The VLDB Journal (2019) 28:221–241
https://doi.org/10.1007/s00778-018-0526-5

REGULAR PAPER

A unified agent-based framework for constrained graph partitioning

Lefteris Ntaflos[1] · George Trimponias[2] · Dimitris Papadias[1]

Received: 13 November 2017 / Revised: 18 July 2018 / Accepted: 11 October 2018 / Published online: 27 October 2018

Abstract
Social networks offer various services such as recommendations of social events, or delivery of targeted advertising material to certain users. In this work, we focus on a specific type of services modeled as *constrained graph partitioning* (CGP). CGP assigns users of a social network to a set of classes with bounded capacities so that the *similarity* and the *social* costs are minimized. The similarity cost is proportional to the dissimilarity between a user and his class, whereas the social cost is measured in terms of friends that are assigned to different classes. In this work, we investigate two solutions for CGP. The first utilizes a game-theoretic framework, where each user constitutes a player that wishes to minimize his own social and similarity cost. The second employs local search, and aims at minimizing the global cost. We show that the two approaches can be unified under a common agent-based framework that allows for two types of deviations. In a *unilateral deviation*, an agent switches to a new class, whereas in a *bilateral deviation* a pair of agents exchange their classes. We develop a number of optimization techniques to improve result quality and facilitate efficiency. Our experimental evaluation on real datasets demonstrates that the proposed methods always outperform the state of the art in terms of solution quality, while they are up to an order of magnitude faster.

Keywords Constrained graph partitioning · Game theory · Local search

1 Introduction

Constrained graph partitioning (CGP) partitions nodes of a graph based on a set of input classes with bounded capacities. This partitioning should (i) respect the capacity constraints and (ii) minimize the dissimilarity between each node and its class, and the weight of the edges crossing classes, i.e., those between nodes assigned to different classes. Formally, given an undirected graph $G = (V, E, W)$ and a set of classes P, with minimum and maximum capacity constraints $\min_p$, $\max_p$, $\forall p \in P$, CGP assigns each node to a single class according to the following conditions:

– The capacity constraints are satisfied:

$$\min_p \leq |p| \leq \max_p, \forall p \in P$$

where $|p|$ is the total number of nodes assigned to class p. The minimum (maximum) capacity $\min_p$ ($\max_p$) of p can be 0 (∞) if there is no corresponding constraint.
– Objective Function (1) is minimized:

$$\alpha \cdot \sum\nolimits_{v \in V} d(v, p_v) + (1 - \alpha) \cdot \sum_{\substack{(v,f) \in E \\ \wedge p_v \neq p_f}} w(v, f) \qquad (1)$$

where p_v is the class assigned to node v, $d(v, p_v)$ is the dissimilarity between v and p_v and $w(v, f)$ is the weight of the edge $(v, f) \in E$. The first sum represents the *similarity* cost, which measures the quality of a solution in terms of the total dissimilarity between nodes and their assigned classes. The second sum is the *social* cost, and equals the total weight of the edges between nodes assigned to different classes. The parameter α ($0 \leq \alpha \leq 1$) adjusts the relative importance of the two factors. At the extreme case where $\alpha = 1$, each node is assigned to the most similar class, independently of the

✉ Dimitris Papadias
dimitris@cs.ust.hk

Lefteris Ntaflos
entaflos@ust.hk

George Trimponias
g.trimponias@huawei.com

[1] HKUST, Clear Water Bay, Hong Kong

[2] Hong Kong Science Park, Shatin, Hong Kong

graph connectivity. At the other extreme, when $\alpha = 0$, CGP reduces to the *multiway graph cut* [9], a well-known NP-hard problem that aims at minimizing the weight of the edges crossing partitions, without considering the similarity cost.

CGP has various applications, especially on social networks. For instance, if the classes represent advertisement opportunities, CGP could assign to each user an advertisement that matches his profile, and at the same time it is posted to several of his friends . As motivating example, we use the social event organization (SEO) problem [15] in geo-social networks, where classes correspond to social events promoted to users.[1] The similarity cost is the distance between each user and his recommended event, whereas the social cost is the total weight of edges between friends in different events. Minimization of Objective Function (1) implies that each user v should be assigned to an event that is in the vicinity of v, and it is also recommended to several of v's friends. Capacity constraints capture real-life situations, e.g., an event may need a minimum number of participants, or may be imposed by the available budget that an event advertiser is willing to spend for its campaign.

Figure 1 illustrates six users (circles) and three events (diamonds) in Austin TX, chosen from our experimental dataset. The positions of users and events on the map correspond to their actual coordinates. The friendship weights are denoted beside the edges. The two numbers next to an event are its minimum and maximum capacity. The colors of users denote the event to which they are assigned: v_2 to p_1, v_1 and v_3 to p_2, and v_4, v_5 and v_6 to p_3. The similarity cost is $\alpha \cdot (d(v_2, p_1) + d(v_1, p_2) + d(v_3, p_2) + d(v_4, p_3) + d(v_5, p_3) + d(v_6, p_3))$, where $d(v, p_v)$ is the distance between user v and assigned event p_v. The social cost is $(1 - \alpha) \cdot (w(v_1, v_2) + w(v_2, v_4) + w(v_2, v_6))$, i.e., the weight of edges between nodes of different color. This solution is optimal for the given problem instance, i.e. , it is the assignment of users to events that yields the lowest value of Objective Function (1), and respects all the capacity constraints. Even though the similarity cost in this example takes into account only the distance, we can also consider other factors such as the similarity between user profiles and event descriptions. In general, CGP involves graph partitioning subject to connectivity and one or more additional criteria that assess the similarity between nodes and input classes.

Armenatzoglou et al. [5] study graph partitioning subject to Objective Function (1) without considering capacity constraints. They propose a game-theoretic framework, based on best-response dynamics, that always converges to a Nash equilibrium (i.e., a local minimum). As we demonstrate, the

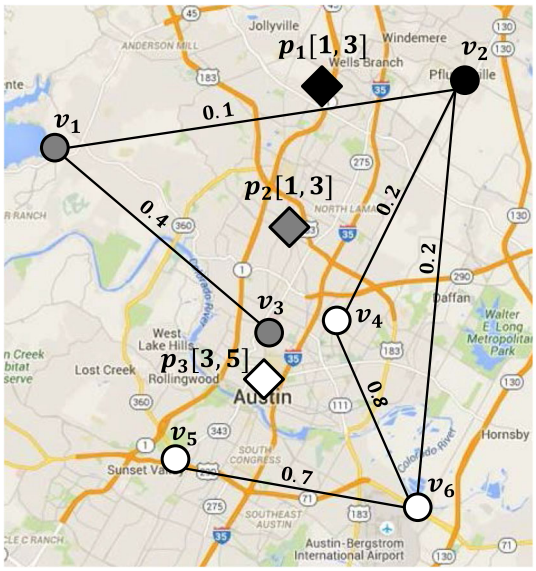

Fig. 1 Running example

Nash equilibrium is inadequate in the presence of capacity constraints, and better solutions can be achieved using the novel concept of *pairwise stability*. Li et al. [15] investigate CGP in the context of SEO.[2] They show that the problem is NP-hard and present greedy heuristics for approximate solutions. However, their algorithms may leave some events empty and some users unassigned, even though there are enough users and sufficiently large capacities. On the other hand, in this case our solutions guarantee the minimum capacity for all events, and assign all users to some event.

In this work, we investigate two solutions, and discuss how they can be unified under a common agent-based framework. The first is a game-theoretic approach that models CGP as a *game*, where users constitute players that choose their most preferred event. The objective is then to obtain an assignment, where no player has an incentive to deviate from his current event. The second solution is a local search framework that directly optimizes Objective Function (1). Our contributions are summarized as follows:

- For the game-theoretic solution, we introduce the notion of *pairwise stability* that allows users to swap classes, if this improves their assignments. We develop a novel type of combined dynamics that converges to a solution that is both a Nash equilibrium and pairwise stable.
- We present a local search approach that allows for unilateral and bilateral deviations. We show how the two solutions can be unified into a common agent-based framework.

[1] In the rest of the paper, we use the terms node/user, edge/friendship and class/event interchangeably.

[2] Although SEO is presented as a utility maximization problem, it could also be defined as a cost minimization problem using Objective Function (1).

– We propose various optimizations that improve the quality of the acquired solution, while reducing the execution time. Additionally, we build data structures for faster real-time performance.
– We perform extensive experiments to demonstrate that our framework achieves superior solutions than the current state of the art, while substantially dropping the execution time. Additionally, the proposed techniques generate the assignments of maximum cardinality subject to the capacity constraints.

The rest of the paper is organized in the following fashion. Section 2 surveys related work. Section 3 introduces the game-theoretic solution for the capacitated graph partitioning problem. Section 4 describes the local search method, and demonstrates that both approaches share a common agent-based structure. Section 5 presents the concrete algorithms and the associated data structures. Section 6 discusses various extensions and special cases of the agent-based framework. Section 7 contains the experimental evaluation. Section 8 concludes the paper.

2 Related work

Section 2.1 overviews existing work on CGP and related problems. Section 2.2 provides background on game theory, and Sect. 2.3 on local search.

2.1 General

Graph clustering has received considerable attention in several application domains. The various approaches can be classified as global versus local (depending on whether they cluster the entire, or part of the, graph), flat versus hierarchical (in which case, clusters may contain sub-clusters) and off-line versus on-line methods that operate without complete knowledge of the graph. Algorithmic solutions have also large diversity, including minimum cuts, maximum flows, spectral methods, Markov chains and random walks. An extensive survey can be found in [20]. Even though graph clustering techniques can partition the graph according to various criteria, in addition to node connectivity CGP considers the similarity of the nodes to a set of input classes. In this sense, CGP is more a classification, rather than clustering problem.

Armenatzoglou et al. [5] show that graph partitioning according to Objective Function (1) constitutes an instance of uniform metric labeling (UML) [11], a well-known NP-hard problem. The experimental evaluation of [5] suggests that theoretical UML algorithms with approximation guarantees are prohibitively expensive for graphs with more than a few hundred nodes. Several computer vision problems such as stereo matching [6,8], photomontage [2] and interactive photo segmentation [7,19] can also be modeled similarly to UML. Correspondingly, those problems are solved by approximations such as graph cuts [8,12], generalized belief propagation [22] and tree re-weighted message passing [21]. Balanced metric labeling (BML) [16] and capacitated metric labeling (CML) [3] extend UML by imposing a maximum capacity to each class. Their difference is that in BML the constraint for all the classes is the same, while CML allows different maximum capacity per class. CML is solved by an $O(\log(|V|))-$ approximation algorithm in $(|V|+1)^{3|P|}poly(|V|,|P|)$ time, which is too slow for practical applications.

Social event organization (SEO) is an application of CGP on large social graphs. The most effective greedy algorithm [15], hereafter referred to as SEOG, maintains all (user, event) pairs in a Fibonacci heap. Initially, the heap is sorted in increasing order of distance (i.e., similarity cost) between every user and event. Until the heap is empty, SEOG pops each pair (v, p); if v is unassigned and the capacity constraints of p are satisfied, v is assigned to p, and the pairs (f, p) of every unassigned friend f of v in the heap are updated because their social cost decreases. An assignment can be temporary, when p has not yet met its minimum capacity, in which case p is called *phantom*. In order to prevent the creation of numerous phantom events, the *deficit* variable counts the number of users needed for the phantom events to be realized. If the value of deficit exceeds the unassigned users, those users will not be enough to realize the already created phantom events; therefore, v will not be assigned to p.

In SEO, the capacity constraints have to be respected *only for the events with at least one user assigned*. Consequently, SEOG: (i) may leave some events empty despite the existence of sufficient users, (ii) may leave some users unassigned despite the fact that the maximum capacities are large enough to accommodate all users. On the contrary, in CGP we consider that, if the number of users is adequate and the maximum capacities are sufficient, *no event will be left empty, and no user will be left unassigned*. In order to use SEOG as a benchmark in our experimental evaluation, we assume that all minimum capacities are zero, so that there are no phantom events.

2.2 Game theory

Game theory studies models of conflict and cooperation between rational decision makers. A game in strategic form (or in normal form) is the ordered triple $\mathcal{G}_{NE} =< V, (\mathcal{P}_v)_{v\in V}, (c_v)_{v\in V} >$, where:

– V is the finite set of players.

Input: Strategic game $\mathcal{G}_{NE} =< V, (\mathcal{P}_v)_{v\in V}, (c_v)_{v\in V} >$
Output: Nash equilibrium
1: Assign an arbitrary strategy to each player $v \in V$
2: **repeat**
3: **for** each player $v \in V$
4: select best-response strategy:
5: $p_v^* \leftarrow \arg\min_{p_v \in \mathcal{P}_v} c_v(p_1, \ldots, p_v, \ldots, p_{|V|})$
6: **until** Nash Equilibrium
7: **return** the strategy of each player $v \in V$

Fig. 2 Best-response dynamics

- $\mathcal{P}_v$ is the finite strategy set of player v, for every player $v \in V$. We denote the strategic space, i.e., the set of all vectors of strategies, as $\mathcal{P} = \times_{v\in V}\mathcal{P}_v = \mathcal{P}_1 \times \cdots \times \mathcal{P}_{|V|}$.
- $c_v : \mathcal{P} \rightarrow R$ is a function associating each vector of strategies $\boldsymbol{p} \in \mathcal{P}$ with the cost $c_v(\boldsymbol{p})$ to player v, for every player $v \in V$.

The *best-response dynamics* in Fig. 2 constitutes a distributed iterative framework that describes the decision-making process. In every *round*, each player plays a best-response strategy to the rest of the players, i.e., player v selects:

$$p_v^* = \arg\min_{p_v \in \mathcal{P}_v} c_v(p_1, \ldots, p_v, \ldots, p_{|V|})$$

A game in strategic form accepts a pure Nash equilibrium (NE) if there exists a vector of strategies $\boldsymbol{p} = (p_1, \ldots, p_v, \ldots, p_{|V|}) \in \mathcal{P}$, such that for every v it holds that: $c_v(p_1, \ldots, p_v, \ldots, p_{|V|}) \leq c_v(p_1, \ldots, p'_v, \ldots, p_{|V|})$, for every $p'_v \in \mathcal{P}_v$. Thus, a NE describes a stable state, where no player has an incentive to change his strategy, provided the other players do not deviate as well. Since best-response dynamics may yield any of (the potentially many) NE, two metrics are widely employed to assess the quality of the acquired equilibrium: (i) Price of stability (PoS) is the ratio between the total cost of the equilibrium with the lowest cost and the optimal solution OPT, called the social optimum; (ii) the price of anarchy (PoA) is the ratio between the total cost of the equilibrium with the highest total cost and OPT. For nonnegative costs, it holds that $1 \leq PoS \leq PoA$.

An important class of games consists of *potential* games, where the incentive of all players to change their strategy is expressed through a global *potential function*. Let $\overline{\mathcal{P}_v} = \times_{u\in V-\{v\}}\mathcal{P}_u$ be the set of strategies of all players except v. Given $\overline{\boldsymbol{p}_v} \in \overline{\mathcal{P}_v}$, we can also write $\boldsymbol{p} = (p_v, \overline{\boldsymbol{p}_v})$, i.e., a vector of strategies $\boldsymbol{p}$ consists of the strategy of v along with the strategies of all players other than v. A game is called *exact potential* if there is a function $\Phi : \mathcal{P} \rightarrow \mathbb{R}$ such that for every $p_v, p'_v \in \mathcal{P}_v$ and $\overline{\boldsymbol{p}_v} \in \overline{\mathcal{P}_v}$, the difference in the individual cost by switching strategy is equal to the difference in Φ:

$$\Phi(p'_v, \overline{\boldsymbol{p}_v}) - \Phi(p_v, \overline{\boldsymbol{p}_v}) = c_v(p'_v, \overline{\boldsymbol{p}_v}) - c_v(p_v, \overline{\boldsymbol{p}_v}) \quad (2)$$

Potential games accept at least one pure NE. Moreover, the best-response dynamics of Fig. 2 always converges to a pure NE in a finite number of rounds. Unconstrained graph partitioning under Objective Function (1) can be modeled as a potential game using Potential Function (3) [5]. However, the framework of [5] is inapplicable to CGP because, due to the capacity constraints, a user v may not be able to switch from p to his preferred event p', if the swap violates the minimum capacity of p or the maximum capacity of p'.

$$\Phi(\boldsymbol{p}) = \alpha \cdot \sum_{v\in V} d(v, p_v) + (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \wedge \\ p_v \neq p_f}} \frac{1}{2} \cdot w(v, f) \quad (3)$$

Various graph problems such as minimum spanning tree and community detection have been solved using game-theoretic techniques. There is also prior work on coalitional, clustering, or coordination games with and without capacity constraints [4,10,18]. Similar to CGP, these approaches assume that every player corresponds to a node in a graph and the graph edges denote the relationships among the players. Coalitions of players can make coordinated deviations to improve their local utilities to reach exact or approximate equilibria with certain guarantees with respect to the socially optimal solution. Our game-theoretic solution shares certain similarities with these works, in the sense that single agents or pairs of agents make coordinated deviations to decrease their local costs. On the other hand, our local search solution focuses on the social welfare, i.e., the global function (1), so it may produce deviations that increase the local costs of some players as long as the global cost decreases.

2.3 Local search in combinatorial optimization

A combinatorial optimization problem can be modeled as a pair $(\mathcal{S}, c)$, where $\mathcal{S}$ is the set of feasible solutions and $c : \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$ is a function that assigns a nonnegative cost to every feasible solution. The goal is to find a globally optimal solution, i.e., a feasible solution S^* such that $c(S^*) \leq c(S)$, $\forall S \in \mathcal{S}$. Local search constitutes a practical approach for solving hard combinatorial optimization problems approximately. A neighborhood function $\mathcal{N} : \mathcal{S} \rightarrow 2^{\mathcal{S}}$ for the problem $(\mathcal{S}, c)$ is a mapping from a feasible solution to a subset of feasible solutions. $\mathcal{N}(S)$ is called the neighborhood of S. We assume that $S \in \mathcal{N}(S)$. A feasible solution $\overline{S}$ is said to be locally optimal with respect to $\mathcal{N}$ if $c(\overline{S}) \leq c(S)$, $\forall S \in \mathcal{N}(\overline{S})$. The local search problem is that of finding a locally optimal solution.

Input: Combinatorial optimization problem $(\mathcal{S}, c)$ with neighborhood function $\mathcal{N}$
Output: Locally optimal solution
1: Compute an initial feasible solution $\overline{S}$;
2: **while** $\overline{S}$ is not locally optimal **do**
3: Choose $S \in \mathcal{N}(\overline{S})$ such that $c(S) < c(\overline{S})$;
4: $\overline{S} \leftarrow S$
5: **return** $\overline{S}$;

Fig. 3 Iterative improvement

The standard local search strategy, also called iterative improvement, is described in Fig. 3. It first computes an initial feasible solution $\overline{S}$. It then repeatedly searches the neighborhood to find a better feasible solution until a local optimum is reached. There are several versions of local search [1]. *Hill climbing* selects (at Line 3) the solution that achieves the largest drop of the cost function. To avoid getting trapped at a local minimum of poor quality, hill climbing can be restarted with different initial solutions. Other variants of local search also allow transitions to solutions with equal (e.g., *tabu search*), or higher cost (*random walk*, *simulated annealing*) in order to escape local minima.

Assuming all cost coefficients are rational numbers, the iterative improvement framework of Fig. 3 terminates in a pseudo-polynomial number of iterations. Indeed, if we multiply all coefficients by their smallest common denominator, then the algorithm will terminate after at most $C_{\max}$ iterations, where $C_{\max}$ is the largest coefficient (after multiplication) because each step is guaranteed to decrease the total cost by at least one unit. Even though it is an open question whether there are polynomial-time algorithms for computing a local optimum in the general case [17], approximate local optima can be computed in polynomial time. In this work, we focus on exact local search and do not investigate more complex approximate schemes, since exact techniques can usually converge very fast even for large input sizes [17].

3 Game-theoretic framework for CGP

We model CGP as the *game* $G =< V, (\mathcal{P}_v)_{v \in V}, (c_v)_{v \in V} >$, where the set of players V corresponds to the users and the strategy set $\mathcal{P}_v$ of user v coincides with the set of events P. Each event $p \in P$ may have capacity constraints $\min_p \leq |p| \leq \max_p$, i.e., the number $|p|$ of users assigned to event p must be between $\min_p$ and $\max_p$. Equation (4) describes the cost of assignment p_v for user v given the strategies $\overline{\boldsymbol{p}_v}$ of the other players. It consists of the weighted sum of the *similarity cost*, i.e., the distance between v and event p_v, and the *social cost*, i.e., half of the total weight of the edges connecting v to his friends that are assigned to different events. Since each edge (v, f) is considered in the cost of both v and f, by summing up the costs of all users, we obtain Objective Function (1).

$$c_v(p_v, \overline{\boldsymbol{p}_v}) = \alpha \cdot d(v, p_v) + (1 - \alpha) \cdot \sum_{\substack{(v,f) \in E \wedge \\ p_v \neq p_f}} \frac{1}{2} \cdot w(v, f) \tag{4}$$

An *assignment* S is a mapping from the set of users V to the set of events P, which is (1) *total*, i.e., every user must be assigned to exactly one event, assuming that the total event capacity can accommodate all users, and (2) *feasible*, i.e., it satisfies the upper and lower capacity constraints. The set of all assignments is denoted as $\mathcal{S}$, in accordance with Sect. 2.3. An assignment $S \in \mathcal{S}$ naturally corresponds to a vector of strategies $\boldsymbol{p} \in \mathcal{P}$, and vice versa. For this reason, we use the two terms and symbols interchangeably in the remainder of the paper.[3] We consider for now that $\sum_{p \in P} \min_p \leq |V| \leq \sum_{p \in P} \max_p$, so that all classes can reach their minimum capacity, and all users can be assigned to some class. We discuss other cases in Sect. 6.

From a game-theoretic perspective, the goal of each player is to select the event that minimizes his own cost, as expressed by Eq. (4). Player v has an incentive to perform a *unilateral deviation* from his assigned event p_v to another one p'_v, if p'_v yields a smaller cost for v. In CGP, a unilateral deviation is *legal*, if it does not violate the minimum capacity of p_v, or the maximum capacity of p'_v. Now consider two users v and u, assigned to p_v and p_u, respectively, and that p_v is at its minimum capacity (or p_u at its maximum capacity), so that p_v cannot drop (or p_u cannot gain) a user. Also, assume that both v and u would benefit by swapping events. Although individual unilateral deviations would be illegal, it is possible for v and u to exchange events by a *bilateral deviation*. In order to cover such cases, we introduce the concept of *pairwise stability* (PS). A pair (v, u) is *unstable* if swapping p_v and p_u decreases the individual cost of v and/or u, and increases the cost of neither.[4] An assignment is *stable*, iff there is no pair of unstable users.

Figure 4 presents the general framework combining NE and PS. Line 1 computes an initial assignment. During this step, all users are assigned to some event, and all events reach their minimum, but do not exceed their maximum capacity. The outer loop (Lines 2–9) corresponds to a *super-round*. Each super-round performs rounds of *unilateral* deviations (Lines 3–5), until reaching a Nash equilibrium. These are similar to the rounds of the conventional best-response dynamics of Fig. 2, except that only *legal deviations* (i.e., not violating capacity constraints) are allowed. Then, rounds of *bilateral* deviations allow unstable pairs of users

[3] Usually, the former emphasizes the entire solution, whereas the latter emphasizes the individual agent strategies.

[4] From a game-theoretic perspective, our definition ensures individual rationality since a player participates in a bilateral deviation, if and only if he is not worse off by participating.

Input: Strategic game $\mathcal{G} =< V, (\mathcal{P}_v)_{v\in V}, (c_v)_{v\in V} >$ with capacity constraints
Output: Nash equilibrium and pairwise stable assignment
1: Compute a feasible initial assignment
2: **repeat**
3: **repeat**
4: perform *legal unilateral* deviations
5: **until** Nash equilibrium
6: **repeat**
7: perform *bilateral* deviations
8: **until** pairwise stability
9: **until** Nash equilibrium and pairwise stability
10: **return** the strategy of each player $v \in V$

Fig. 4 Combined dynamics for CGP

to swap events, until reaching a stable assignment (Lines 6–8). Observe that after a bilateral deviation, a user v may be assigned to an event p, which was not allowed by a unilateral deviation because p was full. This may create new opportunities for v to further drop his cost, which will be considered at the next super-round.

Next, we show that the combined dynamics of Fig. 4 constitutes a potential game using Potential Function (3) and always converges to a $NEPS$ solution (i.e., both a Nash equilibrium and pairwise stable). We start with the unilateral deviations in Lemma 1. The proofs of all Lemmas and Propositions can be found in the "Appendix."

Lemma 1 *The difference in Potential Function (3) $\Delta\Phi_{UNI}(v, p_v, p'_v, \overline{\boldsymbol{p}_v})$ due to a unilateral deviation in assignment $\boldsymbol{p}$ of player v from event p_v to p'_v, while the rest of the players do not deviate, is:*

$$\begin{aligned}\Delta\Phi_{UNI}(v, p_v, p'_v, \overline{\boldsymbol{p}_v}) &= \Phi(p'_v, \overline{\boldsymbol{p}_v}) - \Phi(p_v, \overline{\boldsymbol{p}_v}) \\ &= c_v(p'_v, \overline{\boldsymbol{p}_v}) - c_v(p_v, \overline{\boldsymbol{p}_v}) \\ &= \Big(\alpha \cdot d(v, p'_v) + (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \\ \wedge p_f = p_v}} \frac{1}{2} \cdot w(v, f)\Big) \\ &\quad - \Big(\alpha \cdot d(v, p_v) + (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \\ \wedge p_f = p'_v}} \frac{1}{2} \cdot w(v, f)\Big)\end{aligned} \tag{5}$$

The right side of Eq. (5) describes the local cost change of v due to the unilateral deviation. This implies that NE decreases the potential function, since it only favors deviations that drop the player's local cost.

For bilateral deviations, we first introduce some additional notation. Let $\overline{\mathcal{P}_{vu}} = \times_{z\in V-\{v,u\}} \mathcal{P}_z$ be the set of strategies of all players except v and u. Consider any feasible assignment $\boldsymbol{p} = (p_v, p_u, \overline{\boldsymbol{p}_{vu}})$ *where users $v, u \in V$ are assigned to events p_v, p_u, respectively, with $p_v \neq p_u$*; $c_v(p_v, p_u \cup \overline{\boldsymbol{p}_{vu}})$ is the cost of user v for event p_v assuming that u is assigned to p_u and the rest of the users at their current events. Lemma 2 describes the cost difference in the potential function due to bilateral deviations.

Lemma 2 *Assume v, u swap events in assignment $\boldsymbol{p}$, while the rest of the players do not deviate. For the new assignment $(p'_v, p'_u, \overline{\boldsymbol{p}_{vu}})$, where $p'_v = p_u$ and $p'_u = p_v$, the difference $\Delta\Phi_{BI}(v, u, \boldsymbol{p})$ in Potential Function (3) is:*

$$\begin{aligned}\Delta\Phi_{BI}(v, u, \boldsymbol{p}) &= \Phi(p'_v, p'_u, \overline{\boldsymbol{p}_{vu}}) - \Phi(p_v, p_u, \overline{\boldsymbol{p}_{vu}}) \\ &= \Big(c_v(p'_v, p_u \cup \overline{\boldsymbol{p}_{vu}}) - c_v(p_v, p_u \cup \overline{\boldsymbol{p}_{vu}}) + \frac{1}{2} w(v, u)\Big) \\ &+ \Big(c_u(p'_u, p_v \cup \overline{\boldsymbol{p}_{vu}}) - c_u(p_u, p_v \cup \overline{\boldsymbol{p}_{vu}}) + \frac{1}{2} w(v, u)\Big)\end{aligned} \tag{6}$$

We note that the expression in the right hand describes the sum of the local cost changes of v and u due to their bilateral deviation. This implies that pairwise stability decreases the potential function since by definition it favors bilateral deviations that decrease the sum of the players' local costs. Our next result uses Lemmas 1 and 2 to establish convergence of the combined dynamics.

Proposition 1 *In a finite potential game, from an arbitrary feasible assignment, the combined dynamics of Fig. 4 always converges to a $NEPS$ solution in a finite number of rounds.*

A central question concerns the quality of the assignment that the combined dynamics converges to.

Proposition 2 *The PoS in the capacitated game using NEPS is upper bounded by 2.*

Regarding the PoA, we derive a general upper bound that holds for any setting. For each pair of a player $v \in V$ and an event $p \in P$, we define:

$$\xi(v, p) = \alpha \cdot c(v, p) + (1-\alpha) \cdot \sum_{i=1}^{\max_{\overline{p}}} \frac{1}{2} w_{(i)}(v),$$

where $\max_{\overline{p}} = \sum_{p'\in P \wedge p' \neq p} \max_{p'}$ is the sum of capacities of all events other than p, and $w_{(i)}(v)$ is the ith highest weight friendship of user v among the friendship weights $w(v, f)$, $f \neq v$. The quantity $\xi(v, p)$ can be interpreted as the worst case scenario for user v when he is assigned to event p. Indeed, this happens when all events other than p accommodate v's closest friends, i.e., his neighbors with the largest edge weights.

For every user v, we define a permutation $p^v_1, \ldots, p^v_{|P|}$ on the set of events, so that $\xi(p^v_1, v) \geq \cdots \geq \xi(p^v_{|P|}, v)$. In other words, p^v_k is the event with the kth highest cost $\xi(v, p)$ for user v among all events $p \in P$. Obviously, the worst case scenario (in terms of events) for v occurs when he is assigned to event p^v_1, the second worst case scenario when he is assigned to p^v_2, and so forth.

Based on the these permutations, for each event p we define $V^p_{(k)}$, $1 \leq k \leq |P|$, as the subset of users for whom event p has the kth highest ξ value, i.e., $V^p_{(k)} = \{v \in V : p^v_k = p\}$. Intuitively, the set $V^p_{(k)}$ contains those users for which event p is the kth worse among the $|P|$ scenarios. Let $\hat{V}^p_{(k)} \subseteq V^p_{(k)}$ be the subset of $V^p_{(k)}$ consisting of the $\max_p$ users with the highest $\xi(v, p)$ for event p among all users in $V^p_{(k)}$; if $|V^p_{(k)}| < \max_p$, we simply define $\hat{V}^p_{(k)} = V^p_{(k)}$. Thus, $\hat{V}^p_{(k)}$ simply contains the $\max_p$ users in $V^p_{(k)}$ with the highest ξ values. Finally, we define $\Xi^p_{(k)} = \sum_{v \in \hat{V}^p_{(k)}} \xi(v, p)$, i.e., the sum of the ξ values over all users in $\hat{V}^p_{(k)}$.

Proposition 3 *The PoA in the capacitated game using NE and PS is upper bounded by*

$$\frac{\sum_{k=1}^{|P|} \max_{p \in P} \Xi^p_{(k)}}{\alpha \cdot \sum_{v \in V} \min_{p \in P} c(v, p)}.$$

The above bound is pessimistic since it suggests that the PoA may depend on the worst per user costs as opposed to the uncapacitated game which only depends on the minimum per user costs [5]. But as the following example demonstrates, the PoA can indeed be arbitrarily close to the worst possible one.

Example 1 Consider N users $v_1, \ldots, v_N$, and N events $p_1, \ldots, p_N$ with maximum capacity constraints equal to 1. We further assume no friendships, i.e., all weights are equal to 0, and that $\alpha = 1$. User v_i, $1 \leq i \leq N-1$, has assignment cost 0 for event p_i and ϵ for all other events. User v_N has cost M for event p_N and ϵ for all other events, where M is an arbitrarily large number.

An optimal assignment S^{opt} assigns the first $N-2$ users to their preferred event with 0 assignment cost, and user v_{N-1} (resp. p_N) to event p_N (resp. p_{N-1}) with ϵ cost. Thus, S^{opt} has total cost of $2 \cdot \epsilon$. The worst possible assignment S^{worst} is when v_N is assigned to p_N and the rest of the users to an event different from their preferred one, yielding total cost $(N-1) \cdot \epsilon + M$.

Consider now the assignment $S^{\#}$ where each v_i is assigned to event p_i with total cost M. It is easy to see that $S^{\#}$ is NE and PS, since all users but v_N are already assigned to their preferred event. Unfortunately, as $\epsilon \to 0$, the cost $C(S^{\#}) \to C(S^{\text{worst}})$, resulting in a PoA arbitrarily close to the worst possible one. □

4 Local search framework for CGP

In order to apply the iterative improvement approach of Fig. 3, we set the cost function $c(S)$ to Objective Function (1), and define the neighborhood of a solution S as the subset $\mathcal{N}(S)$ of feasible solutions that can be obtained from S by performing a single unilateral or bilateral deviation. Similar to the game-theoretic framework, a unilateral deviation is legal if it does not violate the minimum capacity of the old event, or the maximum capacity of the new one. However, we now allow bilateral deviations that may increase the cost of an individual user, provided that the total cost of the two users participating in the swap decreases. Given that: (i) the set of feasible solutions $\mathcal{S}$ is finite and, (ii) each iteration drops the cost function $c(S)$, then a solution is generated at most once. Thus, local search terminates to a locally optimal solution after at most $|\mathcal{S}|$ iterations.

A natural question concerns the quality of the locally optimal solution with respect to the global optimum. As the next example demonstrates, the local optimum that the iterative improvement method reaches may be far from the globally optimal solution.

Example 2 Consider a setting of N users $v_1, \ldots, v_N$, and N events $p_1, \ldots, p_N$ with upper capacity bounds equal to 1. Obviously, every event is assigned a single user. Furthermore, assume that no pair of users are friends, so that the social cost of any assignment is always 0. Now, let $0 < d_1 < \cdots < d_N$ be an increasing sequence of N positive numbers. Moreover, assume that the distances of the N users to the N events are as follows:

User	Event				
	p_1	p_2	...	p_{N-1}	p_N
v_1	d_1	d_2	...	d_{N-1}	d_N
v_2	d_N	d_1	...	d_{N-2}	d_{N-1}
...	...	...	...	...	...
v_{N-1}	d_3	d_4	...	d_1	d_2
v_N	d_2	d_3	...	d_N	d_1

Thus, the distance of user v_1 to event p_j is equal to d_j. The distance of every other user v_i to each event can then be computed by circularly shifting the corresponding distance of user v_1, $i-1$ times to the right. Clearly, the optimal assignment S^{opt} assigns user v_i to event p_i for a total cost of $\alpha \cdot N \cdot d_1$. Let $S^{\#}$ be an alternative assignment that assigns v_i to $p_{i+\lceil \frac{n}{2} \rceil - 1}$ (note that we consider modular arithmetic). The key is to note that user v_i prefers to swap with v_j, if and only if, v_j does not want to swap with user v_i. Since the distances of user v_j are generated by circularly shifting the distances of v_i, the cost change of v_i is equal to the opposite of the cost change of v_j. Hence, the sum of the cost changes of v_i and v_j after swapping events is 0, which means that $S^{\#}$ is a local optimum. The total cost of $S^{\#}$ is $\alpha \cdot N \cdot d_{\lceil \frac{n}{2} \rceil}$, yielding a ratio between the local optimum and the global optimum equal to $\frac{d_{\lceil \frac{n}{2} \rceil}}{d_1}$. □

Table 1 Game theory versus local search under the agent-based framework

	Game theory	Local search
Global function	Potential function (3)	Objective function (1)
Unilateral deviation	Individual cost Eq. (4)	Individual cost Eq. (8)
Bilateral deviation	At least one decreases local cost, no player increases cost	Pair of players decreases sum of local costs

Lemma 3 *For a unilateral deviation in assignment* $\boldsymbol{p}$ *where a user* v *switches from event* p_v *to* p'_v*, the change in Objective Function* (1) *is:*

$$\Delta C_{UNI}(v, p_v, p'_v, \overline{\boldsymbol{p}_v}) = \left(\alpha \cdot d(v, p'_v) + (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \\ \wedge p_f = p_v}} w(v, f) \right) - \left(\alpha \cdot d(v, p_v) + (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \\ \wedge p_f = p'_v}} w(v, f) \right) \quad (7)$$

The cost changes in Eq. (7) only depend on the similarity and social cost of the user(s) involved in the deviation because the costs of other users cancel out. Comparing with its game-theoretic counterpart (5), we can see that the two equations are structurally very similar, in the sense that the cost changes are identical except for a factor $\frac{1}{2}$ in the edge costs. This motivates us to introduce the following cost function for v:

$$\tilde{c}_v(p_v, \overline{\boldsymbol{p}_v}) = \alpha \cdot d(v, p_v) + (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \wedge \\ p_v \neq p_f}} w(v, f) \quad (8)$$

Equation 8 enables us to describe local search in a framework analogous to the game theoretic, by defining the cost of an individual agent (similar to that of a player). Specifically, we can then rewrite the cost change due to the unilateral deviation as:

$$\Delta C_{UNI}(v, p_v, p'_v, \overline{\boldsymbol{p}_v}) = \tilde{c}_v(p'_v, \overline{\boldsymbol{p}_v}) - \tilde{c}_v(p_v, \overline{\boldsymbol{p}_v}),$$

which is structurally identical to (5). A similar observation is true for bilateral deviations:

Lemma 4 *For a bilateral deviation in assignment* $\boldsymbol{p}$ *where two users* $v, u \in V$ *swap events* (v/u *from* p_v/p_u *to* p_u/p_v)*, the change in Objective Function* (1) *is:*

$$\begin{aligned} &\Delta C_{BI}(v, u, \boldsymbol{p}) \\ &= \left(\tilde{c}_v(p'_v, p_u \cup \overline{\boldsymbol{p}_{vu}}) - \tilde{c}_v(p_v, p_u \cup \overline{\boldsymbol{p}_{vu}}) + w(v, u) \right) \\ &+ \left(\tilde{c}_u(p'_u, p_v \cup \overline{\boldsymbol{p}_{vu}}) - \tilde{c}_u(p_u, p_v \cup \overline{\boldsymbol{p}_{vu}}) + w(v, u) \right) \end{aligned} \quad (9)$$

Again, we notice the structural similarities between Eqs. (9) and (6). The practical implication of the above is that we can define local search using the game-theoretic paradigm, with two differences. First, we replace the factor $\frac{1}{2}$ in the player's local cost function (4) by 1, as in Eq. (8). Second, since local search focuses on the global objective, we perform bilateral deviations that drop the total cost of the two participating users, even if one user is worse off after the deviation (recall that this is not allowed in the game-theoretic solution). We emphasize that in local search each deviation is guaranteed to decrease Objective Function (1), whereas in the game-theoretic framework each deviation decreases the Potential Function (3). Table 1 summarizes the differences between game-theoretic and local search solutions.

Viewing the two solutions under a unified agent-based framework enables the application of common algorithms and indexing schemes. For instance, the pseudo-code of Fig. 4 can be modified to describe local search as follows: (i) Lines 3–5 perform iterative improvement with unilateral deviations using Eq. (8) for the individual cost; (ii) Lines 6–8 perform iterative improvement with bilateral deviations according to their functionality in local search, and (iii) in Line 9 the process terminates until no further improvement is possible. The next section presents the concrete algorithmic components and data structures of the unified framework.

5 Algorithms

We describe the concrete algorithms and data structures, which are applicable with minor modifications to both the game-theoretic and local search frameworks, hereafter referred to as $GAME$ and LS. Section 5.1 proposes a method for generating the initial assignment based on sampling. Section 5.2 presents the functions for performing unilateral and bilateral deviations. We focus our description on $GAME$, and discuss the changes for LS whenever applicable.

5.1 Initial assignment

A random initial assignment may lead to poor solutions. As shown in our experiments, even more sophisticated heuristics based on proximity (e.g., assigning each user to the closest event) yield low-quality solutions. Instead, we propose an $INIT$ algorithm that involves two phases: Phase 1 fills all events to their minimum capacity, whereas Phase 2 assigns

Input: Geo-social Graph $G = (V, E, W)$
Set of events P with constraints $min_p, max_p, \forall p \in P$
$V_{un} = V, P_{op} = P, \mathcal{C} = \emptyset, n$
$d(v, p), \forall v \in V, \forall p \in P$
Output: Initial assignment

```
1: for each user v
2:    for each event p
3:       c(v,p) ← α · d(v,p)
4:       for each friend f of v
5:          c(v,p) ← c(v,p) + ½(1 − α) · w(v,f)
6:    H_v = min-heap containing c(v,p) for each event p
7: while |P_op| > 0
8:    c_m ← ∞
9:    if |V_un| ≥ n
10:      C ← {n random distinct users from V_un}
11:   else
12:      C ← V_un
13:   for each user v ∈ C
14:      p* ← top(H_v) (p* has min cost for v in P_op)
15:      if c(v,p*) < c_m
16:         c_m ← c(v,p*), v' ← v, p' ← p*
17:   p_v' ← p', V_un ← V_un − {v'}
18:   for each friend f ∈ V_un of v'
19:      c(f,p') ← c(f,p') − ½(1 − α) · w(v',f)
20:      decrease key(H_f, p', c(f,p'))
21:   if |p'| = min_p'
22:      P_op ← P_op − {p'}
23:      for each user v ∈ V_un
24:         remove p' from H_v
25: re-build heaps; P_op = P
26: Repeat 7-25, except Line 7: P_op → V_un, Line 21: min_p' → max_p'
```

Fig. 5 $INIT$ function

all the users unassigned during Phase 1. Both phases adopt an iterative approach based on sampling. At each iteration, a sample of users is randomly selected and the cost of their possible assignments is computed. The assignment with the lowest cost (among the sampled users) is then performed.

Figure 5 illustrates the pseudo-code of $INIT$ for $GAME$. Lines 1–5 perform an initialization step that computes the cost $c(v, p)$ of assigning each user v to every event p according to Eq. (4). These costs are stored in a $|V||P|$ array, which we refer to as the *cost table*. Since in the beginning all users are unassigned, we set a *maximum social cost* per user v and event p, assuming that all the friends of v are at events other than p. For a user v, the costs for all events are stored in a min-heap H_v of size $|P|$; the event p^* with the lowest cost for v is the one at the top of H_v. Since there is a heap per user, the total number of *user heaps* is $|V|$.

Lines 7–24 implement Phase 1. Let P_{op} be the set of *open* events that have not reached their minimum capacity, and V_{un} be the set of unassigned users (initially, $P_{op} = P$ and $V_{un} = V$). While there are still open events, $INIT$ selects a random set $\mathcal{C}$ of distinct users from V_{un}. For each user v in $\mathcal{C}$, it obtains the event p^* with the lowest cost among events in P_{op}, which is at the top of H_v (Lines 13–14). Let v' be the user with the minimum lowest cost and p' be the corresponding event; v' is assigned to p' and removed from V_{un} (Line 17). Lines 18–20 decrease the costs of the friends of v' for p', to reflect the new assignment, and update the corresponding heaps. Observe that since (i) we initially set the maximum social score for each user/event pair, and (ii) during $INIT$ each user is assigned exactly once, the cost of a user cannot increase due to the assignment of a friend. Finally, if the user cardinality $|p'|$ of p' reaches its minimum capacity $\min_{p'}$, p' is excluded from P_{op}, and the event *closes* (i.e., it will not receive more assignments at Phase 1). Lines 23–24 remove closed events from the heaps of all unassigned users. (The rest of the users will not be re-assigned during $INIT$, and their heaps will not be used again.) At the end of Phase 1, since all events close, all heaps become empty.

The second phase (Lines 25–26) repeats the same process with the following differences: (i) The heaps are rebuilt using the user/event costs computed during Phase 1 and stored in the cost table; (ii) P_{op} now contains events that have reached their minimum, but are below their maximum capacity (initially, $P_{op} = P$); (iii) the loop is repeated while there are unassigned users ($|V_{un}| > 0$ instead of $|P_{op}| > 0$ in Line 7); and (iv) an event p from P_{op} closes when it becomes full ($|p'| = \max_{p'}$) in Line 21. For LS, the only modifications of $INIT$ are at Lines 5 and 19, where the user costs are computed according to Eq. (8) (i.e., the factor $\frac{1}{2}$ is removed from the edge costs).

Figure 6 illustrates $INIT$ for LS, using the running example of Fig. 1 and sample size $n = 2$. Phase 1 fills the three events p_1, p_2 and p_3, to their minimum capacities 1, 1 and 3, respectively. Initially, the set P_{op} of open events includes all events, and the set V_{un} of unassigned users contains all users. Assume that at iteration 1 (first row of the table), v_1 and v_4 are randomly selected in the sample set $\mathcal{C}$. The top of the heaps of those users (i.e., events with minimum cost) are shown under column $\mathcal{C}$. In our example, the best event for v_1 is p_1 with cost 0.139, while for v_4 the best event is p_3 with cost 0.260. Therefore, $INIT$ (i) assigns v_1 to p_1, and excludes v_1 from V_{un}. Since p_1 reaches its minimum capacity ($\min_{p_1} = 1$), it is removed from P_{op}. Finally, the assignment of v_1 to p_1 reduces the cost of neighbors v_2 and v_3 for p_1. V_{un}, P_{op} and nodes with cost updates are shown in the rightmost three columns.

Similarly, at iteration 2, the sample set is $\mathcal{C} = \{v_5, v_2\}$; v_2 has the minimum cost and is assigned to p_2, which reaches its minimum capacity and is removed from P_{op}. The only event remaining open is p_3 with a minimum capacity of 3. Accordingly, iterations 3–5 fill p_3 with v_3, v_5 and v_4, concluding Phase 1. Phase 2 assigns the only unassigned user v_6 to the event (p_3) with the minimum cost (0.285), and concludes with a single iteration. The diagram on the right-hand side of Fig. 6 illustrates the resulting assignment after $INIT$. This assignment respects all constraints, since events reach their minimum capacity at Phase 1, and only open events

Phase/Iteration		$\mathcal{C}$	Assignments	V_{un}	P_{op}	Changes
	1/1	$v_1[p_1(0.139)], v_4[p_3(0.260)]$	$v_1 \rightarrow p_1$	v_2, v_3, v_4, v_5, v_6	p_2, p_3	$c(v_2, p_1)$ $c(v_3, p_1)$
	1/2	$v_5[p_3(0.195)], v_2[p_2(0.166)]$	$v_2 \rightarrow p_2$	v_3, v_4, v_5, v_6	p_3	$c(v_1, p_2)$ $c(v_4, p_2)$ $c(v_6, p_2)$
	1/3	$v_3[p_3(0.109)], v_4[p_3(0.260)]$	$v_3 \rightarrow p_3$	v_4, v_5, v_6	p_3	$c(v_1, p_3)$
	1/4	$v_5[p_3(0.195)], v_4[p_3(0.260)]$	$v_5 \rightarrow p_3$	v_4, v_6	p_3	$c(v_6, p_3)$
	1/5	$v_4[p_3(0.260)], v_6[p_3(0.285)]$	$v_4 \rightarrow p_3$	v_6	$\emptyset$	$c(v_2, p_3)$ $c(v_6, p_3)$
	2/1	$v_6[p_3(0.285)]$	$v_6 \rightarrow p_3$	$\emptyset$	p_1, p_2, p_3	$c(v_2, p_3)$ $c(v_4, p_3)$ $c(v_5, p_1)$

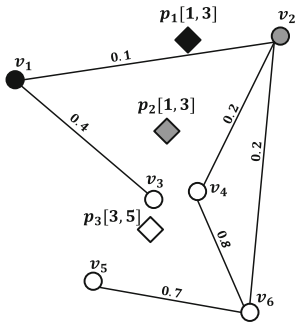

Fig. 6 Initial assignment for running example (LS framework)

(i.e., below their maximum capacity) are assigned users during Phase 2. Proposition 4 describes the complexity of $INIT$.

Proposition 4 *For $n << |V|$, $INIT$ has a time complexity of $O(\max(|V|n, |E||P|, |V||P|\log(|P|)))$, and a space requirement of $\Theta(|V||P|)$, where n, $|V|$, $|E|$, $|P|$ are the number of samples, nodes, edges and classes, respectively.*

5.2 Unilateral and bilateral deviations

After the initial assignment generated by $INIT$, our framework performs super-rounds. Each super-round contains rounds of unilateral and bilateral deviations. We first discuss the concrete algorithm for unilateral deviations for $GAME$, corresponding to Lines 3–5 of Fig. 4. Figure 7 illustrates the pseudo-code of the UNI function, which takes as input an initial assignment generated by $INIT$. It then performs a series of rounds until reaching a point where no player can deviate from his current assignment. A unilateral deviation for user v is allowed only if his current event $p = p_v$ exceeds its minimum capacity $\min_p$ (Line 4). In this case, the event $p' \neq p$ with the minimum cost $c(v, p')$ for v is retrieved from his heap H_v. If p' is full, UNI retrieves the next *cheaper* event, until finding one that can receive more users, or until encountering p (in which case all non-full events have cost higher than the current assignment, and there is no unilateral deviation for v at this round). If a deviation from p to p' occurs, Lines 9–11 update the costs of each friend f of v; specifically, the cost $c(f, p)$ increases by $\frac{1}{2} \cdot (1-\alpha) \cdot w(v, f)$ due to the departure of v, while $c(f, p')$ decreases by the same amount. For LS, the cost of the friends of v change in the same fashion, but without the $\frac{1}{2}$ factor.

Given that in practice $|P| << |V|$, to enhance efficiency, we perform bilateral deviations for pairs of events, instead of pairs of users. Consequently, in addition to the user heaps, utilized by $INIT$ and UNI, this phase exploits event pair heaps. Specifically, for each pair of events p_i, p_j, there is a min-heap EP_{p_i,p_j} that contains, for every user v currently assigned to p_i, the cost change incurred by moving to p_j: $\delta c_v(p_i, p_j) = c(v, p_j) - c(v, p_i)$. The top contains the user

Input: Geo-social Graph $G = (V, E, W)$
Set P of events with constraints $min_p, max_p, \forall p \in P$
Initial assignment of users V to events P
$d(v, p), \forall v \in V, \forall p \in P$
Output: UNI assignment

```
1:  repeat
2:      for each user v ∈ V
3:          p ← p_v
4:          if |p| > min_p
5:              repeat
6:                  p' ← get next(H_v)
7:                  if p' ≠ p and |p'| < max_p'
8:                      p_v ← p'; |p'| ← |p'| + 1; |p| ← |p| − 1
9:                      for each friend f of v
10:                         c(f, p) ← c(f, p) + 1/2 · (1 − α) · w(v, f)
11:                         c(f, p') ← c(f, p') − 1/2 · (1 − α) · w(v, f)
12:                     Goto Line 2
13:             until p' = p_v
14: until no player can unilaterally improve his local cost
```

Fig. 7 UNI function

with the minimum $\delta c_v(p_i, p_j)$, which may be positive or negative (if the swap benefits v). Each user v exists in a single heap $EP_{p_v,p}$, which contains $|P|$ entries with the assignment costs of v to all events. Therefore, the total space requirement for all the event heaps is $|V||P|$, i.e., the same as the user heaps and the cost table.

Figure 8 illustrates the BI function for $GAME$ that corresponds to Lines 6–8 of the general framework of Fig. 4. For each pair of events p_i, p_j, we obtain the users v and u at the top of the heaps EP_{p_i, p_j} and EP_{p_j, p_i}, respectively. Let $\Delta_v \leftarrow \delta c_v(p_i, p_j)$, and $\Delta_u \leftarrow \delta c_u(p_j, p_i)$. In $GAME$, swapping the events of v and u should benefit at least one user and should not increase the cost of either u or v (Line 9). However, if v and u are friends, Δ_v (resp. Δ_u) must increase by $\frac{1}{2} \cdot (1-\alpha) \cdot w(v, u)$ to take into consideration the departure of u (v) from p_j (p_i). If the swapping occurs, Lines 11–16 update the costs of the friends of v and u for both events. Finally, if v and u are friends, we also have to update their costs for both events (Lines 17–21), e.g., $c(v, p_i)$ decreases because of the inclusion of u in p_i. The function for LS is similar, except that (i) the condition at Line 9 is $\Delta_v + \Delta_u < 0$

Input: Geo-social Graph $G = (V, E, W)$
Set P of events with constraints $min_p, max_p, \forall p \in P$
Assignment of users V to events P
$d(v, p), \forall v \in V, \forall p \in P$
Output: Pairwise stable assignment

1: **repeat**
2: **for** each event p_i
3: **for** each event $p_j \neq p_i$
4: $v \leftarrow top(EP_{p_i,p_j}), u \leftarrow top(EP_{p_j,p_i})$
5: $\Delta_v \leftarrow \delta c_v(p_i, p_j), \Delta_u \leftarrow \delta c_u(p_j, p_i)$
6: **if** v and u are friends
7: $\Delta_v \leftarrow \Delta_v + \frac{1}{2} \cdot (1-\alpha) \cdot w(v,u)$
8: $\Delta_u \leftarrow \Delta_u + \frac{1}{2} \cdot (1-\alpha) \cdot w(v,u)$
9: **if** $\Delta_v \leq 0$ and $\Delta_u \leq 0$ and ($\Delta_v < 0$ or $\Delta_u < 0$)
10: $p_v \leftarrow p_j, p_u \leftarrow p_i$
11: **for** each friend f_v of v
12: $c(f_v, p_j) \leftarrow c(f_v, p_j) - \frac{1}{2} \cdot (1-\alpha) \cdot w(v, f_v)$
13: $c(f_v, p_i) \leftarrow c(f_v, p_i) + \frac{1}{2} \cdot (1-\alpha) \cdot w(v, f_v)$
14: **for** each friend f_u of u
15: $c(f_u, p_i) \leftarrow c(f_u, p_i) - \frac{1}{2} \cdot (1-\alpha) \cdot w(u, f_u)$
16: $c(f_u, p_j) \leftarrow c(f_u, p_j) + \frac{1}{2} \cdot (1-\alpha) \cdot w(u, f_u)$
17: **if** v and u are friends
18: $c(v, p_i) \leftarrow c(v, p_i) - \frac{1}{2} \cdot (1-\alpha) \cdot w(v,u)$
19: $c(u, p_i) \leftarrow c(u, p_i) + \frac{1}{2} \cdot (1-\alpha) \cdot w(v,u)$
20: $c(v, p_j) \leftarrow c(v, p_j) + \frac{1}{2} \cdot (1-\alpha) \cdot w(v,u)$
21: $c(u, p_j) \leftarrow c(u, p_j) - \frac{1}{2} \cdot (1-\alpha) \cdot w(v,u)$
22: **until** Pairwise stability

Fig. 8 BI function

since the swap should decrease the total cost of both users, and (ii) the $\frac{1}{2}$ factor is removed from all costs.

Figure 9 continues the example of Fig. 6, assuming the local search framework. The first two rows of the table illustrate the first two rounds of UNI in the first super-round. The *Heaps(top)* column contains the top event in each user heap. Users in bold are currently assigned to their best event; thus, they will not perform unilateral deviations. For instance, v_4, v_5, and v_6 are in p_3 with cost 0.06, 0.02, and 0.085, respectively. Moreover, at round 1 of UNI, v_1 and v_2 cannot deviate from their current events p_1 and p_2 because there are no other users in these events and they would underflow. On the other hand, for v_3, currently assigned at p_3, the best event would be p_1; although it is further than p_3, its overall cost is lower because v_1 (v_3's friend) is there. Thus, v_3 moves to p_1; the switch does not violate $\min_{p_3} = 3$ or $\max_{p_1} = 3$. Accordingly, the cost of v_1 increases for p_3 and decreases for p_1. At the beginning of round 2, all users, except v_2, are already assigned to the event with the lowest cost (i.e., they are bold). User v_2 still cannot deviate from p_2; therefore, UNI stops and the super-round proceeds with bilateral deviations.

The third and fourth rows of Fig. 9 illustrate the first two rounds of BI. The *Heaps(top)* column now contains the top user in each event pair heap. Each line contains the pairs of users that are candidates for swapping events, e.g., in the first line of row 3, users v_3 and v_2 are candidates to exchange events, as they are in the top of the event pair heaps EP_{p_1,p_2} and EP_{p_2,p_1}, respectively. If v_3 and v_2 swap events, the cost change is $\Delta v_3 + \Delta v_2 = 0.06 - 0.064 = -0.004 < 0$. Since the total cost decreases, the swap occurs and the costs of the friends of v_3 and v_2 for p_1 and p_2 change accordingly. Specifically, the updated costs are in the last column of row 3. The other candidate pairs (v_3, v_5), (v_2, v_4) for swap would incur positive cost change ($0.051+0.243$ and $-0.07+0.158$, respectively), and the round terminates without other swaps.

At the second round of BI, the top users of EP_{p_1,p_2} and EP_{p_2,p_1} are v_1 and v_3. Although they both have negative cost change by swapping events ($\Delta v_1 = -0.063$, $\Delta v_2 = -0.06$), because they are friends, their cost changes need to be recalibrated to $\Delta v_1 \rightarrow \Delta v_1 + (1-0.5) \cdot 0.4 = 0.137$ and $\Delta v_2 \rightarrow \Delta v_2 + (1-0.5) \cdot 0.4 = 0.136$ (Lines 10–12 of Fig. 8). Consequently, there is no swap and super-round 1 terminates with the graph shown on the right of Fig. 9. The second super-round performs a single unilateral deviation of v_1 from p_1 to p_2, generating the final solution of Fig. 1.

For simplicity, when describing UNI and BI in Figs. 7 and 8 we omit the heap operations. Specifically, when the cost $c(v, p)$ of assigning user v to event p is updated, the change must be reflected to the user heap H_v and the event pair heap $EP_{p_v,p}$. Recall that each H_v has size $|P|$, while the maximum size of $EP_{p_v,p}$ is $|V|$ if all users are assigned to p_v. The following result discusses the time complexity of a single round:

Proposition 5 *The time complexity of a super-round of unilateral and bilateral deviations is* $O(|P|^2|V|(\log(|P|) + \log(|V|)))$.

Since each deviation decreases a global function, LS and $GAME$ are guaranteed to terminate after a finite number of super-rounds. Although this number depends on the assignment and edge costs, and is pseudo-polynomial, in our experiments with diverse datasets both algorithms always perform at most twelve super-rounds.

6 Special cases and additional constraints

So far, we have assumed that all users can be assigned to some event, and all events can reach their minimum capacity, which may not always hold. Assume first the case where the number of users exceeds the sum of the event capacities, i.e., $|V| > \sum_{p \in P} \max_p$, so that some users remain unassigned. We are then interested in the assignment that (i) fills the capacities of all events, and (ii) minimizes the total cost, excluding the similarity and social cost of unassigned users.

Our framework can be modified to deal with this scenario. In particular, we first run the initial assignment algorithm to fill all $|P|$ events with $\sum_{p \in P} \max_p$ users. The remaining $|V| - \sum_{p \in P} \max_p$ users are assigned to a new *fictitious event*

Super-round/Function/Round	Heaps (top)	Deviations	Changes
$1/UNI/1$	$v_1[p_3(0.079)], v_2[p_3(0.095)], v_3[p_1(0.058)]$ $\boldsymbol{v_4}[p_3(0.06)], \boldsymbol{v_5}[p_3(0.02)], \boldsymbol{v_6}[p_3(0.085)]$	$v_3 : p_3 \rightarrow p_1$	$c(v_1, p_1)$ $c(v_1, p_3)$
$1/UNI/2$	$\boldsymbol{v_1}[p_1(0.039)], v_2[p_3(0.095)], \boldsymbol{v_3}[p_1(0.058)]$ $\boldsymbol{v_4}[p_3(0.06)], \boldsymbol{v_5}[p_3(0.02)], \boldsymbol{v_6}[p_3(0.085)]$		
$1/BI/1$	$EP_{p_1,p_2} : \{\boldsymbol{v_3}(0.06)\}, EP_{p_2,p_1} : \{v_2(-0.064)\}$ $EP_{p_1,p_3} : \{\boldsymbol{v_3}(0.051)\}, EP_{p_3,p_1}\{\boldsymbol{v_5}(0.243)\}$ $EP_{p_2,p_3}\{v_2(-0.07)\}, EP_{p_3,p_2} : \{\boldsymbol{v_4}(0.158)\}$	$v_3 \leftrightarrow v_2$	$c(v_1, p_1), c(v_1, p_2)$ $c(v_4, p_1), c(v_4, p_2)$ $c(v_6, p_1), c(v_6, p_2)$
$1/BI/2$	$EP_{p_1,p_2}\{v_1(-0.063)\}, EP_{p_2,p_1}\{v_3(-0.06)\}$ $EP_{p_1,p_3} : \{v_2(-0.006)\}, EP_{p_3,p_1} : \{v_4(0.197)\}$ $EP_{p_2,p_3} : \{v_3(-0.009)\}, EP_{p_3,p_2} : \{v_5(0.204)\}$		
$2/UNI/1$	$v_1[p_2(0.05)], v_2[p_3(0.095)], v_3[p_1(0.058)]$ $\boldsymbol{v_4}[p_3(0.06)], \boldsymbol{v_5}[p_3(0.02)], \boldsymbol{v_6}[p_3(0.085)]$	$v_1 : p_1 \rightarrow p_2$	$c(v_2, p_1), c(v_3, p_1)$ $c(v_2, p_2), c(v_3, p_2)$
$2/UNI/2$	$\boldsymbol{v_1}[p_2(0.05)], v_2[p_3(0.095)], \boldsymbol{v_3}[p_2(0.019)]$ $\boldsymbol{v_4}[p_3(0.06)], \boldsymbol{v_5}[p_3(0.02)], \boldsymbol{v_6}[p_3(0.085)]$		
$2/BI/1$	$EP_{p_1,p_2} : \{v_2(0.014)\}, EP_{p_2,p_1} : \{\boldsymbol{v_1}(0.063)\}$ $EP_{p_1,p_3} : \{v_2(-0.031)\}, EP_{p_3,p_1} : \{\boldsymbol{v_4}(0.197)\}$ $EP_{p_2,p_3} : \{\boldsymbol{v_3}(0.09)\}, EP_{p_3,p_2}\{\boldsymbol{v_5}(0.204)\}$		

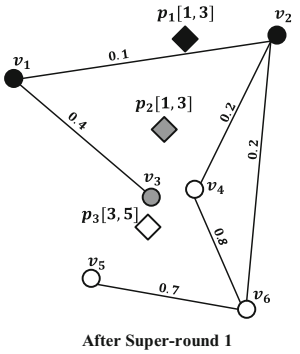

Fig. 9 Unilateral and bilateral deviations for running example (LS framework)

p_f with capacity $|V| - \sum_{p \in P} \max_p$. The total cost of users assigned to p_f is set to 0; that is, users in p_f do not contribute to the total cost through their similarity cost or through their social cost of their friendships. This means that only users in the original set of events are taken into account in the total assignment cost. It is important that p_f does not have spare capacity, otherwise users in P would unilaterally deviate to p_f to decrease their cost to 0, violating the requirement that we must assign as many users as possible to the set P. After initialization, all users are assigned to an event, including p_f, and all events are full. For pairs of users assigned to distinct events in P, we perform bilateral deviations as usual, using either the game-theoretic or the local search framework. On the other hand, for bilateral deviations involving a pair of users (v, u), where $p_v \in P$ and $p_u = p_f$, the game-theoretic pairwise stability cannot work since the user u will have a local cost greater than 0 in any event other than p_u. Instead, we can employ local search by comparing the cost of v (w.r.t. events in P) to the cost of u if the latter deviates to p_v from p_f. If the new cost is lower, then v and u swap. In this way, users unassigned during $INIT$ can now be assigned if this drops the total cost.

In the opposite extreme, when the number of users is very small, i.e., $|V| < \sum_{p \in P} \min_p$, not all events can be filled to their minimum capacity; as a result, certain events will remain empty. Since, however, it is not known in advance which events should be empty in the optimal solution, computing the best events to fill is a hard combinatorial problem. SEOG [15] tackles this problem by introducing phantom events, and greedily identifying the events to be filled that will result in lower total cost. The agent-based framework can play an interesting complementary role to SEOG in this scenario. Concretely, a two-phase algorithm can first decide greedily on the active events (which will be filled to their minimum capacity) (e.g., according to SEOG), whereas a second agent-based phase can refine the SEOG solution to further lower the total cost.

Another extension of the original model focuses on additional constraints on the user side. Concretely, we now assume that subsets of users (e.g., pairs and triplets) must be assigned together, else their cost will be infinite. For instance, a couple may wish to either attend the same event, or attend no event at all. To deal with this case, we replace the set of constrained nodes in the original graph by a new *hypernode*, whose similarity cost from an event is equal to the sum of the distances of its nodes, and whose social cost from any outside node is equal to the sum of the social cost of its constituent nodes to that node. Moreover, we associate the hypernode with a *size*, which equals the number of nodes therein. Unconstrained nodes are also referred to as *simple* nodes, and have a size of one.

Having created all hypernodes, we can then perform the unilateral and bilateral deviations, albeit with one caveat. For unilateral deviations involving hypernodes, we must make sure that (i) the remaining capacity of the new event, where we move the hypernode to, is at least equal to the hypernode size, and (ii) the old event has still sufficient number of users. For bilateral deviations, the hypernodes being swapped should not violate the capacity constraints. To ensure this, we can only allow swaps of hypernodes with the same size. The downside is that a limited number of bilateral deviations is considered, which may compromise the solution quality. An alternative is to swap a set of hypernodes in one event with another set of hypernodes/simple nodes in another event, so that the two sets have the same size. It can be easily shown that the second approach relies on the knapsack problem, which is NP-hard. A compromise is to swap hypernodes of size $s > 1$ with hypernodes of the same size s, or with s simple nodes.

Finally, we discuss the initial assignment in the presence of hypernodes. First, note that it may not be possible to fit all

users into the events, even if the sum of capacities equals the number of users. For instance, assume three hypernodes of sizes 3, 3, 2, and two events with the same maximum capacity of 4; it is easy to verify that there is no solution that assigns all hypernodes to the two events. We can handle this by maintaining a separate fictitious event to include hypernodes that do not fit in any event. Moreover, we can assign hypernodes in decreasing size order, so that the larger hypernodes, which are more difficult to fit, are assigned first.

7 Experiments

For our experimental evaluation, we use the *Gowalla* and *Foursquare* datasets. *Gowalla* [13] contains 12,748 users, connected through 48,419 edges, who checked-in at Austin and Dallas during a weekend in February 2009. For the same time and place, we collected 128 social events from *Eventbrite*.[5] *Foursquare* [14] contains 2,153,371 users and 1,143,092 events/venues, over the world in September 2013. The number of edges is 27,098,490. In both datasets, the weight of all friendships is equal to 1. When we fix the number of events, we randomly select a subset from the corresponding dataset. Capacity constraints are created following the methodology of [15]: the value of $\max_p$ for every event is sampled from a normal distribution with mean $2 \cdot \frac{|V|}{|P|}$ and variance $\frac{|V|}{|P|}$, while the value of $\min_p$ is randomly selected from the interval $[1, \max_p]$. Unless otherwise stated, we set the parameter α of Eq. (1) to 0.5, so that the similarity and social costs have equal weights. In addition, in order to make the similarity and social costs comparable, we follow the *pessimistic normalization* technique of [5]. All the algorithms were implemented in C++ under Linux Ubuntu, and executed on an Intel Xeon E5-2660 2.20 GHz with 16 GB RAM. Section 7.1 evaluates the behavior of the proposed algorithms, and Sect. 7.2 compares them against the state of the art.

7.1 Behavior of proposed algorithms

The first set of experiments assesses the effect of the sample size n on the solution quality and running time. Figure 10 illustrates the cost (i.e., quality) of solutions versus n, for $|P| = 32$ of events. $INIT$ corresponds to the initial assignment function of Sect. 5.1, while $GAME$ and LS are the game-theoretic and local search approaches, respectively. The accuracy converges to its maximum value for $n \geq 8$ samples in both datasets despite their diverse cardinality and characteristics. The application of the agent-based framework, and especially LS, improves the quality of the initial

[5] https://www.eventbrite.com.

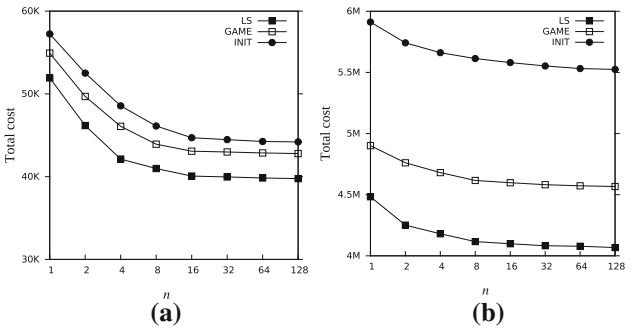

Fig. 10 Quality versus sample size n ($|P| = 32$). **a** *Gowalla*. **b** *Foursquare*

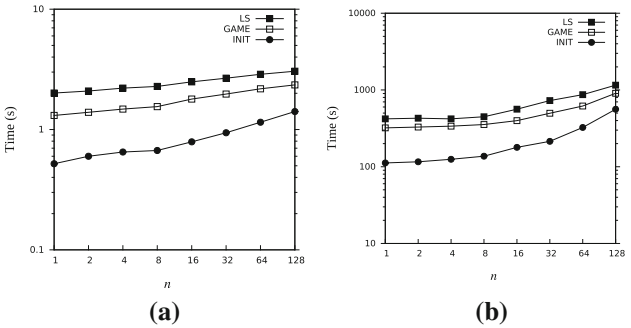

Fig. 11 Time versus sample size n ($|P| = 32$). **a** *Gowalla*. **b** *Foursquare*

assignment significantly. Figure 11 plots the running time of the algorithms (in seconds) versus the sample size n. As expected, the execution time of $INIT$ increases with n. Since $INIT$ is a component of both $GAME$ and LS, the time of those algorithms also grows, and $INIT$ eventually constitutes a large fraction of the total cost.

In order to investigate the effect of the event cardinality, in Figs. 12 and 13 we plot the quality cost and the running time of $INIT$ as a function of n and $|P|$ for *Gowalla*. The diagrams for *Foursquare* are similar and omitted. For all values of P in our evaluation ($8 \leq |P| \leq 128$), exceeding $n = 8$ samples does not offer significant quality improvement, while it incurs considerable time overhead (observe the logarithmic scale for time). Since $n = 8$ provides the best trade-off between accuracy and efficiency, in the following experiments we always use eight samples.

Figure 14 plots the number of unilateral (gray columns) and bilateral (white columns) rounds per super-round for a random execution of $GAME$ and LS with $|P| = 32$ events and $n = 8$ samples. $GAME$ and LS converge in three and nine super-rounds, respectively, in *Gowalla*, and in three and twelve super-rounds in *Foursquare*. The fraction on top of each column denotes the total number of bilateral/unilateral deviations during that super-round. For instance, in *Gowalla* in the first super-round of $GAME$, there were 323 bilateral and 479 unilateral deviations performed in 117 bilateral and

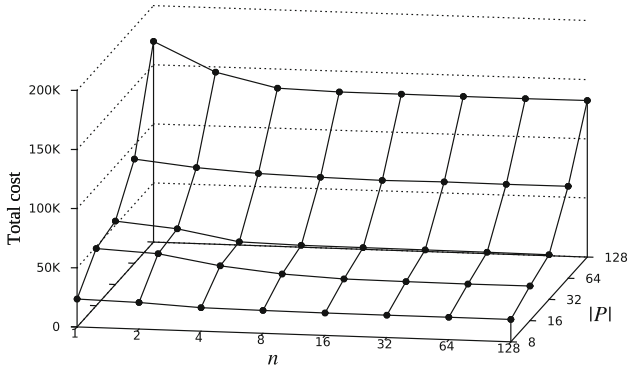

Fig. 12 Quality versus n versus $|P|$ ($Gowalla$)

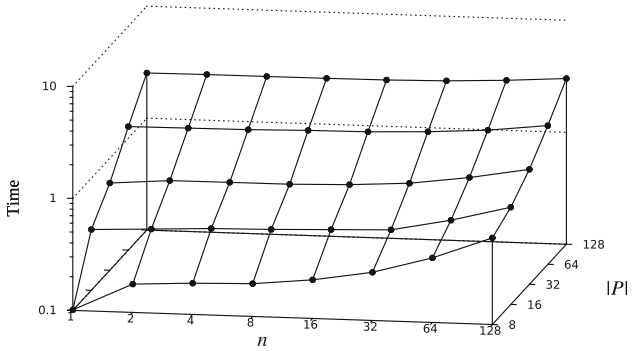

Fig. 13 Time versus n versus $|P|$ ($Gowalla$)

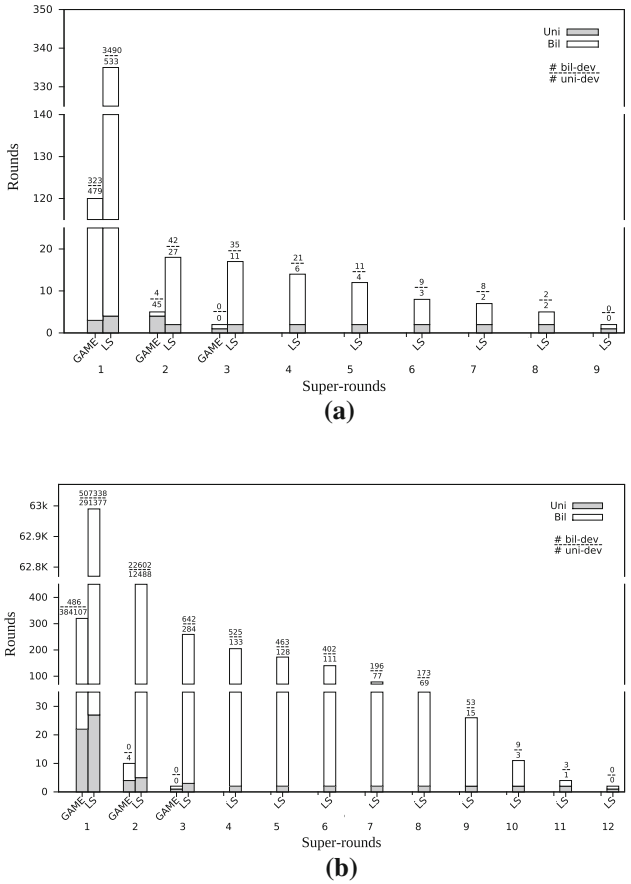

Fig. 14 Rounds (deviations) versus super-rounds ($|P| = 32$). **a** $Gowalla$. **b** $Foursquare$

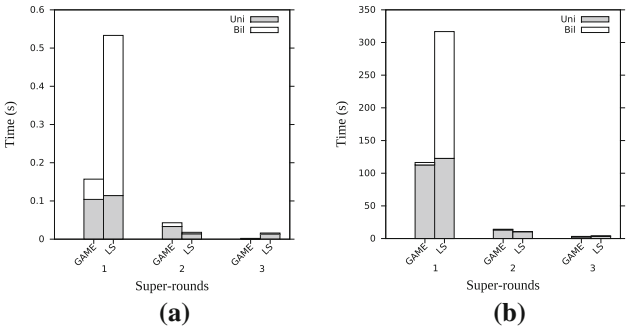

Fig. 15 Time versus super-rounds ($|P| = 32$). **a** $Gowalla$. **b** $Foursquare$

3 unilateral rounds. Since each deviation drops the cost, the solution improvement is analogous to the number of deviations.

Figure 15 illustrates the running time of $GAME$ and LS per super-round for the experiment of Fig. 14. We only display the three first super-rounds because $GAME$ terminates, whereas in LS the time of subsequent super-rounds is negligible. The gray (resp. white) column corresponds to unilateral (bilateral) rounds. Despite the fact that as shown in Fig. 14 there are more bilateral rounds, in most super-rounds the running time is dominated by the unilateral rounds because they iterate over $|V|$ users, as opposed to $|P|^2$ event pairs. This is more obvious in $Foursquare$ since $|V| = 2153371 >> |P|^2 = 32^2$, justifying the use of event-pairs, instead of user-pairs, in the function BI for bilateral deviations. The first super-round is always the most expensive as it incurs the most deviations.

The last experiment evaluates the effectiveness of $INIT$ against other initialization methods. Specifically, we implemented two benchmarks: (i) assign each user to a random event, and (ii) assign each user to the closest event. Similar to $INIT$, both initializations involve two phases in order to fill the minimum capacity constraints of all the events. After initialization, they apply UNI and BI for unilateral and bilateral deviations. The corresponding techniques are denoted as $GAME_R$, LS_R ($GAME_C$, LS_C) for random (closest event) assignments.

Figure 16a assesses the quality of $GAME$ (with the initial assignment $INIT$), $GAME_R$, and $GAME_C$ as a function of the super-rounds, for $Gowalla$. Figure 16b repeats the experiment for LS and $Foursquare$. Super-round 0 corresponds to the initial assignment. Random assignment is by far the worst in both datasets, followed by the closest event assignment. Although the agent-based framework improves

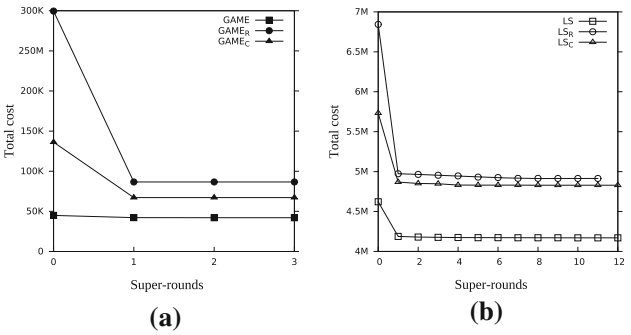

Fig. 16 Quality versus super-rounds ($|P| = 32$). **a** $GAME$, $Gowalla$. **b** LS, $Foursquare$

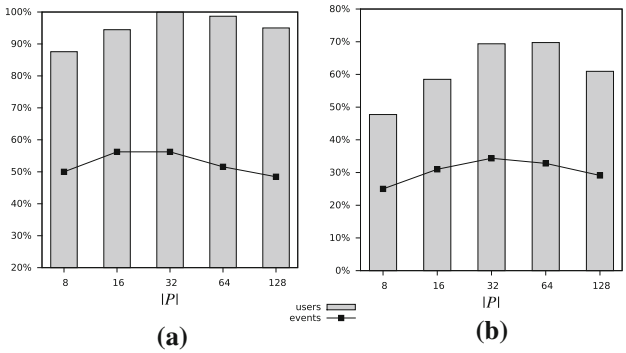

Fig. 17 SEOG assignments (satisfied vs. $|P|$). **a** $Gowalla$. **b** $Foursquare$

Table 2 Running time (s) ($|P|$=32)

	Gowalla	*Foursquare*
$Random$	0.006	3.5
$ClosestEvent$	0.07	12.9
$INIT$	0.53	140
$GAME_R$	2.83	482
$GAME_C$	3.22	465
$GAME$	2.91	497
LS_R	3.24	511
LS_C	3.47	481
LS	3.37	521

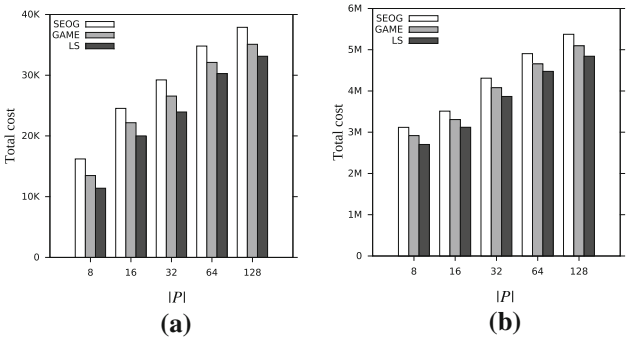

Fig. 18 Quality versus $|P|$ ($\alpha = 0.5$). **a** $Gowalla$. **b** $Foursquare$

significantly their solutions, they cannot reach the quality of $INIT$. In all cases, the largest improvement occurs at super-round 1, which, as shown in the previous diagrams, incurs the most rounds, deviations and longest running time. Table 2 shows the running time of the initial assignments and the complete algorithms for the same experiment. Even though the random and closest event assignments are much faster than $INIT$, the execution times of the complete algorithms are similar because the poor initial solutions yield more deviations during the super-rounds.

7.2 Comparison to state of the art

We compare $GAME$ and LS against $SEOG$, the best algorithm for the *social event organization* problem [15]. As mentioned in Sect. 2.1, for SEO, *only events with at least one user* need to be assigned users. Consequently, $SEOG$ may leave some events empty and some users unassigned, even though there are enough users and sufficiently large capacities. Figure 17 illustrates the percentage of assigned users (column) and events that are within the capacity constraints, as a function of the number of events $|P|$. For instance, in $Gowalla$, when $|P| = 8$, 87% of the users were assigned and just 50% of the events had a number of users within their capacity constraints. Those values drop to 47% for users and 25% for events in $Foursquare$ and $|P| = 8$. On the other hand, $GAME$ and LS are not included in the diagrams because they assign all users and fill each event to a legal capacity value in all cases.

In addition to maximizing the number of assignments, the proposed algorithms achieve solutions of better quality and are significantly faster than $SEOG$. In order to avoid the unassigned users and empty events of $SEOG$, for the following experiments we set the minimum capacity constraints of all the events to zero. The maximum constraints are again created using the method of [15], as in the previous experiments. We generate 30 problem instances with the same events, but different maximum capacities, and report the mean over all instances.

Figure 18 plots the solution quality of $GAME$ and LS against $SEOG$, as a function of $|P|$ ranging from 8 to 128. In all cases, LS generates the best solutions, followed by $GAME$. An interesting observation is that the total cost increases with the number of events. This can be explained by the fact that, when there are numerous events, friends are more likely to be divided, increasing the total social cost.

Figure 19 shows the running time of the algorithms versus the number of events. $GAME$ is the fastest algorithm, followed by LS, which as shown in the experiments of the previous section, performs more super-rounds and deviations than $GAME$. $SEOG$ is by far the slowest algorithm

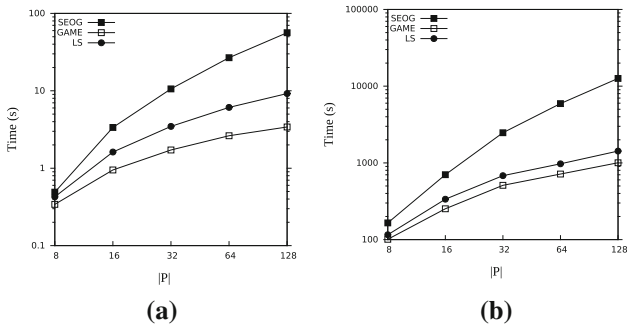

Fig. 19 Time versus $|P|$ ($\alpha = 0.5$). **a** *Gowalla*. **b** *Foursquare*

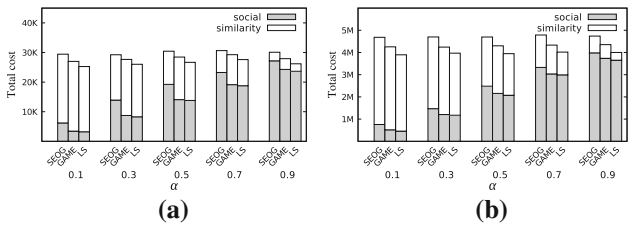

Fig. 20 Quality versus α ($|P| = 32$). **a** *Gowalla*. **b** *Foursquare*

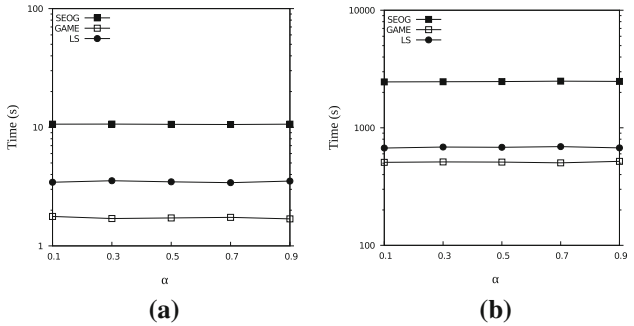

Fig. 21 Time versus α ($|P| = 32$). **a** *Gowalla*. **b** *Foursquare*

in all settings. For example, in the largest problem instance (*Foursquare*, $|P| = 128$), *GAME* and *LS* terminate in 16 and 23 minutes, respectively, whereas SEOG requires 3.5 hours. This is because the greedy algorithm of SEOG uses a large heap of size $|V| \cdot |P|$ that needs to be updated numerous times. In contrast, our algorithms use smaller heaps (for each user and pair of events) that simplify the update process.

Figure 20 illustrates the quality cost versus the value of input parameter α that adjusts the relative importance of the similarity and social cost; the weight of the similarity cost is proportional to α. Accordingly, as α increases, the contribution of the similarity (white part of each column) to the total cost decreases, since its minimization becomes the main focus. On the other hand, the social cost increases, and the total cost remains rather stable for all values of α. Similar to Fig. 18, *GAME* and *LS* always produce solutions of better quality (lower cost) than *SEOG*. As shown in Fig. 21, the running time of all methods is insensitive to α because the algorithms perform the same operations independently of α. The proposed methods outperform *SEOG* by a wide margin.

8 Conclusion

Constrained graph partitioning (CGP) is becoming increasingly important with the proliferation of social networks and related services. In this paper, we introduce an effective and efficient agent-based framework for CGP that unifies the game-theoretic and the local search perspectives, and present concrete implementations that integrate various optimizations to enhance performance. We investigate the behavior of our algorithms through an extensive experimental evaluation with real datasets. The proposed *GAME* and *LS* methods outperform the state-of-the-art *SEOG* in effectiveness, solution quality and efficiency. In terms of effectiveness, *GAME* and *LS* always achieve the maximum number of assignments, by respecting the minimum capacities of all classes and not leaving empty events. They both generate solutions of higher quality than *SEOG*, with *LS* reaching up to 20% improvement. Finally, they are both significantly faster than *SEOG*, with the performance gains of *GAME* exceeding an order of magnitude in some settings.

In the future, we plan to investigate additional search heuristics to improve the efficiency or solution quality of the proposed techniques. For instance, a hill climbing variant would perform, at each round, the unilateral or bilateral deviation that incurs the largest cost drop. We could also explore trilateral or deviations of higher degree, which could improve the solution quality at the expense of running time (which is exponential to the number of nodes involved). Alternatively, we could apply a form of simulated annealing that allows deviations to solutions of worse quality in order to escape local minima. Finally, in our scenario, we assume that each node is assigned to a single class. An interesting extension would allow multiple assignments, e.g., a user could attend different nearby events. A possible way to formulate this problem is by associating each user with the number of events that he is willing to attend. An alternative is to define a distance budget B_v per user v, so that v can only be assigned to some event p, only if the distance $d(v, p)$ between v and p is at most B_v. When such an assignment occurs, $d(v, p)$ is deducted from B_v. Additional assignments can be made, provided that they do not exceed the remaining budget. In either case, this version of CGP would necessitate novel techniques based on the problem definition.

Funding This work was supported by GRF grants 16207914 and 16231216 from Hong Kong RGC.

9 Appendix

Lemma 1 *The difference in Potential Function (3) $\Delta\Phi_{UNI}(v, p_v, p'_v, \overline{\boldsymbol{p}_v})$ due to a unilateral deviation in assignment $\boldsymbol{p}$ of player v from event p_v to p'_v, while the rest of the players do not deviate, is:*

$$\begin{aligned}\Delta\Phi_{UNI}(v, p_v, p'_v, \overline{\boldsymbol{p}_v}) &= \Phi(p'_v, \overline{\boldsymbol{p}_v}) - \Phi(p_v, \overline{\boldsymbol{p}_v}) \\ &= c_v(p'_v, \overline{\boldsymbol{p}_v}) - c_v(p_v, \overline{\boldsymbol{p}_v}) \\ &= \Big(\alpha \cdot d(v, p'_v) + (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \\ \wedge p_f = p_v}} \frac{1}{2} \cdot w(v, f)\Big) \\ &- \Big(\alpha \cdot d(v, p_v) + (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \\ \wedge p_f = p'_v}} \frac{1}{2} \cdot w(v, f)\Big)\end{aligned}$$

Proof After player v deviates from event p_v to p'_v, the change in the potential function is:

$$\begin{aligned}\Delta\Phi_{UNI}(v, p_v, p'_v, \overline{\boldsymbol{p}_v}) &= \Phi(p'_v, \overline{\boldsymbol{p}_v}) - \Phi(p_v, \overline{\boldsymbol{p}_v}) \\ &= \Big(\alpha \cdot d(v, p'_v) - \alpha \cdot d(v, p_v)\Big) \\ &+ \Big((1-\alpha) \cdot \sum_{\substack{(v,f)\in E \\ \wedge p_f \neq p'_v}} \frac{1}{2} \cdot w(v, f) \\ &- (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \\ \wedge p_f \neq p_v}} \frac{1}{2} \cdot w(v, f)\Big)\end{aligned} \tag{10}$$

This result is directly from Eq. (3), as only v and his friends are affected by v's deviation; all the other terms in the potential function are canceled. Additionally, before and after the unilateral deviation, according to Eq. (4) we have:

$$\begin{aligned}c_v(p_v, \overline{\boldsymbol{p}_v}) &= \alpha \cdot \sum_{v\in V} d(v, p_v) \\ &+(1-\alpha) \cdot \sum_{\substack{(v,f)\in E \wedge \\ p_v \neq p_f}} \frac{1}{2} \cdot w(v, f)\end{aligned} \tag{11}$$

$$\begin{aligned}c_v(p'_v, \overline{\boldsymbol{p}_v}) &= \alpha \cdot \sum_{v\in V} d(v, p_v) \\ &+(1-\alpha) \cdot \sum_{\substack{(v,f)\in E \wedge \\ p'_v \neq p_f}} \frac{1}{2} \cdot w(v, f)\end{aligned} \tag{12}$$

If we subtract Eq. (11) from Eq. (12), we have Eq. (10), therefore proving that the difference in the potential function equals the difference in the cost of the player who deviates. Note that in Eq. (3) and also in the above equations, we consider the friends of v that are assigned in different events than p_v or p'_v ($p_f \neq p_v$ and $p_f \neq p_v$). In contrast, to calculate the cost difference $\Delta\Phi_{UNI}$ in Eq. (3), we only consider the friends of v that are assigned to p_v or p'_v ($p_f = p_v$ and $p_f = p'_v$). It is straightforward that the friends of v that we need to consider during the calculation of his cost difference, are those assigned to different events than p_v (resp. p'_v) and especially those assigned to p'_v (resp. p_v). This is because the weights of friends assigned to events other than p'_v (resp. p_v) are canceled out. □

Lemma 2 *Assume v, u swap events in assignment $\boldsymbol{p}$, while the rest of the players do not deviate. For the new assignment $(p'_v, p'_u, \overline{\boldsymbol{p}_{vu}})$, where $p'_v = p_u$ and $p'_u = p_v$, the difference $\Delta\Phi_{BI}(v, u, \boldsymbol{p})$ in Potential Function (3) is:*

$$\begin{aligned}\Delta\Phi_{BI}(v, u, \boldsymbol{p}) &= \Phi(p'_v, p'_u, \overline{\boldsymbol{p}_{vu}}) - \Phi(p_v, p_u, \overline{\boldsymbol{p}_{vu}}) \\ &= \Big(c_v(p'_v, p_u \cup \overline{\boldsymbol{p}_{vu}}) - c_v(p_v, p_u \cup \overline{\boldsymbol{p}_{vu}}) + \frac{1}{2}w(v, u)\Big) \\ &+ \Big(c_u(p'_u, p_v \cup \overline{\boldsymbol{p}_{vu}}) - c_u(p_u, p_v \cup \overline{\boldsymbol{p}_{vu}}) + \frac{1}{2}w(v, u)\Big)\end{aligned}$$

Proof In order to express the bilateral deviation as two unilateral deviations (v to p_u and u to p_v), we add and deduct the term $\Phi(p'_v, p_u, \overline{\boldsymbol{p}_{vu}})$ to/from the difference in the potential function:

$$\begin{aligned}\Delta\Phi_{BI}(v, u, \boldsymbol{p}) &= \Phi(p'_v, p'_u, \overline{\boldsymbol{p}_{vu}}) - \Phi(p_v, p_u, \overline{\boldsymbol{p}_{vu}}) \\ &= \Big(\Phi(p'_v, p_u, \overline{\boldsymbol{p}_{vu}}) - \Phi(p_v, p_u, \overline{\boldsymbol{p}_{vu}})\Big) \\ &+ \Big(\Phi(p'_v, p'_u, \overline{\boldsymbol{p}_{vu}}) - \Phi(p'_v, p_u, \overline{\boldsymbol{p}_{vu}})\Big)\end{aligned} \tag{13}$$

Each of the two terms of the sum in Eq. (13) represents a deviation of a single player. Specifically, the first sum is the unilateral deviation of v, while the second sum is the unilateral deviation of u, supposing that v deviates first and u follows right after.[6] Based on Eq. (2), Eq. (13) becomes:

$$\begin{aligned}\Delta\Phi_{BI}(v, u, \boldsymbol{p}) &= \Big(c_v(p'_v, p_u \cup \overline{\boldsymbol{p}_{vu}}) - c_v(p_v, p_u \cup \overline{\boldsymbol{p}_{vu}})\Big) \\ &+ \Big(c_u(p'_u, p'_v \cup \overline{\boldsymbol{p}_{vu}}) - c_u(p_u, p'_v \cup \overline{\boldsymbol{p}_{vu}})\Big)\end{aligned} \tag{14}$$

However, since $p'_v = p_u$, $p'_u = p_v$ and $p_v \neq p_u$, for the second term of the sum in Eq. (14) we will have:

$$c_u(p'_u, p'_v \cup \overline{\boldsymbol{p}_{vu}}) = c_u(p'_u, p_v \cup \overline{\boldsymbol{p}_{vu}}) + \frac{1}{2}w(v, u) \tag{15}$$

$$c_u(p_u, p'_v \cup \overline{\boldsymbol{p}_{vu}}) = c_u(p_u, p_v \cup \overline{\boldsymbol{p}_{vu}}) - \frac{1}{2}w(v, u) \tag{16}$$

Equation (15) implies that u's cost for his new strategy ($p'_u = p_v$), *after* v has changed strategy (v switched to $p'_v = p_u$),

[6] It is trivial to show that the proof also stands if we suppose that u deviates first.

is equal to u's cost for p'_u *before* v changed strategy plus their friendship weight (if they are friends, otherwise the friendship weight is zero); this is because after v changed strategy, u (may) lost a friend from p'_u; thus, his cost for p'_u needs to be increased. In a similar manner, Eq. (16), means that u's cost for his old strategy (p_u), equals to u's cost for p_u *before* v changed strategy minus their friendship weight (because v will switch to $p'_v = p_u$ and u will have one more friend in p_u). Therefore, by combining Eqs. (14)–(16) we prove Lemma 2. □

Proposition 1 *In a finite potential game, from an arbitrary feasible assignment, the combined dynamics of Fig. 4 always converges to a NEPS solution in a finite number of rounds.*

Proof First, note that we only allow unilateral deviations that do not violate the events' capacity constraints, while bilateral deviations leave the events' cardinalities unchanged. Thus, if we start from a feasible assignment, the assignment will remain feasible after a unilateral or bilateral deviation. Second, in a unilateral deviation the cost of the deviating player decreases; thus, from Eq. (2) it follows that the potential function also drops. In a bilateral deviation, both users benefit (or one benefits and the other's cost is unchanged). But then the sum $\Delta\Phi_{BI}(v, u, \boldsymbol{p})$ which by Lemma 2 equals to:

$$\left(c_v(p'_v, p_u \cup \overline{\boldsymbol{p}_{vu}}) - c_v(p_v, p_u \cup \overline{\boldsymbol{p}_{vu}}) + \frac{1}{2}w(v,u)\right) + \left(c_u(p'_u, p_v \cup \overline{\boldsymbol{p}_{vu}}) - c_u(p_u, p_v \cup \overline{\boldsymbol{p}_{vu}}) + \frac{1}{2}w(v,u)\right)$$

is always negative, which guarantees that the potential function will drop. Since the game is finite, the combined dynamics eventually terminates to an assignment that is $NEPS$. □

Proposition 2 *The PoS in the capacitated game using NEPS is upper bounded by 2.*

Proof Consider any (feasible) assignment S. Let $C(S)$ be the total cost for all users under S, i.e.,

$$C(S) = \alpha \cdot \sum\nolimits_{v \in V} d(v, p_v) + (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \wedge \\ p_v \neq p_f}} w(v,f)$$

It is straightforward to see that:

$$\frac{1}{2}C(S) \leq \Phi(S) \leq C(S) \tag{17}$$

For the socially optimal assignment OPT, we have by definition that $C(OPT) \leq C(S)$ for any feasible assignment S. Now, let $S^{\min}$ be the assignment that minimizes the potential function, i.e., $\Phi(S^{\min}) \leq \Phi(S)$ for any feasible assignment S. An interesting property is that $S^{\min}$ is $NEPS$. Indeed, if that were not the case, then by Lemma 1 there would be a unilateral or bilateral deviation that would further drop the potential function. Inequality (17) together with the above observations implies that: $\frac{1}{2}C(S^{\min}) \leq \Phi(S^{\min}) \leq \Phi(OPT) \leq C(OPT)$, or equivalently, $\frac{C(S^{\min})}{C(OPT)} \leq 2$. The $NEPS$ with the lowest total cost S^{best} will have a total cost of at least $C(S^{\min})$, so we conclude that the PoS is upper bounded by 2. □

Proposition 3 *The PoA in the capacitated game using NE and PS is upper bounded by*

$$\frac{\sum_{k=1}^{|P|} \max_{p\in P} \Xi^p_{(k)}}{\alpha \cdot \sum_{v\in V} \min_{p\in P} c(v,p)}.$$

Proof Assume an assignment that is $NEPS$. Consider any pair of users $u, v \in V$ that are assigned to two distinct events $p_u \neq p_v \in P$. We will now argue that it is not possible to simultaneously hold that $p_u = p^u_1$ and $p_v = p^v_1$, except for the trivial cases where $\xi(u, p^u_1) = \xi(u, p^u_2)$ or $\xi(v, p^v_1) = \xi(v, p^v_2)$. Indeed, if that were the case, users u and v would be incentivized to swap their corresponding events, since they are already in the worst possible case and they cannot lose by deviating. For k users assigned to k distinct events, we can similarly show that if a user u is assigned to his worst event p^u_1, then the rest $k-1$ users must be assigned to at most their second-worst event (except for the trivial cases where multiple events share the same ξ value). Among the remaining $k-1$ users, if one of them, say v, is assigned to his second-worst event p^v_2, then the other $k-2$ must be assigned to at most their third-worst events (except for the trivial cases of equal ξ values among multiple events). We can continue this argument to show that if users $u_1, \ldots, u_{k-1}$ are assigned to their worst, second-worst, up to $(k-1)$th-worst events, then user k can be assigned to at most his kth-worst event.

The above result suggests that if we pick any $|P|$ users assigned to distinct events, then the total cost is upper bounded by the quantity

$$\max_{p\in P, v\in V}\{\xi(v, p^v_1)\} + \cdots + \max_{p\in P, v\in V}\{\xi(v, p^v_{|P|})\}.$$

Note that the values $\max_{p\in P, v\in V}\{\xi(v, p^v_k)\}$ and $\max_{p\in P, v\in V}\{\xi(v, p^v_{k'})\}, k \neq k'$, may occur in the same event or for the same user, even though we assumed (i) distinct events and (ii) that a user can only be assigned to a single event in any feasible assignment. This is, however, not a concern, since we are interested in an upper bound.

In a similar manner, we can obtain an upper bound on the worst possible cost of any $NEPS$ as follows: Assume that one event takes the worst possible value $\Xi^p_{(1)}$ among all $|P|$ events; another event takes the worst possible value

$\Xi^p_{(2)}$ among all $|P|$ events; and, finally, the remaining event takes the worst possible value $\Xi^p_{(|P|)}$ among all $|P|$ events. In other words, we can upper bound the total cost of any $NEPS$ equilibrium by the quantity $\sum_{k=1}^{|P|} \max_{p\in P} \Xi^p_{(k)}$. On the other hand, the optimal solution has a cost of at least $\sum_{v\in V} \min_{p\in P} c(v, p)$, since in the best case scenario each user v is assigned to his event with the least cost and all v's friends with positive friendship weight are assigned to the same event.

Since the numerator is an upper bound on the total cost of a $NEPS$ and the denominator a lower bound on the total cost of the optimal assignment, we can upper bound the PoA by

$$\frac{\sum_{k=1}^{|P|} \max_{p\in P} \Xi^p_{(k)}}{\alpha \cdot \sum_{v\in V} \min_{p\in P} c(v, p)}.$$

□

Lemma 3 *For a unilateral deviation in assignment $\boldsymbol{p}$ where a user v switches from event p_v to p'_v, the change in Objective Function (1) is:*

$$\begin{aligned}
&\Delta C_{UNI}(v, p_v, p'_v, \overline{\boldsymbol{p}_v}) \\
&= \Big(\alpha \cdot d(v, p'_v) + (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \\ \wedge p_f = p_v}} w(v, f)\Big) \\
&- \Big(\alpha \cdot d(v, p_v) + (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \\ \wedge p_f = p'_v}} w(v, f)\Big)
\end{aligned}$$

Proof Before v deviates, Objective Function (1) is equal to:

$$\begin{aligned}
&\alpha \cdot \sum_{\substack{u\in V \\ \wedge u\neq v}} d(u, p_u) + (1-\alpha) \cdot \sum_{\substack{(u,f_u)\in E \\ \wedge u\neq v \\ \wedge p_{f_u}\neq p_u}} w(u, f_u) \\
&+\alpha \cdot d(v, p_v) + (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \\ \wedge p_f \neq p_v}} w(v, f)
\end{aligned} \tag{18}$$

After v deviates to p'_v, Objective Function (1) becomes:

$$\begin{aligned}
&\alpha \cdot \sum_{\substack{u\in V \\ \wedge u\neq v}} d(u, p_u) + (1-\alpha) \cdot \sum_{\substack{(u,f_u)\in E \\ \wedge u\neq v \\ \wedge p_{f_u}\neq p_u}} w(u, f_u) \\
&+\alpha \cdot d(v, p'_v) + (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \\ \wedge p_f \neq p'_v}} w(v, f)
\end{aligned} \tag{19}$$

If we subtract Eq. (18) from Eq. (19), we have:

$$\begin{aligned}
&\Delta C_{UNI}(v, p_v, p'_v, \overline{\boldsymbol{p}_v}) \\
&= \Big(\alpha \cdot d(v, p'_v) + (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \\ \wedge p_f \neq p'_v}} w(v, f)\Big) \\
&- \Big(\alpha \cdot d(v, p_v) + (1-\alpha) \cdot \sum_{\substack{(v,f)\in E \\ \wedge p_f \neq p_v}} w(v, f)\Big)
\end{aligned} \tag{20}$$

which proves Lemma 3. (The right-hand sides of Eq. (20) and Lemma 3 are equal, for the same reasoning as in the proof of Lemma 1.) □

Lemma 4 *For a bilateral deviation in assignment $\boldsymbol{p}$ where two users $v, u \in V$ swap events (v/u from p_v/p_u to p_u/p_v), the change in Objective Function (1) is:*

$$\begin{aligned}
&\Delta C_{BI}(v, u, \boldsymbol{p}) \\
&= \Big(\tilde{c}_v(p'_v, p_u \cup \overline{\boldsymbol{p}_{vu}}) - \tilde{c}_v(p_v, p_u \cup \overline{\boldsymbol{p}_{vu}}) + w(v, u)\Big) \\
&+ \Big(\tilde{c}_u(p'_u, p_v \cup \overline{\boldsymbol{p}_{vu}}) - \tilde{c}_u(p_u, p_v \cup \overline{\boldsymbol{p}_{vu}}) + w(v, u)\Big)
\end{aligned}$$

Proof In a similar manner to the proof of Lemma 2, we express the bilateral deviation as two unilateral deviations. The only difference is the factor 2 in the cost $(1-\alpha)\cdot w(v, u)$, obviously because in LS the agent cost does not include the factor $\frac{1}{2}$ as in $GAME$. □

Proposition 4 describes the complexity of $INIT$; it uses the following lemma.

Lemma 5 *The expected number of trials to draw with replacement m distinct items from a set of $M \geq m$ items is $M \cdot (H_M - H_{M-m})$, where $H_i = \sum_{k=1}^{i} \frac{1}{k}$ is the i-th Harmonic number (with $H_0 = 0$).*

Proof Let the random variable $X_i, 1 \leq i \leq m$, be the number of trials to draw the ith item after the first $i-1$ have been obtained already. The total number of trials is then: $X = X_1 + \ldots + X_m$. By linearity of expectation, the expected number of trials to see all m items will then be: $E[X] = E[X_1] + \ldots + E[X_m]$. Consider the random variable X_i. Since all items are equally likely to be drawn, the probability of drawing an item different from the $i-1$ already seen is: $P_i = 1 - \frac{i-1}{M} = \frac{M-i+1}{M}$. Given the trials are independent (Bernoulli), the expected number of draws to see item i given the first $i-1$ is $\frac{1}{P_i} = \frac{M}{M-i+1}$. But then $E[X] = E[X_1] + \ldots + E[X_m] = \sum_{i=1}^{m} \frac{M}{M-i+1} = M \cdot (H_M - H_{M-m})$. □

Proposition 4 *For $n << |V|$, $INIT$ has a time complexity of $O(\max(|V|n, |E||P|, |V||P|\log(|P|)))$, and a space requirement of $\Theta(|V||P|)$, where n, $|V|$, $|E|$, $|P|$ are the number of samples, nodes, edges and classes, respectively.*

Proof The total running time of $INIT$ consists of three components. The first concerns initialization (Lines 1–6) and heap creation. During initialization, obtaining $c(v, p)$ for a single user v and event p (Lines 3–6) incurs $1+deg_v$ computations, where deg_v is the degree of v. Repeating the process for all $|P|$ events and summarizing over all users yields complexity $\sum_{v\in V}|P|(1+deg_v) = |V||P| + 2|E||P| = O((|V|+|E|)|P|)$. Recall that Phase 2 uses the user-event costs from Phase 1, without requiring initialization. Regarding the complexity of heap creation, since the size of each user heap is $O(|P|)$, heapifying the costs of all users requires $O(|V||P|)$, which is dominated by $O((|V|+|E|)|P|)$.

The second component of the cost is due to the while iterations (Lines 7–24) of both phases. Observe that the total number of iterations equals the number of users, since every iteration performs an non-revocable assignment of a user. An iteration (i) finds the minimum among the top of the heaps of the n samples (Lines 13–17), (ii) updates the cost heaps for each friend of the user v to be assigned (Lines 18–20), and (iii) if the assigned event closes, it is removed from the heaps of all unassigned users (Lines 21–24). Summarizing, over all iterations (i.e., number of users $|V|$), the total cost of (i) is $O(|V|n)$. Item (ii) involves updating the heaps of deg_v friends of v. The cost for all users is $\sum_{v\in V} deg_v \log(|P|) = 2|E|\log(|P|) = O(|E|\log(|P|))$. Regarding (iii), observe that an event closes at least once (when it reaches its minimum capacity during Phase 1), and at most twice (if it also reaches its maximum capacity during Phase 2). When this happens, the event must be removed from the heaps of at most $|V|$ users with cost $O(|V|\log(|P|))$. Repeating for all events yields $O(|V||P|\log(|P|))$. Summarizing items (i) to (iii), the complexity of the second component is $O(|V|n + (|V||P| + |E|)\log(|P|))$.

Recall that each iteration of $INIT$ selects n distinct users from the set V_{un}. If users are drawn equiprobably with replacement, then by Lemma 5 the expected number of draws until n users are selected is $|V_{un}|(H_{|V_{un}|} - H_{|V_{un}|-n})$. $INIT$ performs sampling when $|V_{un}| = \{n+1, n+2, \ldots, |V|\}$; for $|V_{un}| \leq n$ it just considers all users in $|V_{un}|$. So, the total number of draws is $\sum_{k=n+1}^{|V|} k(H_k - H_{k-n}) = \sum_{k=n+1}^{|V|} k(\frac{1}{k-n+1} + \cdots + \frac{1}{k})$. Observe that each term $\frac{1}{k}$ can only occur at most n times with coefficients $k, k+1, \ldots, k+n-1$. The last expression can be rewritten as $\sum_{k=1}^{|V|}\frac{1}{k}(k+k+1+\cdots+k+n-1) \leq \sum_{k=1}^{|V|}\frac{1}{k}n(k+n-1) = n\sum_{k=1}^{|V|}\frac{1}{k}(k+n-1) = n\sum_{k=1}^{|V|}(1+\frac{n-1}{k}) = n|V| + n(n-1)\sum_{k=1}^{|V|}\frac{1}{k}$. Since the sum of the first $|V|$ harmonic numbers is $O(\log(|V|))$, the required number of samples in expectation is $O(|V|n + n^2\log(|V|))$.

Considering that in practice $n << |V|$, the complexity of $INIT$, including all three components, can be simplified to $O(\max(|V|n, |E||P|, |V||P|\log(|P|)))$. The space requirements are $\Theta(|V||P|)$ because of the $|V|$ user heaps, each of size $|P|$, and the cost table $|V||P|$. □

Proposition 5 *The time complexity of a super-round of unilateral and bilateral deviations is* $O(|P|^2|V|(\log(|P|) + \log(|V|)))$.

Proof Regarding the complexity of UNI, each round considers all users. In the worst case, for each user v we must scan the entire heap of size $|P|$ in order to find a valid event to re-assign v. After re-assignment, the costs of deg_v friends change, triggering deg_v heap operations in user heaps (resp. event pair heaps) with complexity $O(deg_v \log(|P|))$(resp. $O(deg_v \cdot \log(|V|))$). Summarizing over all users, the complexity of a single round is $\sum_{v\in V}(|P| + deg_v(\log(|P|) + \log(|V|)) = O(|V||P| + |E|(\log(|P|) + \log(|V|))$. Similarly, for BI, each swap between users v and u updates the costs of their friends (and possibly the costs of v, u, if they are connected) for both events, triggering a maximum of $deg_v + deg_u + 2$ heap operations (in both user and event pair heaps) per swap. Given that deg_v and deg_u are $O(|V|)$ and assuming that all event pairs incur swaps, the complexity of a round of BI is $O(|P|^2|V|(\log(|P|) + \log(|V|)))$. □

The bound in Proposition 5 is loose, because in practice only a small fraction of event pairs causes swaps and $deg_v << |V|$.

References

1. Aarts, E., Lenstra, J.K. (eds.): Local Search in Combinatorial Optimization, 1st edn. Wiley, Hoboken (1997)
2. Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., Cohen, M.: Interactive digital photomontage. ACM Trans. Graph. **23**(3), 294–302 (2004)
3. Andrews, M., Hajiaghayi, M.T., Karloff, H., Moitra, A.: Capacitated metric labeling. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 11, SIAM, pp. 976–995 (2011)
4. Anshelevich, E., Sekar, S.: Approximate equilibrium and incentivizing social coordination. CoRR arXiv:1404.4718 (2014)
5. Armenatzoglou, N., Pham, H., Ntranos, V., Papadias, D., Shahabi, C.: Real-time multi-criteria social graph partitioning: a game theoretic approach. In: SIGMOD (2015)
6. Barnard, S.: Stochastic stereo matching over scale. Int. J. Comput. Vis. **3**(1), 17–32 (1989)
7. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary and region segmentation of objects in N–D images. In: Proceedings of Eighth IEEE International Conference on Computer Vision, vol. 1, pp. 105–112 (2001)
8. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. PAMI **23**(11), 1222–1239 (2001)
9. Calinescu, G., Karloff, H., Rabani, Y.: Improved approximation algorithms for multiway cut. In: Proceedings of the ACM Symposium on Theory of Computing, ACM (1998)

10. Feldman, M., Friedler, O.: A unified framework for strong price of anarchy in clustering games. In: Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming, Part II, Springer International Publishing, Lecture Notes in Computer Science, vol. 9135, pp. 601–613 (2015)
11. Kleinberg, J., Tardos, E.: Approximation algorithms for classification problems with pairwise relationships: metric labeling and Markov random fields. JACM **49**(5), 616639 (2002)
12. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts. IEEE Trans. PAMI **26**, 147–159 (2004)
13. Leskovec, J., Krevl, A.: SNAP datasets: Stanford large network dataset collection (2014). http://snap.stanford.edu/data. Accessed May 2015
14. Levandoski, J.J., Sarwat, M., Eldawy, A., Mokbel, M.F.: Lars: a location-aware recommender system. In: 2012 IEEE 28th International Conference on Data Engineering, pp 450–461. IEEE, Washington, DC, USA (2012). https://doi.org/10.1109/ICDE.2012.54
15. Li, K., Lu, W., Bhagat, S., Lakshmanan, L.V., Yu, C.: On social event organization. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '14, pp 1206–1215. ACM, New York, NY, USA (2014). https://doi.org/10.1145/2623330.2623724
16. Naor, J., Schwartz, R.: Balanced metric labeling. In: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing. STOC '05, pp 582–591. ACM, New York, NY, USA (2005). https://doi.org/10.1145/1060590.1060676
17. Orlin, J.B., Punnen, A.P., Schulz, A.S.: Approximate local search in combinatorial optimization. In: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'04, pp. 587–596 (2004)
18. Rahn, M., Schäfer, G.: Efficient Equilibria in Polymatrix Coordination Games, pp. 529–541. Springer, Berlin (2015)
19. Rother, C., Kolmogorov, V., Blake, A.: "GrabCut"-interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. **23**(3), 309–314 (2004)
20. Schaeffer, S.E.: Survey: graph clustering. Comput Sci Rev **1**(1), 27–64 (2007). https://doi.org/10.1016/j.cosrev.2007.05.001
21. Wainwright, M., Jaakkola, T., Willsky, A.: Map estimation via agreement on trees: message-passing and linear programming. IEEE Trans. Inf. Theory **51**, 3697–3717 (2005)
22. Yedidia, J., Freeman, W., Weiss, Y.: Generalized belief propagation. In: Advances in Neural Information Processing Systems, pp. 689–695 (2000)