, a profit $p_{i,j} \in \mathbb{N}$ of assigning student i to seminar j, we want to find an assignment of students to seminars that satisfies the cardinality constraints and maximizes the total profit. Krumke and Thielen [12] recently showed that this problem is NP-hard. We have the following easy result:

THEOREM 1. *The decision version of Social Event Organization is NP-complete.*

PROOF. Clearly, SAP is isomorphic to a restricted version of SEO where all innate affinities are natural numbers, all social affinities are zero (and $\alpha = 0$). The NP-hardness of SEO follows from this. Membership in NP is straightforward: given an assignment, both checking its feasibility and whether its social welfare exceeds a given threshold can be done in polynomial time. □

Next, we analyze how hard it is to approximate SEO. Our analysis involves two special cases of the problem, one with only innate affinities (which is isomorphic to the SAP problem) and the other with only social affinity.

Special Case 1: Innate Utility Only. In this case, we only consider instances of SEO where $w(u, v) = 0$, for all users u and v (and $\alpha = 0$). Thus, the problem is to find a feasible assignment $M : U \to A$ that maximizes the social welfare, which is solely determined by innate affinities. We call this version SEO-Innate.

THEOREM 2. *It is NP-hard to approximate SEO-Innate within a factor of* $(1 - 1/n + \epsilon)$ *(*$\forall \epsilon > 0$*) in polynomial time, where* n *is the number of users.*

PROOF. We prove the result by showing that if such an algorithm existed, it could solve the SUBSET-SUM problem, an NP-complete problem, in polynomial time. Since this is impossible unless P = NP, the theorem follows. Consider an arbitrary instance $\mathcal{I}$ of SUBSET-SUM consisting of a set of integers $T = \{t_1, t_2, \ldots, t_N\}$ and a target number τ. We must find out if there is a subset of T whose elements sum to exactly τ. Algorithm $\mathcal{B}$ first creates an instance $\mathcal{J}$ of the SEO problem with τ users and N events, and for each event $a \in [1, N]$, it sets $\gamma_a = \delta_a := t_a$. Also, $\sigma_{u,a} = 1$ for all $u \in U, a \in A$ and $w(u, v) = 0$ for all $u, v \in U$. Then $\mathcal{B}$ runs $\mathcal{A}$ on $\mathcal{J}$, and outputs YES (indicating there is a subset of T summing to exactly τ) if and only if $\mathcal{A}$ outputs an assignment with social welfare τ. We next prove that algorithm $\mathcal{B}$ can correctly distinguish between the YES- and NO-instances of SUBSET-SUM.

If $\mathcal{I}$ is a YES-instance, then by the above reduction, $OPT_{\mathcal{J}} = \tau$, i.e., the maximum possible social welfare of any feasible assingment on instance $\mathcal{J}$ is τ. By assumption, $\mathcal{A}$ will output an assignment with welfare $\geq (1 - 1/\tau + \epsilon)\tau > \tau - 1$. Since all utility values are 1, the output of $\mathcal{A}$ will be exactly τ, and thus $\mathcal{B}$ answers correctly: YES.

If $\mathcal{I}$ is a NO-instance, then for $\mathcal{J}$, by construction, not all users can be fit into an event, implying $OPT_{\mathcal{J}} < \tau$. Since the output of $\mathcal{A}$ is always $\leq OPT_{\mathcal{J}}$, it will be surely $< \tau$. Again, algorithm $\mathcal{B}$ answers correctly: NO. This was to be shown. □

We note that complementarily, Krumke and Thielen [12] show that it is NP-hard to approximate SAP within a factor of $1 - \epsilon_0/3$ even when all profits are in $\{0, 1\}$, where $\epsilon_0 > 0$ is a constant associated with the hardness gap of the 3-Bounded 3-Dimensional Matching problem [16]. As a consequence, they show that SAP does not admit PTAS (polynomial time approximation scheme). This hardness is clearly inherited by SEO-Innate as well. Note that both $1 - \epsilon_0/3$ and $(1 - 1/n)$ are close to 1 (the latter when n is large). This raises the question whether coarser approximations exist for SEO. Our result in the next section suggests it is unlikely.

Special Case 2: Pair-wise Affinity Only. In this case, we restrict attention to instances of SEO where $\sigma_{u,a} = 0$ for all users and all events (and $\alpha = 1$). We refer to this restricted version of SEO as SEO-Social. We show that this restricted problem is hard to approximate within any constant factor under hardness assumptions about certain combinatorial problems. Specifically, our result is achieved by a reduction from the DENSE k-SUBGRAPH problem. That problem, given a graph $G = (V, E)$ and a parameter k, asks to find a subgraph G' of G induced by k nodes such that average degree of nodes in G' is maximum. The average degree of a node in $G' = (V', E')$ is given by $2|E'|/|V'|$. By assuming the hardness of a conjecture known as Unique Games with Small Set Expansion Conjecture [11], Raghavendra and Streurer [18] show that DENSE k-SUBGRAPH is hard to approximate within any constant factor. We "lift" this to show a similar hardness of approximation of SEO-Social. For lack of space, we refer the reader to [18] for details and history of this conjecture.

THEOREM 3. *Assuming the Unique Games with Small Set Expansion Conjecture, it is NP-hard to approximate SEO-Social within any constant factor in polynomial time.*

PROOF. Given an instance $\mathcal{I}$ of DENSE k-SUBGRAPH defined by $G = (V, E)$ and a positive integer k, create an SEO instance $\mathcal{J}$ by setting $A = \{a\}$, $\gamma_a = \delta_a = k$, and $U = V$. I.e., there is just one event whose lower and upper bound on cardinality is k and there is a user corresponding to each node of G. For all $u \in U$, set $\sigma_{u,a} = 0$, and for all $u, v \in U$, $w(u, v) = 1$. By construction, $OPT_{\mathcal{I}} = OPT_{\mathcal{J}}$, i.e., the maximum social welfare of a feasible assignment on instance $\mathcal{J}$ is identical to the maximum average degree of a k-vertex induced subgraph of G in instance $\mathcal{I}$. Suppose there is a PTIME algorithm $\mathcal{A}$ approximating SEO within a factor of $c \in (0, 1)$. Then, $\mathcal{B}$ can approximate DENSE k-SUBGRAPH within the same factor by converting $\mathcal{I}$ to $\mathcal{J}$ as above, running $\mathcal{A}$ on $\mathcal{J}$, and outputting the nodes corresponding to the users chosen by $\mathcal{A}$ to attend event a. This is impossible unless the Unique Games with Small Set Expansion Conjecture doesn't hold. □

To conclude this section, not only is SEO NP-complete, it is also made up of two hard subproblems. Our results show that it is unlikely to be approximable within any constant factor in polynomial time, unless some hardness assumptions about important combinatorial problems break down. It should be clear from the proof of Theorems 2 and 3 that this is true regardless of the choice of α in Eq. (1).

4. PROPOSED SOLUTIONS

Given that SEO is NP-hard to even approximate, we propose a variety of heuristics that give emphasis to different aspects of the problem, such as the social affinity, the innate affinity, and the cardinality constraints. Before diving into the details of the algorithms, it is important to first understand what constitutes a feasible (valid) solution to the SEO problem.

Characterizing Feasible Solutions. For any event, if the event has reached its minimum cardinality, we refer to it as a *real event*; otherwise, we call it a *phantom event*. For example, a tennis or chess game with only one player is a phantom event, that will become real only after one other person is assigned to the event. An event is *open* and can accept more participants if the maximum cardinality has not been reached yet, otherwise it is declared *closed*. By definition, all phantom events are open. However, a real event can be either open or closed. A user u is *available* if u has not been assigned to a real event yet. A user-event assignment is valid if it involves an available user and an open event. During the assignment process, if the event to which a user was assigned becomes real, then the user's assignment is fixed, and she is marked *unavailable*. Consequently, all other previous and future assignments involving that particular user are deemed invalid. Recall that the final solution must be feasible, i.e., respect cardinality constraints. As shown in §3, in some instances it is not possible to find an assignment for every user, so the assignment function may be partial. In practice, we can deal with this by supposing that there are sufficiently many events with large capacity such as movies or theatrical shows to which users left unassigned by the algorithm may be assigned.

Notice that each assignment decision impacts the utility received from future assignments, due to the coupling effect with other users in the form of social affinity. Therefore, it is non-trivial to achieve a (feasible) solution to SEO that achieves a high social welfare.

Solution Template. In the remainder of this section we discuss a variety of algorithms. At a high level, each algorithm (except *Random* and NRMP+) follows the following template. First, a sorted list $\mathcal{L}$ of potential assignments of users to events is generated. The additional contents of the list may vary depending on the algorithm. This list may be updated and re-ordered several times during a run of the algorithm. Second, users are assigned to events by making one pass on this list, and the "state" of users (available, unavailable), and events (phantom, real open, real closed) is updated appropriately. In particular, it marks an event a as "real" when it reaches its minimum cardinality γ_a, and "closed" when it reaches its maximum cardinality δ_a. These assignments are tentative and are recorded by membership in the "S" sets, e.g., $u \in S_a$ means u is tentatively assigned to event a, and possibly to other events. All events are "open" and "phantom" to start with. Once an event a becomes real, the users in the set S_a are marked "unavailable" for future assignments and we set $M(u) = a, \forall u \in S_a$. Additionally, the algorithm invalidates any previous and future assignments of each user $u \in M^{-1}(a)$, by cleaning up other "S" sets as needed. Finally, the algorithm performs post-processing to ensure that the output is a feasible solution, i.e., no phantom events are left behind.

4.1 Baselines

In this section, we present a set of intuitive baseline solutions, in increasing level of sophistication. We build on these baselines and present our proposed algorithms in §4.2.

Random. Our first baseline is called *Random*, which randomly assigns users to events while respecting the cardinality constraints. Specifically, the algorithm first randomly shuffles the list $\mathcal{L}$ containing user-event pairs $\langle u, a\rangle$. It then traverses the list and at each tuple $\langle u, a\rangle$ assigns user u to event a if u is available and a is open. It appropriately marks an event as real when it reaches its minimum cardinality constraint. Clearly, this is a "straw man" approach, which does not take into account the "utility" of any assignment.

Algorithm 1 Dynamic Greedy $(U, A, \sigma, w, \alpha)$

$M(u) \leftarrow \perp, \forall u \in U$; $S_a \leftarrow \emptyset, \forall a \in A$
for all $(u, a) \in U \times A$ s.t. $\sigma_{u,a} > 0$ **do**
  $g(u, a \mid S_a) \leftarrow (1-\alpha)\sigma_{u,a}$
  $\mathcal{L}.\mathsf{insert}(\langle u, a, g(u, a \mid S_a)\rangle)$
while $\langle u, a, g(u, a \mid S_a)\rangle \leftarrow \mathcal{L}.\mathsf{pop}()$ **do**
  if $M(u) = \perp$ and $|S_a| < \delta_a$ **then**
    $S_a \leftarrow S_a \cup \{u\}$
    if $|S_a| > \gamma_a$ **then**
      $M(u) \leftarrow a$
      for all $a' \in A$ s.t. $a' \neq a \wedge u \in S_{a'}$ **do**
        $S_{a'} \leftarrow S_{a'} - \{u\}$
    if $|S_a| = \gamma_a$ **then**
      for all $v \in S_a$ **do**
        $M(v) \leftarrow a$
        for all $a' \in A$ s.t. $a' \neq a \wedge v \in S_{a'}$ **do**
          $S_{a'} \leftarrow S_{a'} - \{v\}$
    for all v : $(w(u, v) > 0 \wedge M(v) = \perp)$ **do**
      $\mathcal{L}.\mathsf{update}(\langle v, a, g(v, a \mid S_a)\rangle)$ // Using Eq. (3)
Reassign available users $\{u \mid M(u) = \perp\}$
return M

NRMP+. Given the connection between SEO and NRMP, a natural question is whether we can leverage algorithms developed for NRMP. We next describe an adaptation of the NRMP algorithm [8] for handling the lower bound constraints present in SEO. In this adaptation, called NRMP+, we cast SEO in the NRMP framework by associating a preference list of events for each user, ordered by the innate affinity, and a preference list of users for each event, ordered by the innate affinity.[3]

NRMP+ repeats the following procedure until each user is either "accepted" by an event or is "rejected" by all events she applied to. Initially, all users are unassigned and no acceptance or rejection has been made yet. In each round, each unassigned user "applies" to the most preferred event that has not rejected her. Each event a then picks the δ_a most preferred users who applied to it, or all applicants if there are fewer than δ_a of them, and puts them on a waiting list. Users who failed to get on the waiting list are rejected and are thus unassigned after this round. After this iterative process stops, we check if there exists any phantom event. If yes, redistribute users assigned to phantom events to existing real open events in a greedy manner, based on innate affinity. After that, either no user is left unassigned or there are no real events to which unassigned users can be assigned, and we stop.

It is unclear how NRMP can be directly adapted to take social affinity into account in deciding acceptances. In Section 5, we empirically compare NRMP+ with the more sophisticated baselines we develop next, as well as with our main algorithm.

Static Pairwise Greedy. Our next baseline, called Static Pairwise Greedy (SG), assigns pairs of users to events, taking into account both innate event and pairwise social affinities. Specifically, a sorted list $\mathcal{L}$ containing tuples $\langle u, v, a\rangle$ representing potential assignment of pairs of users to events is generated. The list is ordered by non-increasing potential gain (of "utility"), defined as

$$g((u,v),a) = (1-\alpha)(\sigma_{u,a} + \sigma_{v,a}) + 2\,\alpha\, w(u,v).$$

The list is traversed and at each tuple $\langle u, v, a\rangle$ users u and v are assigned to event a if both users are available, and a is open with at least two spots. User and event states are appropriately updated.

[3] Events don't have independent preference lists: we induce them from user-event affinities for the sake of simulation of NRMP.

Finally, any users that are in phantom events at the end of one pass over the list $\mathcal{L}$ are redistributed greedily among remaining open events. We call this a *static* approach as the function g is computed only once. Therefore, the partial ordering of the $\mathcal{L}$ is static, and it only incurs deletions (with some assignments being invalidated). Therefore, the assignment of a pair of users to an event is oblivious to which users are already assigned to that event.

Dynamic Greedy. Our next algorithm, called Dynamic Greedy (DG), is more sophisticated. It updates its estimation of the "utility" a user will attain from an assignment, based on the current set of assignments. To achieve this, the algorithm updates the list $\mathcal{L}$ at each assignment, where $\mathcal{L}$ is composed of tuples $\langle u, a, g(u, a|S_a)\rangle$, $S_a = \emptyset$ initially, and the potential gain g is

$$g(u, a \mid S_a) = (1-\alpha)\sigma_{u,a} + \alpha \sum\nolimits_{v \in S_a} w(u, v). \qquad (3)$$

Clearly, when no users are assigned to an event a, $g(u, a \mid \emptyset) = (1-\alpha)\sigma_{u,a}$. Algorithm 1 presents the pseudocode for the DG method. At each step, the assignment that gives the largest potential gain $g(.)$ is performed. Let N_u be the set of users v: $w_{u,v} > 0$. Now, whenever a user u is assigned to an event a, we need to recompute the potential gain $g(v, a \mid S_a)$ for all neighbors $v \in N_u$ of user u in the network, since their utility for event a increases after the assignment of u to that event. In practice, the algorithm maintains the list $\mathcal{L}$ as a priority queue of tuples in non-increasing order of $g(u, a \mid S_a)$. We use the notation $M(u) = \perp$ to denote that the user u is available. We remark that the update operation (line 18) is in charge of not only updating the existing entries in the priority queue but also checking whether the pair (v, a) is already in the heap and if not, adding it.

The post-processing in Line 19 is done in a greedy manner. Let A' be the set of currently open real events, i.e., $A' = \{a \mid \gamma_a \le |S_a| < \delta_a\}$ and U' be the set of users assigned to phantom events. The list $\mathcal{L}$ is re-constructed with users from U' and events from A', and we rerun the DG algorithm over this list. This process can be repeated as needed. The post-processing stops when either all users are assigned or there are not enough real events (i.e., all real events are closed). In the latter case, from a practical perspective, the remaining users may be assigned to new events with a large capacity, e.g., watching a movie or a theatrical show.

Implementation Notes and Time and Space Complexity. We remark that in our implementation, we optimize the algorithm for better performance. In particular, we tune our implementation to suit a Fibonacci heap based realization of the priority queue. Specifically, we commit to making an assignment for new users right after line 7. This allows us to avoid assigning users to multiple phantom events. Thus, when the priority queue is updated, "decrease key" operation will not be needed, permitting us to implement the priority queue using a Fibonacci heap. We note, however, in this case the assignment function M no longer corresponds to only real events since some phantom events may be left behind by the algorithm at termination. For ease of exposition, we have chosen to present our algorithm in a conceptually simpler way.

When the priority queue $\mathcal{L}$ is implemented using a Fibonacci heap, then for each assignment (u, a), the algorithms performs $O(|N_u|)$ update operations (Line 18), each of which is constant time. Summing over all users, the total running time of all update operations is $O(|E|)$, where E is the edge set of the social network of users, or more precisely the set of pairs (u, v) with $w(u, v) > 0$. Next, since the number of all possible (u, a) pairs is bounded by $|\mathcal{L}|$, we need at most that many pop operations, which in total takes $O(|\mathcal{L}| \log(|\mathcal{L}|))$ time. In addition, insertion is constant time for Fibonacci heaps so the initialization step takes $O(|\mathcal{L}|)$. Hence, the total time complexity for Dynamic Greedy is $O(|\mathcal{L}| \log(|\mathcal{L}|) + |E|)$. Finally, the space complexity is $\Omega(|\mathcal{L}| + |E|)$.

Algorithm 2 PCADG $(U, A, \sigma, w, \alpha)$

1: $M(u) \leftarrow \perp, \forall u \in U$; $S_a \leftarrow \emptyset, \forall a \in A$; $\mathcal{P} \leftarrow \emptyset$
2: deficit $\leftarrow 0$; $V \leftarrow U$
3: **for** each $(u, a) \in U \times A$ s.t. $\sigma_{u,a} > 0$ **do**
4: $\quad g(u, a \mid S_a) \leftarrow (1-\alpha)\sigma_{u,a}$
5: $\quad \mathcal{L}$.insert$(\langle u, a, g(u, a \mid S_a)\rangle)$
6: **while** $\langle u, a, g(u, a \mid S_a)\rangle \leftarrow \mathcal{L}$.pop() **do**
7: $\quad$ **if** $M(u) = \perp$ and $|S_a| < \delta_a$ **then**
8: $\quad\quad$ **if** $|S_a| = 0$ **then**
9: $\quad\quad\quad$ **if** deficit $+ \gamma_a \le |V|$ **then**
10: $\quad\quad\quad\quad$ deficit $\leftarrow$ deficit $+ \gamma_a - 1$
11: $\quad\quad\quad\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{a\}$
12: $\quad\quad\quad$ **else**
13: $\quad\quad\quad\quad$ **continue**
14: $\quad\quad$ **else**
15: $\quad\quad\quad$ **if** $a \in \mathcal{P}$ **then**
16: $\quad\quad\quad\quad$ deficit $\leftarrow$ deficit $- 1$
17: $\quad\quad S_a \leftarrow S_a \cup \{u\}$
18: $\quad\quad V \leftarrow V \setminus \{u\}$
19: $\quad\quad$ **if** $|S_a| > \gamma_a$ **then**
20: $\quad\quad\quad M(u) \leftarrow a$
21: $\quad\quad\quad$ **for all** $a' \in A$ s.t. $a' \neq a \wedge u \in S_{a'}$ **do**
22: $\quad\quad\quad\quad S_{a'} \leftarrow S_{a'} - \{u\}$
23: $\quad\quad$ **if** $|S_a| = \gamma_a$ **then**
24: $\quad\quad\quad \mathcal{P} \leftarrow \mathcal{P} \setminus \{a\}$;
25: $\quad\quad\quad$ **for all** $v \in S_a$ **do**
26: $\quad\quad\quad\quad M(v) \leftarrow a$
27: $\quad\quad\quad\quad$ **for all** $a' \in A$ s.t. $a' \neq a \wedge v \in S_{a'}$ **do**
28: $\quad\quad\quad\quad\quad S_{a'} \leftarrow S_{a'} - \{v\}$
29: $\quad\quad$ **for all** v : $(w(u, v) > 0 \wedge M(v) = \perp)$ **do**
30: $\quad\quad\quad \mathcal{L}$.update$(\langle v, a, g(v, a \mid S_a)\rangle)$ // Using Eq. (4)
31: **return** M

4.2 Greedy with Look Ahead Estimation

The Dynamic Greedy algorithm is intuitive: it tries to make assignments based on the largest estimated gain, given the current state of the assignment. However, it has its limitations: (i) it is oblivious to which events a user's friends are likely to be assigned; (ii) the first pass may result in a large number of phantom events as it is ignorant of which events are likely to materialize. To address these limitations, we propose the *Phantom- and Community-Aware Dynamic Greedy or PCADG* (pseudocode in Algorithm 2). A key feature of this algorithm is that it looks ahead at the forthcoming assignments to inform the decision at each step.

Community Awareness. The earlier algorithms do not take into account the potential gain that future assignments of friends to an event may bring to a user. In our Phantom- and Community-Aware Dynamic Greedy algorithm, the awareness of friends' potential assignments is incorporated into the decision making process. More specifically, here, the potential gain g is defined as

$$\begin{aligned} g(u, a|S_a) =& (1-\alpha)\sigma_{u,a} + \alpha \sum\nolimits_{v \in S_a} w(u, v) \\ &+ \alpha(\delta_a - |S_a|) \sum\nolimits_{v \in V} w(u, v)/(|V| - 1), \end{aligned} \qquad (4)$$

where V is the set of users that have not yet been assigned to any event, phantom or real: $V = \{u \mid u \in U \wedge M(u) = \perp \wedge u \notin \bigcup_{a \in A} S_a\}$.

Notice that the first two terms in this equation are identical to the two terms in Equation 3 of the Dynamic Greedy method. The third term corresponds to the "look-ahead" whereby it is optimistically assumed that the remaining spots in the event a will be filled with friends of u, with an average social affinity toward u. The average is computed over users in V. While the first two terms are based on the current membership of S_a, the third term is based on anticipation. Thus, this algorithm gives equal importance to both innate affinity and social affinity in gauging the potential gain that could come from an assignment. Other variants can be easily designed to weigh the importance of the innate and social affinities differently.

Phantom Awareness. Another observation about Dynamic Greedy is that it tends to be somewhat indiscriminate in the creation of phantom events: it never restricts the creation of new events as the condition $|S_a| < \delta_a$ (line 6 of Algorithm 1) will always be true for new events, for which $|S_a| = 0$ by definition. All phantom events that are left behind after a pass over the list $\mathcal{L}$ have to be cleaned up later. PCADG tries to limit the creation of phantom events, by employing a stronger condition for phantom event creation. Intuitively, it keeps track of phantom events as it proceeds, and procrastinates on creating new events if the number of unassigned users is less than the currently unfilled spots in existing phantom events that need to be filled in order for the events to materialize.

To achieve this, PCADG maintains a variable called `deficit`, defined to be

$$\texttt{deficit} = \sum_{a \in \mathcal{P}} (\gamma_a - |S_a|),$$

where $\mathcal{P}$ is the set of phantom events and S_a is the set of users assigned to event a. If, at any point of the execution of PCADG, it happens that $\texttt{deficit} > |V|$, then there must be at least one phantom event at the end of one pass over $\mathcal{L}$, thus requiring reassignment. Therefore, the algorithm performs an assignment based on a tuple $\langle u, a, g(.)\rangle$ *only if* it does *not* push `deficit` over $|V|$ (Line 9). Otherwise, it will skip that tuple and continue to the next one in $\mathcal{L}$. As can be seen, by using `deficit` and V, the algorithm does limit the creation of new phantom events.

We now give more details about the PCADG algorithm. At the beginning, when $\mathcal{P} = \emptyset$, $\texttt{deficit} = 0$ and $V = U$. The size of V is non-increasing throughout the algorithm. For any assignment $\langle u, a, g(u, a \mid S_a)\rangle$ that consists of an available user u and open event a, if a is a new event with no users, i.e., $|S_a| = 0$, we increment `deficit` by $\gamma_a - 1$ (because a is a phantom event and it would take $\gamma_a - 1$ more users to make a real). Otherwise, we decrease `deficit` by 1 (since u removes one "unit" of `deficit`). Furthermore, u is removed from V, if $u \in V$ before this assignment was made.

It is worth noting that PCADG may consider user-event combinations for which the innate affinity may be zero, as this assignment may potentially result in a large gain owing to the social affinity with other users assigned to this event, as per Equation 4. This is in keeping with our stated objective of maximizing social welfare. Therefore, it is possible that some users may be assigned to low innate affinity events as a result. We measure this tradeoff empirically in §5 by measuring the "regret ratio" (defined in §5) of the assignment returned by the various algorithms.

In order to explicitly study the impact of community and phantom awareness, we will compare PCADG with a lesser variant called *Phantom-Aware Dynamic Greedy (PADG)* that only takes into account phantom awareness. As such, the PADG algorithm is identical to Algorithm 2, with the one change that the potential gain is defined using Eq. (3) instead of Equation 4. In §5, we compare the running times and scalability of different proposed algorithms, in addition to comparing them on their quality.

We close this section by noting that the time and space complexity of the PCADG algorithm is the same as that of DG (and PADG) except that the size of the list $\mathcal{L}$ may potentially become $O(|U| \times |A|)$ in the worst case.

5. EXPERIMENTAL ANALYSIS

We evaluate our Phantom- and Community-Aware Dynamic Greedy (PCADG) method against the Phantom-Aware variant (PADG) and the baselines described in §4.1, i.e., Random, NRMP+, Static Greedy (SG) and Dynamic Greedy (DG).

Evaluation Metrics. The most natural evaluation metric is the *total social welfare*, $\omega(M)$ defined in Eq. ((2)). In addition, since optimal or near-optimal solution cannot be found in polynomial time, we measure the effectiveness of each algorithm "indirectly" by comparing the utility a user gets against a *coarse upper bound*, which is the maximum utility she could have enjoyed by going to her favorite event with her best friends. Formally, we define the *regret ratio* for a user u, denoted $\rho(u)$, to be

$$\rho(u) = 1 - \frac{(1-\alpha)\sigma_{u,M(u)} + \alpha \sum_{v \in S_{M(u)}} w(u,v)}{\max_{a \in A}((1-\alpha)\sigma_{u,a} + \alpha \sum_{v \in B_u} w(u,v))}, \quad (5)$$

where M is the assignment made by an algorithm, and $B_u \subseteq N_u$ is the set of top-$K_{u,a}$ friends of u in terms of social affinity $w(u,v)$, and $K_{u,a} = \min\{|N_u|, \delta_a - 1\}$.

Ideally an algorithm that performs well w.r.t. this metric should assign users to events in such a way that many users experience low regret in the assignment made. Note that $\sum_{u \in U} \max_{a \in A}(\sigma_{u,a} + \sum_{v \in B_u} w(u,v))$ is guaranteed to be greater than the optimal social welfare, except when the favorite event of all users is the *same*, in which case the optimal welfare may reach this quantity. Moreover, ρ is a very pessimistic ratio, as the denominator is an upper bound on the maximum possible utility that the user can achieve. Also, it is worth pointing out that our algorithms are not designed to directly minimize individual regrets of users, but rather social welfare that is the overall utility enjoyed by all users. The purpose here is to gain some insights about to what extent our algorithms would lead to a compromise w.r.t. regret.

We set α to 0.5 in Eq. (1) and (5) for all experiments unless otherwise mentioned. We also conduct a detailed experiment on real data, where we vary α and study its effect, illustrated in Figure 5.

5.1 Experiments on Synthetic Datasets

To better understand the impact of various parameters on our proposed algorithms, we first perform rigorous evaluations on synthetically generated data. We also conduct scalability tests on large synthetic data to show that our algorithms are scalable and efficient.

For all events $a \in A$, δ_a is sampled from a normal distribution with mean 20 and variance 10, and given that, γ_a is sampled uniformly at random from the interval $[1, \delta_a]$. The social network $G = (U, E)$ is generated by a random power-law graph model with a power-law exponent of 1.5 [1]. The data generated has 500 users and 50 events. A user is interested in an event with probability 0.05, and thus the average number of events he is interested is 2.5. For each $(u,v) \in E$, the social affinity value is sampled from a normal distribution with mean 1.5 and variance 3, as is innate affinity $\sigma_{u,a}$, for each (u,a) pair in the user-event relation. All values less than 10^{-5} are set to be zero: all affinity values are thus non-negative. This gives the default setting for the parameters used to generate the data, however, throughout this section, we vary these parameters to understand their effect on the performance of

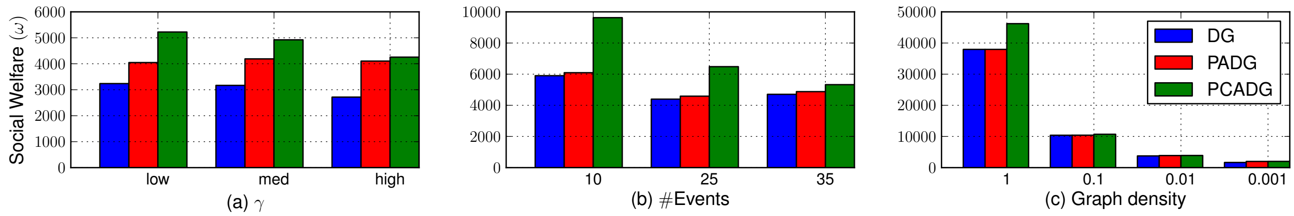

Figure 1: Effects of parameter choices on the performance of PADG and PCADG

Algorithm	$\|U\|$	Number of Events				
		50	100	150	200	500
PADG	1K	0.085	0.085	0.108	0.152	0.187
	10K	0.343	0.435	0.572	1.54	1.6
	100K	2.75	4.61	6.63	15.0	27.1
PCADG	1K	0.315	0.759	1.46	3.17	9.05
	10K	2.09	10.07	14.47	32.1	77.2
	100K	22.10	64.50	166	207	1085

Table 1: Running time (in seconds) of PADG and PCADG

	Random	NRMP+	SG	DG	PADG	PCADG
Time (s)	0.98	0.14	12.7	0.31	0.43	10.1

Table 2: Running time (in seconds) of all algorithms with 10K users and 100 events

our three dynamic greedy methods, DG, PADG and PCADG. Figure 1 shows the social welfare achieved by these algorithms with three types of comparisons: minimum cardinality constraints, total number of events, and density of the social network graph.

Effect of Minimum Cardinality. We alter minimum cardinality constraint (γ) and evaluate its effect on PADG and PCADG. We keep the maximum cardinality constraint δ_a's unchanged from the basic setting for each event $a \in A$, and vary their γ_as. We generate three groups of data, by sampling the minimum cardinality constraint γ_a uniformly at random from $[\delta_a/2, \delta_a]$ (low mean), $[\delta_a - \delta_a/4, \delta_a]$ (medium mean), and $[\delta_a - \delta_a/8, \delta_a]$ (high mean) respectively. As seen in Figure 1(a), there is a noticeable improvement in social welfare for PADG and PCADG over DG (which does not account for phantom events). As γ increases, more phantom events may be created, which increases the gain of PADG.

Effect of the Number of Events. We vary the number of events as 10, 25 and 35, for a 500-user dataset. From Figure 1(b), as the number of events increases, each event needs to accommodate fewer users and thus the effect of social affinity decreases. With fewer events, PCADG is at an advantage over PADG and DG, as it is community aware and takes into account social affinity values by looking ahead.

Effect of Graph Density. Next, we test various graph densities of the underlying social network graph by generating four different graphs using Erdős-Rényi model $G(500, p)$ where the edge existence probability being p = 0.001, 0.01, 0.1, and 1 respectively. For denser graphs, more social affinity values are present and hence we observe that the margin between PCADG and the other two is higher in comparison with sparse graphs (Figure 1(c)).

Running Time. Finally, we evaluate the scalability of the algorithms by varying the number of users (1K, 10K and 100K) and events (50, 100, 150, 200, and 500). We maintain the default settings for all other parameters. Table 1 shows the running time of PADG and PCADG. Overall, both algorithms achieve good efficiency, finishing within seconds or minutes in all cases, and PADG always runs faster than PCADG. As the trend is similar, we show the running time comparison with Random, NRMP+, SG, and DG for one case of 10K users and 100 events. Random, DG, and SG take 0.98, 0.31, and 12.7 seconds to finish, while PADG takes 0.43 and PCADG takes 10.07 seconds respectively (see Table 2). SG was consistently the slowest algorithm in all cases that we tested. For our largest simulation of 100K users and 500 events, PCADG takes 20 mins. This is a reasonable for large scale event organization, where real-time response is not critical.

5.2 Experiments on Real Event Datasets

5.2.1 Data Description and Preprocessing

We evaluate our algorithms on the following three real datasets. *Plancast* and *Meetup* are local event organization websites that allow groups of users to interact and plan events. We use the data[4] released by [15]. Since in Plancast and Meetup users and events are from all over the world, we select several big cities and project the data to such locations. The third dataset is *SIGCOMM2009* (referred to as SIG), collected by Pietilainen et al. [17] (details later). The following pre-processing was performed on the datasets to match our problem setting.

Plancast. We extract two subsets from the Plancast dataset in [15] corresponding to users and events in the vicinity of Chicago (CHI) and Vancouver (VAN). The two datasets CHI and VAN have 2338 and 2327 users, and 339 and 360 events, respectively. For computing the innate affinity, we set $\sigma_{u,a}$ to 1 for events located within 0.01 units of Euclidian distance from the user's geolocation. Plancast allows users to subscribe to each other and receive updates on friends' activities, which gives us a ground-truth social network. We compute the Katz distance [10] for each adjacent pair of users in the subscription graph to learn social affinities $w(u, v)$. By definition of Katz, we have $w(u,v) := \sum_{\ell=1}^{\infty} \beta^{\ell} \cdot |P_{u,v}^{(\ell)}|$, where $P_{u,v}^{(\ell)}$ is the set of paths of length ℓ between u and v in the graph, and β is a damping factor so that the measure counts short path more heavily, and is set to 0.01 in our case. Higher Katz score intuitively implies that two nodes are more connected, and hence gives a higher social affinity value.

Meetup. Similarly, we project Meetup data from [15] on events and users located in San Francisco and New York City and refer to these datasets as SFO and NYC. SFO contains 6438 users and 59 events, while NYC contains 10328 users and 127 events. Meetup allows users to create and participate in groups, which in turn organize events. For both locations, we consider the labeled heterogeneous tripartite graph between users, groups, and events, with edges labeled by the relation that connects the pairs of nodes: e.g., if user u is a member of group g that organized event a which the user attended, we draw edges (u, g) and (g, a). We then define $\sigma_{u,a}$ as the Katz distance between the user node u and event node a in

[4]http://www.largenetwork.org/ebsn

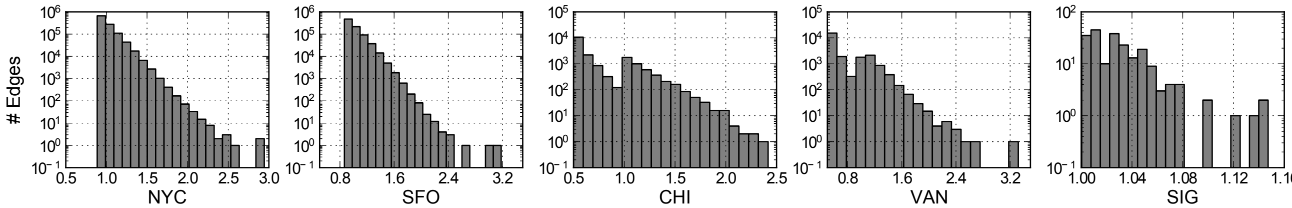

Figure 2: Distributions of social affinity values

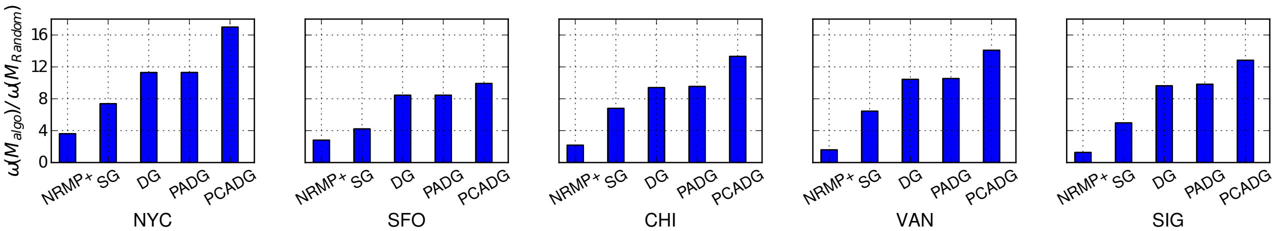

Figure 3: Improvement of social welfare yielded by various algorithms over Random

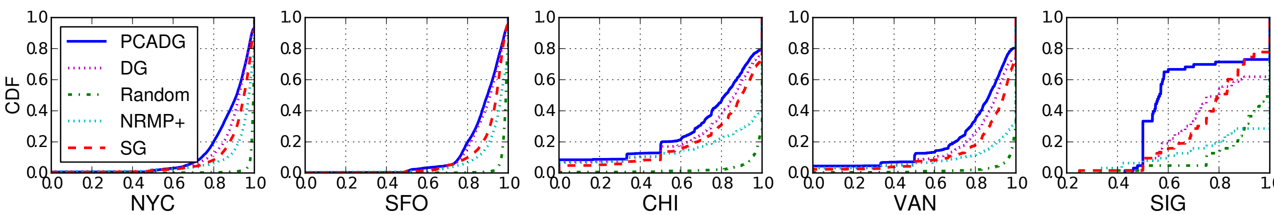

Figure 4: CDF showing regret ratio ρ for different algorithms

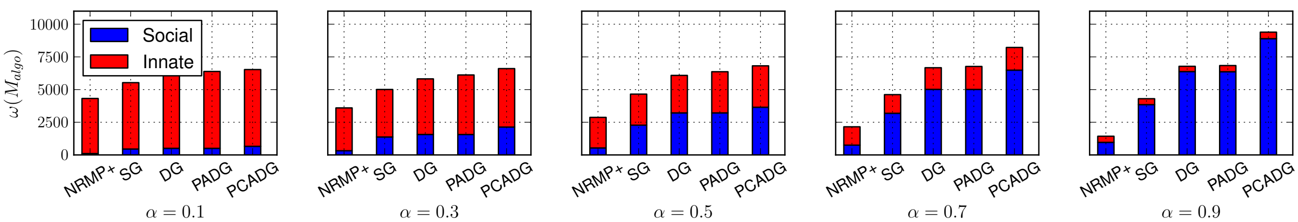

Figure 5: Dominance of PCADG is preserved on varying contributions of social and innate affinities (VAN)

the heterogeneous graph, with higher Katz score indicating higher affinity. For social affinity between users, we utilize tags provided in the data. More specifically, users on Meetup can attach tags in their profiles to indicate preferences and interests. Intuitively, users sharing common interests may enjoy being with each other in activities they both like. Thus, for each $(u, v) \in E$, we compute $w(u, v)$ as the Jaccard similarity coefficient of u's tag sets and v's tag sets: $w(u, v) := \frac{|T_u \cap T_v|}{|T_u \cup T_v|}$, where T_u and T_v is are sets of tags of u and v respectively. Note that users without any tags in their profile are filtered out.

SIGCOMM2009. This is a small dataset with 76 users, and 11 events, collected by a mobile application that records bluetooth device discovery information. Users' innate utility for events is 1 if a user attends an event as indicated by the data, and 0 otherwise. The data includes ground-truth Facebook friendship information, and thus as in the case of Plancast, we take this underlying social network and compute the Katz scores for social affinities.

Finally, for all five datasets described above, the user-event innate affinity is set to 1 if the user attends the event as indicated by the data, and to 0 otherwise. Also due to lack of ground-truth data, we generate the maximum and minimum cardinality constraints on the events as in the basic synthetic case, i.e., let r be the ratio between the number of users and the number of events; then for all $a \in A$, maximum cardinality constraint δ_a is sampled from a normal distribution with mean $2r$ and variance r, while minimum cardinality γ_a is sampled uniformly at random from $[1, \delta_a]$. Figure 2 shows the distributions of social affinity values for all datasets.

5.2.2 *Evaluations and Analysis*

Social Welfare. Figure 3 shows the social welfare (Eq. (2)) obtained by the various methods considered relative to the welfare obtained using a Random assignment. It shows the ratio between the social welfare yielded by assignments made by NRMP+, SG, DG, PADG, PCADG and that by Random. PCADG consistently has the highest social welfare, followed by PADG, DG, SG and NRMP+. The margin between PCADG and the rest of the algorithms is significant in all cases, indicting its overall effectiveness. The biggest lead of PCADG is observed in NYC, 17 times better than Random, while the smallest happens in SIG, in which it is still 4 times better. On all datasets PADG and DG have similar values, with PADG outperforming only slightly. This is because the generated minimum cardinality constraints for events are much smaller than maximum cardinality constraints. Recall from Figure 1(a) that when γs are high, the gap between PADG and DG is also high.

Regret Ratio. Figure 4 shows the cumulative distribution function (CDF) of the regret ratio $\rho(u)$ (see Eq. (5)) achieved by the various methods (PADG skipped for clarity) on the five datasets. The general trend is similar to that of social welfare, where PCADG outperforms all baselines. For instance, in NYC, 20% of the users have a regret below 0.8 using PCADG, while only 8% have that regret with NRMP+. This gap is even more pronounced in CHI and VAN, with the largest difference in SIG. In particular, 70% of users have a regret ratio below 0.7 using PCADG, while only 18% have that regret using NRMP+.

Varying Contribution of Social and Innate Affinities. A possible concern about our definition of social welfare (Eq. (1)) is that it may be dominated by social affinity. However, Eq. (1) allows the organizer to control the relative importance of social and innate affinities and not let the overall objective be undesirably dominated by either. To analyze this, in this final experiment, we first boost the innate affinities such that at $\alpha = 0.5$, innate and social affinities (found by PADG) are nearly equal, and stress-test the various algorithms on VAN by varying α. This allows us to vary the relative contribution of social and innate affinities to the overall welfare achieved and study its impact on the algorithms. Figure 5 shows the absolute value of the total welfare and the individual contribution of the two affinities for $\alpha = 0.1, 0.3, 0.5, 0.7$ and 0.9. As can be seen, the dominance of PCADG over other algorithms is maintained across all values of α, with higher overall welfare for larger α. We also measured regret ratios in all cases. The trend was found to be the same as in Figure 4 and the plot is omitted for brevity.

In summary, through extensive experiments on both synthetic and real world data, we have demonstrated the effectiveness (in terms of social welfare and regret) and efficiency (in terms of running time) of our proposed solutions to the SEO problem.

6. DISCUSSION AND CONCLUSIONS

While there has been considerable work on detecting emerging events from social media, the related, equally important area of organizing events has received much less research attention. In this paper, motivated by popular social event organization platforms like Meetup and Plancast and by applications such as organizing events using these platforms over a wide local area, and organizing events co-located with large conferences and conventions, we study the novel research direction of Social Event Organization (SEO).

Our focus in this work has been on the assignment problem, where we consider two critical factors: *innate affinity*, which captures a user's intrinsic preferences for the events being offered, and *social affinity*, which captures the social connections among the users themselves. We formally define the overall measure of *social welfare* in event organization and formalize the SEO problem as the problem of maximizing social welfare subject to constraints on the participation cardinality of the events. We show that this problem is not only NP-hard, but also hard to approximate. We propose a set of heuristic solutions that leverage the notion of *phantom events* and the technique of *looking ahead* the potential gain that may accrue as a result of future assignments to the current event for which a user is being assigned. Using an extensive set of experiments over synthetic datasets and three real datasets including those from the platforms Meetup and Plancast, we demonstrate that our heuristic algorithms are scalable and furthermore outperform the baseline algorithms significantly in terms of social welfare and regret ratio.

Several interesting questions remain. Innate and social affinities may be interdependent. E.g., Al and Ed are close friends but don't like to play poker together since they know each other's tells, while Al may like going to a hockey game (for which she has low innate affinity) with Bo who is knowledgeable and keeps her engaged. Handling such interdependency is important. Predicting which events a user will attend in the future is also an interesting problem. It is also worth comparing the assignments made by SEO to ground-truth from event participations, which may be challenging to obtain. Also, a systematic study of alternative ways of aggregating innate and social affinities for defining social welfare is worthwhile.

7. REFERENCES

[1] W. Aiello, F. R. K. Chung, and L. Lu. A random graph model for massive graphs. In *STOC*, pages 171–180, 2000.

[2] S. Amer-Yahia, S. B. Roy, A. Chawla, G. Das, and C. Yu. Group recommendation: Semantics and efficiency. *PVLDB*, 2(1), 2009.

[3] A. Anagnostopoulos et al. Online team formation in social networks. In *WWW*, 2012.

[4] N. Armenatzoglou, S. Papadopoulos, and D. Papadias. A general framework for geo-social query processing. *PVLDB*, 6(10), 2013.

[5] P. Biró et al. The college admissions problem with lower and common quotas. *Theor. Comput. Sci*, 411(34-36):3136–3153, 2010.

[6] C. Chekuri and S. Khanna. A ptas for the multiple knapsack problem. In *SODA*, 2000.

[7] L. Fleischer et al. Tight approximation algorithms for maximum general assignment problems. In *SODA*, 2006.

[8] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *Amer. Math. Monthly*, 69, 1962.

[9] K. Hamada, K. Iwama, and S.Miyazaki. The hospitals/ residents problem with quota lower bounds. In *MATCH-UP: Matching Under Preferences Workshop at ICALP*, 2008.

[10] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1), 1953.

[11] S. Khot. On the power of unique 2-prover 1-round games. STOC, 2002.

[12] S. O. Krumke and C. Thielen. The generalized assignment problem with minimum quantities. *European Journal of Operational Research*, 2013.

[13] T. Lappas, K. Liu, and E. Terzi. Finding a team of experts in social networks. In *KDD*, 2009.

[14] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7), 2007.

[15] X. Liu et al. Event-based social networks: linking the online and offline social worlds. In *KDD*, 2012.

[16] E. Petrank. The hardness of approximations : Gap location. *Computational Complexity*, 4:133–157, 1994.

[17] A.-K. Pietilainen and C. Diot. CRAWDAD data set thlab/sigcomm2009 (v. 2012-07-15). Downloaded from http://crawdad.cs.dartmouth.edu/thlab/sigcomm2009, 2012.

[18] P. Raghavendra and D. Steurer. Graph expansion and the unique games conjecture. In *STOC*, 2010.

[19] A. E. Roth. The evolution of the labor market for medical interns and residents: a case study in game theory. *Journal of Political Economy*, 6(4):991–1016, 1984.

[20] D. B. Shmoys et al. An approximation algorithm for the generalized assignment problem. *Math. Program.*, 62:461–474, 1993.

[21] M. Sozio and A. Gionis. The community-search problem and how to plan a successful cocktail party. In *KDD*, pages 939–948, 2010.