Graph-Theoretic Approach for Increasing Participation in Social Sensing

Waseem Abbas
Vanderbilt University
Nashville, Tennessee 37212
waseem.abbas@vanderbilt.edu

Aron Laszka
Vanderbilt University
Nashville, Tennessee 37212
aron.laszka@vanderbilt.edu

Xenofon Koutsoukos
Vanderbilt University
Nashville, Tennessee 37212
xenofon.koutsoukos@vanderbilt.edu

ABSTRACT

Participatory sensing enables individuals, each with limited sensing capability, to share measurements and contribute towards developing a complete knowledge of their environment. The success of a participatory sensing application is often measured in terms of the number of users participating. In most cases, an individual's eagerness to participate depends on the group of users who already participate. For instance, when users share data with their peers in a social network, the engagement of an individual depends on its peers. Such engagement rules have been studied in the context of social networks using the concept of k-core, which assumes that participation is determined solely by network topology. However, in participatory sensing, engagement rules must also consider user heterogeneity, such as differences in sensing capabilities and physical location. To account for heterogeneity, we introduce the concept of (r, s)-core to model the set of participating users. We formulate the problem of maximizing the size of the (r, s)-core using 1) anchor users, who are incentivized to participate regardless of their peers, and by 2) assigning capabilities to users. Since these problems are computationally challenging, we study heuristic algorithms for solving them. Based on real-world social networks as well as random graphs, we provide numerical results showing significant improvement compared to random selection of anchor nodes and label assignments.

CCS CONCEPTS

•**Human-centered computing** →**Collaborative and social computing theory, concepts and paradigms;**

KEYWORDS

Social sensing, participatory sensing, social networks, graph theory, k-core, (r, s)-core

1 INTRODUCTION

Participatory sensing enables users to share measurements of their environment [3, 6, 15, 16], such as traffic and parking situations [8, 9], waiting times at businesses, weather information, or disaster scenarios [10]. Through sharing, users can develop a knowledge of their environment that is significantly more complete than what each individual could develop on its own, relying only on one's limited sensing capabilities. Since the success of a participatory-sensing application is very often measured in terms of the number of its users, incentivizing individuals to join the application is a crucial problem for participatory sensing. For example, previous work has proposed to use a reverse auction mechanism to preserve participation despite various social concerns [11].

A key feature of participatory sensing is that an individual's benefit from the application depends strongly on what measurements are shared by the other users. Consequently, an individual's eagerness to participate depends on the group of users who already participate, which results in a cascading effect: as the size of the user base grows, other individuals become more eager to join. Hence, the number of users can be significantly increased by incentivizing a few individuals to participate, for example, by providing them with rewards or by reaching out to them through targeted advertisement. Due to the cascading effect, incentivizing a small but strategically selected set of individuals can lead to a very large number of users following them and joining the participatory-sensing application.

However, for privacy reasons, many users choose to limit sharing measurements to their peers, such as their friends in a social network. As a result, an individual's benefit is restricted by the structure of the social network, and they will participate only if their peers do [17]. Such engagement rules have been studied in the context of social networks using the concept of k-core, which assumes that an individual will remain part of a social network if it has at least k peers. Based on the concept of k-core, it has been shown that participation can be significantly increased by incentivizing a few individuals, called anchors, to remain part of the network even when they have fewer than k peers [4, 7].

A major limitation of k-core is that it assumes participation to be determined solely by the network structure, ignoring the heterogeneity of users. In participatory sensing, however, the heterogeneity of users and their measurements plays a key role. For example, users may take measurements at different geographical locations (e.g., different streets or even different cities) and they may have devices with different sensing capabilities. In order to develop a more complete knowledge of the environment, these heterogeneous measurements must be combined. As a consequence, a user's benefit from a participatory-sensing application depends
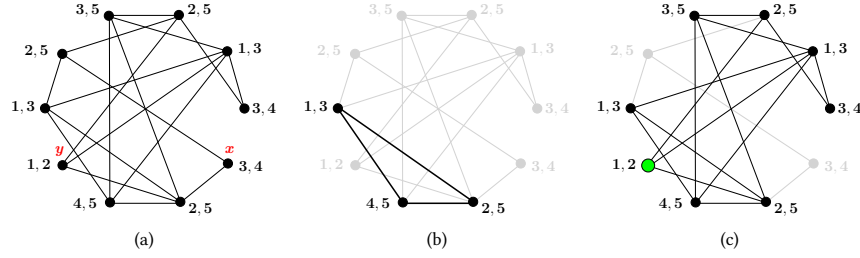
Waseem Abbas, Aron Laszka, and Xenofon Koutsoukos

Figure 1: (a) Example network. (b) $(5, 2)$-core of the example network. (c) $(5, 2)$-core with an anchor node (marked green).

not only on the number of participating peers, but also on the heterogeneous nature of the measurements shared by them.

To account for heterogeneity, we introduce the concept of (r, s)-core, which models the set of users engaging in a participatory-sensing application. We assume that each user has a set of attributes, which capture the user's sensing capabilities, geographical location, etc., and that a user's participation depends on the sets of attributes possessed by its peers [1, 18]. Based on the concept of (r, s)-core, we study the problem of increasing the number of participating individuals (i.e., the size of the (r, s)-core) using two distinct approaches. First, we investigate the problem of selecting a few individuals, called anchor nodes, who are incentivized to participate regardless of their peers. Second, we consider the problem of changing the labels of the nodes (e.g., by incentivizing users to change their attributes).

2 NETWORK MODEL

We model the social network as a simple, undirected graph $G(V, E)$, in which the node set V represents the set of users, and the edge set E represents connections between users. Two nodes $x, y \in V$ are *adjacent* in G if the corresponding users share measurements with each other. Nodes that are adjacent to node x are called *neighbors* of x, and the set of neighbors of node x is denoted by $N(x)$.

Each node of the graph has attributes, which model the properties of the measurements that are shared by the corresponding user, such as geographical location, sensing capabilities, etc. We model these attributes as a *label set*, $R = \{1, 2, \cdots, r\}$, and assign a subset of s labels to each node in G, depending on the attributes of the node. As a result, we have a labeled graph that captures measurement sharing between users with different attributes. We denote labels assigned to node x by $\ell(x)$, i.e.,

$$\ell : \quad V \longrightarrow [R]_s. \tag{1}$$

Here, $[R]_s$ is the set of all s-subsets of R (i.e., all subsets of R that have exactly s elements).

A user *participates* (or engages) in the participatory-sensing application as long as its neighbors provide the user with all the labels that are missing from its own label set (i.e., as long as they provide the user with all the measurements that the user cannot take on its own). Formally, our engagement rule says that a node x participates in the network as long as the following condition is satisfied:

$$\bigcup_{y \in (\{x\} \cup N(x))} \ell(y) = R. \tag{2}$$

This engagement rule models networking phenomena in which a node participates in the network and shares its information as long as it can acquire all missing information from its neighbors. If the condition in (2) is not satisfied for a node, then the node will simply leave the network.

3 PROBLEM DESCRIPTION

A node leaving the network can have a cascading effect as it may cause its neighbors to also leave the network. For instance, consider a node x with a label $a \in \ell(x)$, and let $y \in N(x)$ be such that x is the only node in $\{y\} \cup N(y)$ with the label a. Then, node x leaving the network will also result in node y leaving. Thus, the removal of a node from a network may cause a cascading effect, or *unraveling*, due to which nodes that initially satisfy the condition (2) may also get removed from the network. In the end, we are left with a subnetwork of nodes that all satisfy the engagement rule (2). We call the remaining subnetwork the (r, s)-*core* of the network. More precisely, we define (r, s)-core as follows:

Definition 3.1 ((r, s)-Core). Given a graph $G(V, E)$, a set of r labels, denoted by R, and an assignment $\ell : V \rightarrow [R]_s$ (i.e., assigning s labels from R to each node $v \in V$), the (r, s)-core of G, denoted by $\tilde{G}(\tilde{V}, \tilde{E})$, is the maximal subgraph in which every node satisfies

$$\bigcup_{y \in (\{x\} \cup N(x)) \cap \tilde{V}} \ell(y) = R, \; \forall x \in \tilde{V}. \tag{3}$$

Note that it follows from the definition readily that every graph has a unique (r, s)-core.

Example. As an example, consider the network shown in Figure 1(a). This network has ten nodes, each of which has $s = 2$ labels from the set $R = \{1, 2, 3, 4, 5\}$. Initially, there are two nodes x and y that do not satisfy the condition (2). As a result, they leave the network, which leads to further node removals. In the end, only three nodes remain, as shown in Figure 1(b). Each of these nodes has all five labels between itself and its neighbors; thus, they constitute the $(5, 2)$-core of the network.

The (r, s)-core of a social network represents individuals that will engage in the participatory-sensing application by sharing measurements with their peers. To increase the participation of individuals, we desire to increase the size of the (r, s)-core. Thus, from a design perspective, the following question arises.

For a given network G, label set R, and a positive integer s, how can we modify or design our network so as to maximize the size of its (r, s)-core?

One way to achieve this objective is to find an *optimal assignment* of labels to nodes, i.e., a labeling ℓ defined in (1) that maximizes the size of the (r, s)-core. However, in some situations, the labeling ℓ is fixed, that is, the labels assigned to nodes are given and cannot be changed. For example, in participatory sensing, users can have devices with fixed sensing capabilities that cannot be easily changed. A new approach is needed in these situations to increase participation. In this paper, we explore the idea of significantly increasing participation by incentivizing a few selected individuals, called anchor nodes, to engage regardless of their peers' attributes.

4 INCREASING THE SIZE OF (r, s)-CORE

In this section, we first look at maximizing the (r, s)-core of a network given that the labels of nodes cannot be changed in Section 4.1. Then, we study maximizing (r, s)-core through relabeling nodes in Section 4.2.

4.1 (r, s)-core with Anchors

The departure of a node from the network might lead to the departure of its neighbors, thereby causing a cascading phenomenon. The size of the (r, s)-core of the network can be larger if we can prevent this cascading effect, in which nodes leave successively. One way to achieve this is to incentivize some users in the network to participate even if condition (2) is not satisfied (e.g., by providing them with rewards for participating). As a consequence, the departure of their neighbors can be prevented, and the unraveling of the network can be avoided. We call such nodes *anchors* – nodes that never leave the network irrespective of the labels assigned to them or to their neighbors. Users can be made anchors by incentivizing them, for example, by offering them rewards for their participation when condition (2) is not satisfied. We now define the concept of (r, s)-core with anchors as follows:

Definition 4.1 ((r, s)-Core with Anchors). Given a graph $G(V, E)$, a set of r labels, denoted by R, an assignment $\ell : V \rightarrow [R]_s$ (i.e., assigning s labels from R to each node $v \in V$), and a set of anchor nodes $A \subseteq V$, the (r, s)-core with anchors A is the maximal subgraph $\tilde{G}_A(\tilde{V}_A, \tilde{E}_A)$ consisting of all anchor nodes as well as non-anchor nodes satisfying

$$\bigcup_{y \in (\{x\} \cup N(x)) \cap \tilde{V}_A} \ell(y) = R, \; \forall x \in \tilde{V}_A \setminus A. \tag{4}$$

Note that it follows from the definition readily that for every graph and set of anchor nodes, the (r, s)-core with anchors exists uniquely.

An example of (r, s)-core with an anchor node is shown in Figure 1(c). Without any anchor node, the maximum subgraph in which each node has all five labels between itself and its neighbors consists of only three nodes, that is, the size of the $(5, 2)$-core is three, as shown in Figure 1(b). However, with only one anchor node, we obtain a subgraph consisting of eight nodes in which each non-anchor node has a complete set of five labels between itself and its neighbors. Thus, we see a significant improvement in terms of users participation even with a single anchor.

To obtain the (r, s)-core with anchors, we iteratively remove those *non-anchor* nodes from $G(V, E)$ that do not satisfy condition (2). We repeat this until we are left with a subgraph $\tilde{G}_A(\tilde{V}_A, \tilde{E}_A)$,

which is the (r, s)-core with anchors. We denote this simple algorithm as follows:

$$\tilde{G}_A(\tilde{V}_A, \tilde{A}_A) \leftarrow \mathsf{rs\text{-}anchored\text{-}core}(G, \ell, A, R) \tag{5}$$

Next, we study how to maximize the number of participating individuals using a given number of anchors, denoted by α. Formally, our goal is to find anchors $A \subseteq V$ such that $|A| \leq \alpha$ and the resulting (r, s)-core with anchor nodes A is of maximum size. Unfortunately, this problem is computationally hard.

THEOREM 4.2. *Given a graph $G(V, E)$, a label set R, an integer s, a labeling ℓ, a number of anchor nodes α, and a threshold core size K, determining if there exists a set A of at most α anchor nodes that results in an anchored (r, s)-core whose cardinality is at least K is an NP-hard problem.*

PROOF SKETCH. We show that the above problem is computationally hard using a reduction from a well-known NP-hard problem, the Set Cover Problem.

Definition 4.3 (Set Cover Problem). Given a base set U, a collection C of subsets of U, and a number k, determine if there exists a subcollection $C' \subseteq C$ of at most k subsets such that every element of U is contained by at least one subset in C'.

Given an instance of the Set Cover Problem, we construct an instance of the anchored (r, s)-core problem as follows:
- $R = \{1, 2, 3\}$, $s = 1$, and $\alpha = k$;
- for every $c \in C$, there is a node c with label 1;
- for every $u \in U$, there is a node u with label 2;
- there exist three other nodes o_1, o_2, and o_3 with labels 1, 2, and 3, respectively;
- for every $u \in U$ and $c \in C$, nodes u and c are adjacent if and only if $u \in c$;
- every u is adjacent to o_3;
- o_1, o_2, and o_3 are adjacent to each other;
- $K = \alpha + |U| + 3$.

Clearly, the above reduction can be carried out in time that is polynomial in the size of the Set Cover Problem instance. Hence, it remains to show that the Set Cover Problem has a solution iff the anchored (r, s)-core problem does.

Firstly, if there exists a set cover C', then the anchor set $A = C'$ is a solution to the anchored (r, s)-core problem. To see this, consider that every node $u \in U$ is adjacent to at least one node in A, which provides label 1. Since labels 2 and 3 are provided to every node $u \in U$ by itself and by o_3, the (r, s)-core includes every node $u \in U$. Finally, nodes o_1, o_2, and o_3 are always part of the core since they provide each other with every label, which means that the size of the (r, s)-core is $\alpha + |U| + 3$ (including the α anchor nodes).

Secondly, the other direction (i.e., proving that any solution A to the anchored (r, s)-core problem is also a set cover) follows from a similar argument. $\square$

Since the problem of finding the (r, s)-core maximizing set of anchors is computationally hard, we present two heuristic algorithms: first, one based on a simple *greedy heuristic*, and second, one using a *noisy best response* based strategy.

Greedy Heuristic. Let graph $G(V, E)$, R, s, ℓ, and the number of anchor nodes α be given. We iteratively add nodes to the set of anchors A one-by-one, in each iteration adding a node that maximizes the size of the resulting anchored (r, s)-core, until a set of α anchors is reached. The greedy heuristic is described formally in Algorithm 1.

Algorithm 1 Greedy selection of anchors

1: **Given:** $G(V, E), \ell, R, s, \alpha$
2: **Initialization:** $A = \emptyset$.
3: **While** $|A| \leq \alpha$ **do**
4:     **For All** $v \in (V \setminus A)$ **do**
5:         $\tilde{G}_A(\tilde{V}_A, \tilde{E}_A) \leftarrow$ rs-anchored-core(G, ℓ, A, R).
6:         $f(v) \leftarrow |\tilde{V}_A|$
7:     **End For**
8:     $v' \leftarrow \text{argmax}_{v \in (V \setminus A)} f(v)$
9:     $A \leftarrow A \cup \{v'\}$
10: **End While**

Noisy Best Response. Let graph $G(V, E)$, R, s, ℓ, and the number of anchor nodes α be given. We start with a random set A of α anchors, and iteratively try to improve it. In each iteration, we remove a randomly chosen anchor node from A and add a randomly chosen non-anchor node, and we make this switch permanent with a certain probability that depends on the sizes of the anchored (r, s)-cores before and after the switch. The noisy best response heuristic is described formally in Algorithm 2.

Algorithm 2 Selecting anchors to increase the size of (r, s)-core

---

1: **Given:** $G(V, E), \ell, R, s, \alpha$
2: **Initialization:** Pick a small $\epsilon \in \mathbb{R}_+$, number of iterations, and a subset of α nodes A.
3: $\tilde{G}_A(\tilde{V}_A, \tilde{E}_A) \leftarrow$ rs-anchored-core(G, ℓ, A, R)
4: $\kappa \leftarrow |\tilde{V}_A|$
5: **While** $i \leq$ number of iterations **do**
6:     Randomly select $x \in A$ and $y \in V \setminus A$.
7:     $A' \leftarrow (A \setminus \{x\}) \cup \{y\}$.
8:     $\tilde{G}_{A'}(\tilde{V}_{A'}, \tilde{E}_{A'}) \leftarrow$ rs-anchored-core(G, ℓ, A', R)
9:     $\kappa' \leftarrow |\tilde{V}_{A'}|$
10:     $P_\epsilon \leftarrow \frac{\epsilon^\kappa}{\epsilon^\kappa + \epsilon^{\kappa'}}$.
11:     **If** rand$(0, 1) \leq P_\epsilon$
12:         $A \leftarrow A', \kappa \leftarrow \kappa'$
13:     **End If**
14:     $i \leftarrow i + 1$
15: **End While**

4.2 Increasing (r, s)-Core by Node Relabeling

The size of the (r, s)-core depends on both the structure of the network and the assignment ℓ of labels to nodes. For a fixed network structure, the size of the (r, s)-core can be significantly increased by finding a suitable ℓ. In practice, re-assigning labels can be achieved by incentivizing users to change their sensing attributes, for example, incentivizing them to share measurements at different locations or times. In this subsection, we present and discuss the performance

of a heuristic algorithm that re-assigns labels to nodes with the goal of maximizing the size of the (r, s)-core.

First, we present our heuristic for re-assigning labels to nodes to increase the size of the (r, s)-core. For this, we first need to quantify the usefulness of the labels that are assigned to a node x. For instance, if x is the only node providing the labels $\ell(x)$ to its neighbors $N(x)$, then the departure of x will surely lead to the departure of its neighbors. Thus, the labels assigned to x are of great value to the neighbors of x. On the other hand, if for each $y \in N(x)$, the labels $\ell(x)$ are also available from nodes in $N(y) \setminus x$, then the departure of x has no effect on whether $y \in N(x)$ remains in the network or leaves it.

In light of this, for each $y \in N(x)$, we quantify the value (or usefulness) of the labels assigned to x by counting the number of labels in $\ell(x)$ that are not assigned to any node in $N(y) \setminus \{x\}$. The overall value of the labels assigned to x, called the *score*, is simply the sum of the usefulness values for all nodes in $N(x)$. More formally, the score of $\ell(x)$ can be computed using Algorithm 3 below. Here, $\ell(V \setminus x)$ simply denotes the assignment of labels to all nodes other than x.

Algorithm 3 Finding the score Score$(x, \ell(x), \ell(V \setminus x))$ of labels assigned to x

---

1: **Given:** $x, \ell(x), \ell(V \setminus x)$
2: $\sigma = 0$
3: **For All** $y \in N(x)$ **do**
4:     $\beta_y \leftarrow \left[\bigcup_{z \in (N(y) \setminus \{x\})} \ell(z) \right] \cup \ell(y)$.
5:     $\sigma \leftarrow \sigma + |\ell(x) \setminus \beta_y|$
6: **End For**
7: Score$(x, \ell(x), \ell(V \setminus x)) \leftarrow \sigma$

For an illustration, consider the example in Figure 2. Here, node x has three neighbors, y_1, y_2, and y_3. Note that for y_1, x is the only node in $N(y_1)$ with label 2. Similarly, for both y_2 and y_3, x is the only node with labels $\{1, 2\}$ in $N(y_2)$ and $N(y_3)$, respectively. As a result, the overall score of $\ell(x)$ is $1 + 2 + 2 = 5$.
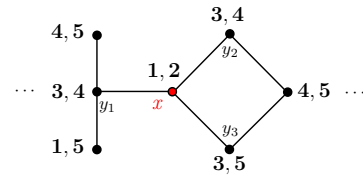

Figure 2: The score of labels assigned to node x is 5.

Next, we present our heuristic algorithm, which employs a noisy best-response strategy. The idea is to choose a node at random, and then select a subset of s labels randomly from the label set R. The labels of the node are replaced with the randomly selected set with high probability if the replacement results in a better score. The occasional selection of suboptimal labels for a node depends on a parameter ϵ, commonly referred to as the *temperature*, and prevents the search from getting stuck in a local optimum. We note that this heuristic belongs to a class of learning algorithms, known
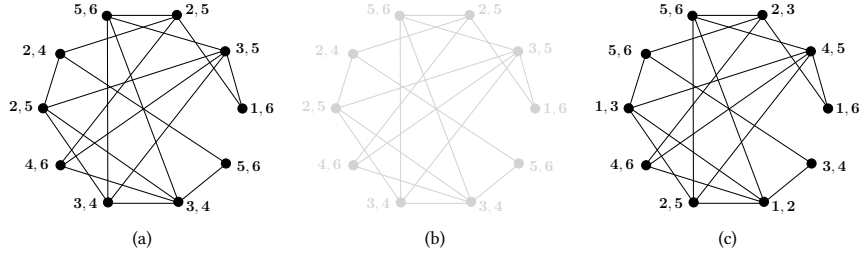
Figure 3: (a) Example network. (b) $(6, 2)$-core with initial label assignment. (c) $(6, 2)$-core with re-assigned labels.

as *log-linear learning* [5, 14], which are known to converge to an optimal solution (asymptotically) if the problem can be formulated as a *potential game*. In [2], we showed that the problem of assigning labels to nodes such that the number of labels missing from the neighborhood of nodes is minimized can be formulated as a potential game. In Algorithm 4, we formally describe our heuristic for relabeling nodes to increase the size of the (r, s)-core.

Algorithm 4 Relabeling to increase the size of (r, s)-core

1: **Given:** $G(V, E), \ell, R, s$
2: **Initialization:** Pick a small $\epsilon \in \mathbb{R}_+$ and number of iterations.
3: **While** $i \leq$ number of iterations **do**
4:     Randomly select a node $x \in V$.
5:     Randomly select s labels for x, i.e, $\ell'(x) \in [R]_s$.
6:     $\sigma \leftarrow \text{Score}(x, \ell(x), \ell(V \setminus x))$
7:     $\sigma' \leftarrow \text{Score}(x, \ell'(x), \ell(V \setminus x))$
8:     $P_\epsilon \leftarrow \frac{\epsilon^\sigma}{\epsilon^{\sigma'} + \epsilon^\sigma}$.
9:     With probability P_ϵ, set $\ell(x) \leftarrow \ell'(x)$.
10:     $i \leftarrow i + 1$
11: **End While**
12: $\tilde{G}(\tilde{V}, \tilde{E}) \leftarrow \text{rs-core}(G, \ell, R)$

An example is shown in Figure 3. With the initial assignment of two labels to each node from the label set $R = \{1, 2, \cdots, 6\}$ (see Figure 3(a)), the $(6, 2)$-core of the network has no node (see Figure 3(b)). The nodes are re-labeled using Algorithm 4 by selecting $\epsilon = 0.01$ and performing 500 iterations. As a result, we achieve a labeling with which the $(6, 2)$-core consists of the whole network (see Figure 3(c)).

5 NUMERICAL RESULTS

We evaluate our results on two different types of networks, including a real-world social network of 4,039 Facebook users [12, 13] and randomly-generated preferential attachment networks. Each preferential attachment network consists of 500 nodes, and it is obtained by adding nodes to the network one at a time. Each new node is connected to three existing nodes, which are chosen with probabilities proportional to the degrees (i.e., number of neighbors) of the nodes. In all of our simulations, labels from the label set are initially assigned uniformly at random to the nodes.

In the case of our sample Facebook network, we first illustrate the size of the (r, s)-core as a function of s, which is the number of labels assigned to each node (Figure 4). As expected, the size of the (r, s)-core increases as the ratio s/r increases. In Figure 5,

we illustrate the size of anchored (r, s)-core as a function of the number of anchors selected using Algorithms 1 and 2. We consider the label set $R = \{1, 2, \cdots, 15\}$, assigning a single label to each node randomly. In Algorithm 2, we perform 5,000 iterations.
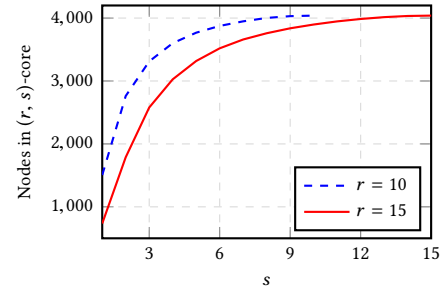

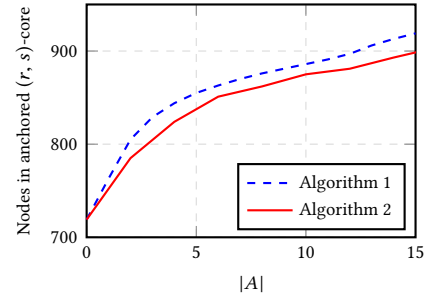Figure 4: Size of (r, s)-core as a function of s for $r = 10, 15$.

Figure 5: Size of anchored $(15, 1)$-core as a function of the number of anchors selected using Algorithms 1 and 2.

Similarly, for the preferential attachment networks, we plot the size of anchored (r, s)-core as a function of the number of anchors chosen by Algorithms 1 and 2 in Figure 6. Here, we consider r and s to be 6 and 2 respectively. We also note that each point in Figure 6 is an average of 25 randomly generated instances. For Algorithm 2, we perform 2,500 iterations while selecting ϵ to be 0.08. The figure shows that the greedy algorithm performs slightly better than Algorithm 2. We also note that the greedy algorithm makes $O(|A| \cdot |V|)$ calls to the rs-anchored-core routine, whereas Algorithm 2 makes $O(iter)$ such calls. Here, *iter* is the number of iterations performed in the Algorithm 2. The results obtained from

Algorithm 2 can be refined further by performing more iterations. A sample plot illustrating the size of (r, s)-core as a function of the number of iterations in Algorithm 2 is shown in Figure 7.

From both Figures 5 and 6, we see that by adding few anchors, we can significantly increase user participation within the network.
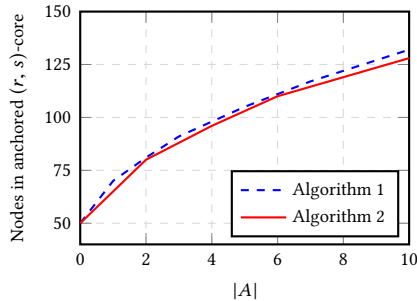

Figure 6: Size of anchored $(6, 2)$-core as a function of the number of anchors selected using Algorithms 1 and 2.
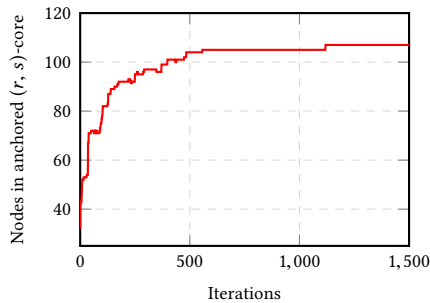

Figure 7: Size of anchored $(6, 2)$-core as a function of the number of iterations in Algorithm 2 for the preferential attachment network with 500 nodes.

6 CONCLUSIONS AND FUTURE DIRECTIONS

In participatory sensing, an individual's intent to participate by sharing information also depends on what information and measurements are shared by others. In a social network, a user's participation depends not only on the number of peers but also on the types of peers participating, which may be different in terms of their sensing capabilities, resources, information etc. To model such engagement rules among heterogeneous nodes within a network, we introduced the notion of (r, s)-core. To maximize the number of users participating within a network, we proposed two approaches. In the first approach, we proposed that few individuals, called anchors, be incentivized to participate regardless of their peers. In the second approach, we proposed to re-assign capabilities to users. To select anchors, we proposed a greedy and a noisy response based heuristics. Similarly, we presented a heuristic to re-assign labels to individuals to maximize the number of participating users within a network. Finally, we demonstrated the efficacy of our approaches through simulation results.

In future work, we aim to develop a generalized solution by combining two approaches – re-assigning labels and selecting anchors – to maximize the number of users participating within the network. We would also like to consider an adversarial setting in which a competitor tries to minimize the size of the (r, s)-core by manipulating a small group of users.

REFERENCES

[1] Waseem Abbas and Magnus Egerstedt. 2014. Characterizing heterogeneity in cooperative networks from a resource distribution view-point. *Communications in Information and Systems* 14, 1 (2014), 1–22.
[2] Waseem Abbas, Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. 2016. Scheduling resource-bounded monitoring devices for event detection and isolation in networks. *arXiv preprint arXiv:1608.07329* (2016).
[3] Charu C Aggarwal and Tarek Abdelzaher. 2013. Social sensing. In *Managing and mining sensor data.* Springer, 237–297.
[4] Kshipra Bhawalkar, Jon Kleinberg, Kevin Lewi, Tim Roughgarden, and Aneesh Sharma. 2015. Preventing unraveling in social networks: the anchored k-core problem. *SIAM Journal on Discrete Mathematics* 29, 3 (2015), 1452–1475.
[5] Lawrence E Blume. 1993. The statistical mechanics of strategic interaction. *Games and economic behavior* 5, 3 (1993), 387–424.
[6] Jeffrey A Burke, Deborah Estrin, Mark Hansen, Andrew Parker, Nithya Ramanathan, Sasank Reddy, and Mani B Srivastava. 2006. Participatory sensing. In *Proceedings of the 1st Workshop on World-Sensor-Web (WSW).* ACM.
[7] Rajesh Chitnis, Fedor V Fomin, and Petr A Golovach. 2013. Preventing unraveling in social networks gets harder. *arXiv preprint arXiv:1304.6420* (2013).
[8] Raghu K Ganti, Nam Pham, Hossein Ahmadi, Saurabh Nangia, and Tarek F Abdelzaher. 2010. GreenGPS: A participatory sensing fuel-efficient maps application. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys).* ACM, 151–164.
[9] Raghu K Ganti, Nam Pham, Yu-En Tsai, and Tarek F Abdelzaher. 2008. PoolView: stream privacy for grassroots participatory sensing. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys).* ACM, 281–294.
[10] Siyu Gu, Chenji Pan, Hengchang Liu, Shen Li, Shaohan Hu, Lu Su, Shiguang Wang, Dong Wang, Tanvir Amin, Ramesh Govindan, and others. 2014. Data extrapolation in social sensing for disaster response. In *Proceedings of the 2014 IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS).* IEEE, 119–126.
[11] Buster O Holzbauer, Boleslaw K Szymanski, and Eyuphan Bulut. 2012. Socially-aware market mechanism for participatory sensing. In *Proceedings of the 1st ACM International Workshop on Mission-Oriented Wireless Sensor Networking (MiSeNet).* ACM, 9–14.
[12] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data. (June 2014).
[13] Jure Leskovec and Julian J Mcauley. 2012. Learning to discover social circles in ego networks. In *Advances in Neural Information Processing Systems (NIPS).* 539–547.
[14] Jason R Marden and Jeff S Shamma. 2010. Revisiting log-linear learning: Asynchrony, completeness and payoff-based implementation. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on.* IEEE, 1171–1172.
[15] Moo-Ryong Ra, Bin Liu, Tom F La Porta, and Ramesh Govindan. 2012. Medusa: A programming framework for crowd-sensing applications. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys).* ACM, 337–350.
[16] Dong Wang, Md Tanvir Amin, Shen Li, Tarek Abdelzaher, Lance Kaplan, Siyu Gu, Chenji Pan, Hengchang Liu, Charu C Aggarwal, Raghu Ganti, and others. 2014. Using humans as sensors: An estimation-theoretic perspective. In *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks (IPSN).* IEEE, 35–46.
[17] Shaomei Wu, Atish Das Sarma, Alex Fabrikant, Silvio Lattanzi, and Andrew Tomkins. 2013. Arrival and departure dynamics in social networks. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining (WSDM).* 233–242.
[18] Jaewon Yang, Julian McAuley, and Jure Leskovec. 2013. Community detection in networks with node attributes. In *13th IEEE international conference on Data Mining (ICDM).* 1151–1156.