ABSTRACT
Understanding a social network is a fundamental problem in social network analysis because of its numerous applications. Recently, user engagement in networks has received extensive attention from many research groups. However, most user engagement models focus on global user engagement to maximize (or minimize) the number of engaged users. In this study, we formulate the so-called Local User Engagement Maximization (LUEM) problem. We prove that the LUEM problem is NP-hard. To obtain high-quality results, we propose an approximation algorithm that incorporates a traditional hill-climbing method. To improve efficiency, we propose an efficient pruning strategy while maintaining effectiveness. In addition, by observing the relationship between the degree and user engagement, we propose an efficient heuristic algorithm that preserves effectiveness. Finally, we conducted extensive experiments on ten real-world networks to demonstrate the superiority of the proposed algorithms. We observed that the proposed algorithm achieved up to 60% more engaged users compared to the best baseline algorithms.

ACM Reference Format:
Junghoon Kim, Jungeun Kim, Hyun Ji Jeong, and Sungsu Lim. 2018. LUEM: Local User Engagement Maximization in Networks. In Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX'). ACM, New York, NY, USA, 15 pages. https://doi.org/XXXXXXXXXXXXXX

1 INTRODUCTION
With the proliferation of mobile devices and the development of IT industry, many people use social networking services via mobile devices every day, anywhere, and anytime. One of the biggest social networking services, Facebook, has reached over 2.8 billion monthly active users in 2021. Moreover, approximately seven-in-ten U.S. adults use Facebook.

With the increasing popularity of social networking services, understanding social networks is an important and fundamental problem [41, 70]. There have been several efforts to capture the characteristics of social networks, such as the importance of user nodes [12, 16, 67] or social relationships [17, 26, 58], stability of a network [87], degree distribution [3], small average distance [1], and community size distribution [74].

In addition, encouraging user engagement [9, 34, 51, 52, 56, 84, 86] in social networks has recently received extensive attention from many research groups. This study focused on engaged (activated) users in a social network. In practice, many services have many registered users, but the amount of people who actively use the service is another matter. We can observe that many of the previously used services were shut down. Even if there are many registered users, if people do not actively use it, the service becomes meaningless. The most widely accepted model [8, 18, 45, 51, 85] for measuring the user engagement in a social network is based on the minimum degree [72], which indicates that every engaged user has at least k friends on a social network. Formally, given a graph \( G = (V, E) \) and positive integer \( k \), if a user has at least \( k \) friends, the user is considered to be engaged. Otherwise, the user is disengaged from the network. If any user is disengaged, a set of users can iteratively be disengaged in a cascade manner because the number of friends of the remaining users changes [9, 83, 84]. Note that the number of engaged users in a social network can be identified by computing \( k \)-core [72]. There are two main research directions regarding user engagement in social networks [8, 22, 54, 54, 84]. (1) The anchored \( k \)-core problem was proposed by Bhawalkar et al. [8]. It aims to maximize the number of engaged users by anchoring \( b \) disengaged users. In other words, the problem is to find \( b \) important disengaged users. (2) The collapsed \( k \)-core problem was proposed by Zhang et al. [84]. It aims to minimize the number of engaged users by removing \( b \) users, that is, the problem is to find \( b \) important engaged users.

Motivation. The above user engagement problems focus on graph-level user engagement, that is, they aim to maximize (or minimize) the number of globally engaged users in a network. This can be considered as a macro-level user engagement. To get all the engaged users, most approaches utilize existing cohesive subgraph models, such as \( k \)-core [72] or \( k \)-truss [24]. In the resultant cohesive subgraph, existing approaches try to find a set of anchored nodes [22] in the non-engaged users or collapsers [84] in the engaged users to maximize (or minimize) the number of engaged users globally, i.e., the total number of engaged or disengaged users is the major matter in the macro-level user engagement. Note that these approaches assume that a given network is a snapshot of a social
network, i.e., each vertex is a real person and each edge indicates a friend relation in a social networking service.

However, let assume that we want to perform targeted marketing for promotion, which is known as an effective marketing tool. Targeted marketing involves breaking the target users into segments and then selecting influential users who widely spread our promotion. In targeted marketing, selecting influential users for each segment is very important. Since the target user is found by each segment, the user should have a local impact. In these days, according to the proliferation of social networks, many companies use social networks to find influential users. Hence, we propose the problem that finds a kind of micro-level (or called local) user engagement from social networks.

This problem is distinguished from the anchored k-core problem[22] and collapsed k-core problem[84]. The following example describes that our problem can be utilized to promote a social networking service.

**Example 1.** By utilizing the LUEM problem, we can find a set of people who effectively promote the products to their friends. The selected people may have many friends and there are sufficient social relationships among friends, i.e., they are important from the perspective of user engagement. This approach is distinguished from the degree-based approach. If we consider only the degree of the nodes, a set of fake accounts which has many fake friends will be selected. These fake accounts have no effect on increasing the number of engaged users since they may have only few relationships to each other.

As an alternative to the difficult-to-expect effects in global user engagement, we focused on local user engagement, which unifies three key concepts: (1) The existence of seed nodes, (2) minimum-degree-based user engagement; and (3) seed-based distance cohesiveness. These key concepts are described as follows:

- **Existence of seed nodes**: We focused on the local structure of seed nodes to measure user engagement. This changes the perspective from a global network structure to a local network structure. Intuitively, when a node is selected as the seed node, a set of nodes closely related to the seed node can be engaged. This can be considered as a micro-level of user engagement. This has led to new applications. Figure 1 depicts the difference between the global user engagement problem (macro-level, green-coloured) and proposed local user engagement problem (micro-level, blue-coloured). Notice that local user engagement has seed nodes, and the engaged users are affected by the seed nodes and can overlap. There have been several similar attempts at social network analysis. The influence maximization (IM) problem[42] aims to identify a set of seed nodes to maximize the spread of influence in a social network under information diffusion models.

- **Minimum-degree-based structural cohesiveness**: We incorporated the widely used minimum-degree user engagement model[8, 81, 83, 84] to measure user engagement. Formally, a user is engaged if he/she has at least k friends in the local structure. Otherwise, the user is disengaged.

- **Distance-generalized cohesiveness**: In the social sciences field, studying the 1-hop neighbor structure to capture structural characteristics is well established[53]. For the user engagement-related problem, the 1-hop neighbor structure can be considered because many social networks are scale-free[3]; thus, the degree of the nodes in social networks follows a power-law distribution. Therefore, we generalize the relationship between two nodes by incorporating the graph distance. i.e., if a node is reachable from the seed node within a specific user-defined distance threshold, the node can be engaged.

In this paper, we aim to find an answer to the question *How many close neighbor nodes will be engaged when a user is activated (becomes a seed node)*? This question is answered by proposing a new information diffusion model based on user engagement in the IM problem. Compared to the traditional information diffusion models, our model is characterized by a spreading limitation (distance) and influenced limitation (minimum degree) without using any simulation models. Hence, the results are deterministic. Figure 2 illustrates a summary of our problem.

We formulate a local user engagement maximization (LUEM) problem. First, we propose a seed-based engaged group(SEG) for a specific node v. A set of nodes H is called the SEG of the node v if the nodes in H are engaged by the seed node v. That is, the minimum degree of the induced subgraph of nodes H is larger than or equal to threshold k, and all nodes in H are reachable from v at distance r. The LUEM problem is defined as follows. Given a graph G, budget b, engagement threshold k, and distance threshold r, LUEM aims to find b seed nodes that can maximize the number of engaged users who have at least k friends in each SEG and are reachable within a distance r from the seed nodes. We show that this problem is NP-hard, and the objective function is monotone submodular. Therefore, we present an approximation algorithm that holds 1 − 1/ε-approximation ratio. We observe that the main bottleneck of the proposed algorithm is to compute the SEGs of the nodes because for each SEG, it is necessary to compute the k-core and connected component containing a seed node O(|V | + |E|)). Thus, to improve efficiency, we propose a new heuristic algorithm that incorporates the approximately neighborhood functions[10] without computing all SEGs at the initial stage.
**Applications.** The applications of our proposed problem are listed as follows:

- **Organizing a party.** Let us assume that Joy wants to host a farewell party for three consecutive holiday days by serving dinner. She would like to invite three of her friend groups. Because her friends are from different groups, such as a table tennis club, graduate association club, lab colleagues, etc., she would like to invite as many non-engaged people in the party as possible. Here, finding a few friends to help her organize party.

- **Finding top influential communities.** Finding the engaged users of a specific node can be considered as finding an influential community of the specific node.

- **Finding key users in a social network.** Important users are nodes that can affect the network structure to a greater extent compared to other nodes [78]. Our LUEM can be used to identify important users in a social network using the user engagement model.

- **Virus propagation prediction/estimating influence spreading.** LUEM can identify a set of engaged users of a specific node, and thus, it can be utilized for predicting the virus propagation and estimating the influence spreading. If a user is infected, a set of close neighbor nodes has a high probability of getting infected.

**Challenges and Contributions** Because the LUEM problem is NP-hard, computing an exact solution within polynomial time is prohibited. Thus, the first challenge is to compute an effective solution. The second challenge is the efficient computation of the solution. To address these challenges, we propose two algorithms: (1) An effective $r$-neighbor-based approximation algorithm (ERA) and (2) Fast circle algorithm (FCA). First, the ERA algorithm incorporates a traditional greedy algorithm to maximize the number of engaged users. Owing to the submodularity of our objective function, we can design an efficient strategy by not computing all engaged users for every iteration. ERA holds a $1 - 1/e$ approximation ratio. In section 3, we present our main idea and rigorous proof to demonstrate why the pruning strategy preserves the approximation ratio. Next, we propose a heuristic algorithm called FCA to improve the efficiency of the proposed algorithms. Even if our proposed ERA significantly improves efficiency, in the worst case, it has the same time complexity as a traditional greedy algorithm. Thus, in FCA, we propose a very fast heuristic algorithm that incorporates the approximated neighborhood function based on our observations. The contributions of this research are summarized as follows:

- **Problem definition:** To the best of our knowledge, this is the first study to identify a set of seed nodes to maximize the number of engaged users in a social network.

- **Theoretical analysis:** We prove that the objective function of LUEM is monotone submodular, and the LUEM problem is NP-hard.

- **Designing new algorithms:** Because our problem is NP-hard, we propose a $1 - 1/e$ approximation algorithm, as well as a heuristic algorithm to improve efficiency.

- **Extensive experimental study:** Using real-world datasets, we conduct extensive experiments to demonstrate the superiority of the proposed algorithms.

**2 PROBLEM STATEMENTS**

We present a Local User Engagement Maximization (LUEM) problem. In this study, we consider an unweighted and undirected graph. Given a graph $G = (V, E)$ and set of nodes $H \subseteq V$, we denote $G[H] = (H, E[H])$ as a subgraph of $G$ induced by nodes $H$. Table 1 lists the basic notations used in this study. First, We introduce basic definitions of some terminology for presenting our problem.

**Definition 1.** (User engagement constraint $\delta(.)$) Given a graph $G = (V, E)$ and positive integer $k$ called the minimum degree threshold, a subgraph $H \subseteq V$ satisfies the user engagement constraint if it is connected, and the minimum degree of induced subgraph $G[H]$ is larger than or equal to $k$, i.e., $\delta(H) \geq k$.

Given a graph $G = (V, E)$, finding a maximal subgraph satisfying the minimum degree constraint is the same as that in the classic $k$-core problem. $k$-core can be computed in polynomial time.

**Definition 2.** (Distance constraint $\tau(.)$) Given a graph $G = (V, E)$, seed node $s \in V$, and positive integer $r$ called a distance threshold, a subgraph $H \subseteq G$ satisfies the distance constraint if it contains $s$, and the distance from $s$ to any node in the induced subgraph $G[H]$ is less than or equal to $r$, that is, $\tau(s, H) \leq r$.

We next define $r$-neighbors based on the distance threshold $r$.

**Definition 3.** ($r$-neighbors $N_r$) Given a graph $G = (V, E)$, node $v \in V$, and positive integer $r$, $r$-neighbors of $v$, denoted as $N_r(v, G)$, is a set of nodes that are reachable from $v$ within a distance $r$ in graph $G$. We use $N(v, G) = N_1(v, G)$, and $N(v)$ if it is obvious.

We are now ready to discuss user-level engagement by defining a set of users engaged by the seed node as follows.

**Definition 4.** (Seed-based engaged group(SEG)) Given a graph $G = (V, E)$, seed node $s$, and positive integers $k$ and $r$, seed-based engaged group of the node $s$, denoted as $\mathcal{K}_{k,r}(s)$, is a maximal set of nodes satisfying $\delta(\mathcal{K}_{k,r}(s)) \geq k$ and $\tau(s, \mathcal{K}_{k,r}(s)) \leq r$. If it is obvious, we use $\mathcal{K}(s)$ instead of $\mathcal{K}_{k,r}(s)$.

Note that a set of users in $\mathcal{K}(s)$ is considered as engaged by the seed node $s$. As we have discussed in section 1, we aim at maximizing the number of engaged users by selecting seed nodes. We next define the engagement gain of SEG.

| Table 1: Notation |
|-------------------|
| Description | Notation |
| minimum degree threshold | $k$ |
| distance threshold | $r$ |
| user engagement constraint | $\delta(.)$ |
| distance constraint | $\tau(.)$ |
| approximated neighbor value | $ANV$ |
| $r$-neighbors of node $v$ in $G$ | $N_r(v, G)$ |
| effective $r$-neighbors size of node $v$ | $\mathcal{E}_r(v)$ |
| SEG of node $v$ | $\mathcal{K}_{k,r}(v)$ |
| distinct engaged users in $\mathcal{K}$ | $\rho(S)$ |
| engagement gain | $\rho(v, S)$ |
Assume that $S \subseteq T \subseteq V$ and there is a node $u \in V \setminus T$ which makes $\rho(S \cup \{u\}) - \rho(S) < \rho(T \cup \{u\}) - \rho(T)$. It implies that $\rho(u, T)$ is larger than $\rho(u, S)$. However, we know that (1) $\rho(T) \geq \rho(S)$; and (2) a user engaged by the set of seed nodes $S$ is always the engaged user by the set of seed nodes $T$. Hence, the number of engageable users by the node $u$ in $T$ is smaller than or equal to the number of engageable users by the node $u$ in $S$. Hence, due to $S \subseteq T$, $\rho(S \cup \{u\}) - \rho(S) < \rho(T \cup \{u\}) - \rho(T)$ does not hold. It implies that our assumption is not true, and our function $\rho(.)$ is submodular. □

Property 3. $\rho(.)$ function is monotone.

Proof. The proof is trivial. For any $S \subseteq T \subseteq V$, $\rho(S) \leq \rho(T)$ always holds. Therefore, $\rho(.)$ is monotone. □

Theorem 1. LUEM problem is NP-hard.

Proof. Proof can be checked in B □

Comparing with [73]. Note that computing SEG is the same with finding a solution to a community search problem [73]. The problem is that: given a graph $G$ and a set of query nodes $Q$, their model aims to find a connected subgraph while maximizing the minimum degree such that (1) contains all the query nodes $Q$ and 2) all nodes in the subgraph are at a distance to $Q$ less than a threshold. Hence, by restricting $|Q| = 1$ and applying a degree constraint, [73] can be utilized to find the SEG. However, note that finding SEGs is different from finding a solution for LUEM. When we use a greedy approach to find a solution for LUEM by utilizing [73], the approach is the same as our basic algorithm 3.3.

3 APPROXIMATION ALGORITHMS

In this section, we introduce approximation algorithms to solve the LUEM problem. First, we present a framework of the approximation algorithm using the characteristics of our objective function and present a basic algorithm (BA). To improve efficiency, we present an effective $r$-neighbor-based algorithm (ERA) with a pruning strategy.

3.1 Computing SEG

Algorithm 1: Computing SEG

```
Algorithm 1: Computing SEG

input : $G = (V, E)$, user parameters $b$, $k$, and a seed node $s$
output : $SEG \mathcal{K}_{b,k}(s)$
1 $\mathcal{K}_{b,k}(s) \leftarrow \emptyset;$
2 $D \leftarrow k$-core($G[N_r(s), k]$);
3 if $D$ contains $s$ then
4     return connectedComp($D, s$);
5 end
6 return null;
```

In this section, we present how to compute an SEG given a seed node $s$ using Algorithm 1. Computing an SEG is simple and intuitive. First, we obtain an induced network of $r$-neighbors from the seed node $s$, then compute the $k$-core (line 2). Because the $k$-core returns multiple connected components, we select a connected component that contains the seed node $s$ (line 4). If there is no connected
component that contains seed node \( s \) in the \( k \)-core, the algorithm returns null (line 5). Hence, computing an SEG requires \( O(|V|+|E|) \), and computing all SEGs in a graph requires \( O(|V|(|V|+|E|)) \).

### 3.2 Framework

**Algorithm 2: Algorithmic framework of hill-climbing approach**

| Input | Output |
|-------|--------|
| \( G = (V,E), k, r, b \) | A set of seed nodes \( S \) |

1. \( S \leftarrow \emptyset \);
2. for \( |S| \neq b \) {
   3. \( s \leftarrow \arg\max_{v \in V} \rho(v,S) \);
   4. \( S \leftarrow \{s\} \cup S \);
   5. }
3. return \( S \);

Algorithm 2 depicts the framework of the \( 1 - \frac{1}{e} \) approximation algorithm. This framework incorporates a widely used greedy optimization search named hill-climbing approach [62] to find a solution. It iteratively identifies a seed node that maximizes the number of engaged users when it merges with the current solution \( S \). Note that after adding a seed node, the number of engageable users of possible seed nodes can be decreased or unchanged. Therefore, it is required to check which seed node has the largest number of disengaged users at every iteration. After finding \( b \) seed nodes, the procedure is terminated.

### 3.3 Basic algorithm (BA)

The direct implementation of the Algorithm 2 is the basic algorithm (BA) with a \( 1 - \frac{1}{e} \) approximation ratio. The procedure of the BA is as follows.

**Procedure 1.** At the initial stage, it computes all the SEGs (Lines 1-3). Then, until finding \( b \) seed nodes, it iteratively finds a node that can maximize the number of engaged users. The selected node will be added to the solution (Lines 4-6). Finally, it returns the selected seed nodes as a result (Line 7).

**Example 3.** In Figure 4, suppose that \( k = 2, r = 1 \), and \( b = 2 \). To find a solution, we first select a node ‘f’ because the \( \rho(f,S) \) is 4. Therefore, the nodes \{d, e, f, g\} are engaged. Next, we select a node ‘a’, which enables engaging nodes \{a, b, c\}. When we find two seed nodes \{f, a\}, we terminate the algorithm because \( b = 2 \).

In the following, we briefly demonstrate the \( 1 - \frac{1}{e} \) approximation ratio of the framework. Please refer to [42, 62] for further details on this topic.

**Theoretical analysis.** We next check the approximability of our proposed algorithm.

**Lemma 1.** Let \( OPT \) denote an optimal solution. We suppose that there is an identified solutions \( S \) such that \( |S| < b \). Then the following inequality always holds.

\[
\max_{x \in V} \left[ \rho(S \cup \{x\}) - \rho(S) \right] \geq \frac{1}{b} \left( \rho(OPT) - \rho(S) \right) \tag{1}
\]

**Proof.** Proof of Lemma 1 can be checked in C

**Theorem 2.** Algorithm 2 holds \( 1 - \frac{1}{e} \) approximation ratio, i.e., \( \rho(S) \geq (1 - \frac{1}{e}) \rho(OPT) \).

**Proof.** Proof can be checked in D

**Time complexity.** The basic algorithm takes \( O(|V|(|V|+|E|) + b|V|^2) \). The time complexity of each component is as follows.

- For initialization, it takes \( O(|V|(|V|+|E|)) \) to compute all SEGs.
- For each iteration, it is required to compute the set difference \( |V| \) times. Since the set difference takes \( O(|V|) \) time complexity, it takes \( O(b|V|^2) \).

**Limitation of BA.** We point out three major limitations of BA: (1) Memory consumption: it requires considerable memory to store all SEGs. It takes \( O(|V|^2) \) memory space in the worst case. (2) Initialization bottleneck: it requires computing all SEGs in the initial stage, which requires \( O(|V|(|V|+|E|)) \) time complexity. (3) Update bottleneck: at every iteration, we need to check all SEGs to find the best seed node.
In the following sections, we propose an efficient method to address the above three issues. Specifically, ERA focuses on resolving the update bottleneck issue of the BA while preserving its effectiveness. In section 4, we solve these issues simultaneously without losing accuracy.

3.4 Effective \( r \)-neighbors-based approximation algorithm

In section 3.3, we discuss the characteristics of the \( \rho \) function and approximation ratio of BA. Even if BA is effective (a.k.a. \( 1 \sim 1/e \)), it cannot sufficiently handle large datasets. Therefore, in this section, we propose a pruning technique called the \( r \)-neighbors-based pruning strategy to improve efficiency (update bottleneck) while preserving effectiveness. First, we define some terminology.

**Definition 7.** (Effective \( r \)-neighbor size (\( er\)-neighbors))

Given a graph \( G = (V, E) \), a set of seed nodes \( S \) at current iteration \( i \), node \( v \in V \), minimum degree threshold \( k \), and distance threshold \( r \), effective \( r \) neighbor size (\( er\)-neighbors) of \( v \), denoted as \( E_{r}(v) \), is defined as follows.

\[
E_{r}(v) = \begin{cases} |N_r(v)| & \text{default value} \\ \rho(v, S^i) & \text{if } \rho(v, S^i) \text{ where } j < i \text{ is already computed} \end{cases}
\]  

(2)

We can consider that \( er\)-neighbors implies the number of possible engageable nodes based on our observation. This \( er\)-neighbors is helpful to prune a set of nodes as an upper bound. For example, at the initial stage, all \( |N_r(o)| \) is the same with \( er\)-neighbors because all nodes in \( N_r(v) \) can be engaged (without considering the degree constraint). When we compute \( \rho(v, S) \), we update the \( er\)-neighbors value of node \( v \) to keep the recent value.

Note that at the initial stage, we do not need to compute SEGs of all nodes because it takes a long time. We only compute the SEG if it is required to be computed; in other words, we adopt the lazy update manner. Note that \( |E_{r}(v)| \) is always larger than or equal to \( \rho(v, S) \), that is, it can be an upper bound of \( \rho(v, S) \). Thus, we can design the following pruning strategy.

Algorithm 4: Pseudo description of ERA

```plaintext
input : G = (V, E), k, r, b
output : A set of seed nodes S
1 S ← ∅;
2 H ← \{\{v_1, |N_r(v_1)|\}, \{v_2, |N_r(v_2)|\}, \ldots, \{v_{|V|}, |N_r(v_{|V|})|\}\};
3 H ← sortByValue(H);  // Sort by er-neighbors
4 for (|S| ≠ b) {
5    Cur, U ← ;
6    for (u ∈ H.keys()) {
7        \( E_{r}(u) \leftarrow H.get(u) \);
8        if \( \rho(Cur, S) ≥ E_{r}(u) \) then
9            break;
10       end
11      if \( \rho(Cur, S) < \rho(u, S) \) then
12        Cur ← u;
13       end
14       \( E_{r}(u) \leftarrow \rho(u, S) \);
15       U ← U ∪ \{u\};
16      }
17      K ← K ∪ Cur;
18      H ← H \ Cur;
19    Update order U \ {Cur} of H;  // using binary search
20  }
21 return K;
```

Instead of computing all SEGs at the beginning of the algorithm, an effective \( r \)-neighbors-based pruning iteratively computes a few
SEGs that are promising candidates for selection. The high-level idea is to avoid computing SEGs using er-neighbors because er-neighbors of a specific node v is an upper bound of \( \rho(v, S) \). The detailed procedure is described as follows:

**Procedure 2.** At the initial stage, the nodes are ordered based on er-neighbors in descending order (Lines 1-3), then, we compute SEGs to iteratively obtain the \( \rho \) value (Lines 8-11). Note that we retain the current best node \( \text{Cur} \), which has the largest \( \rho \) value (Line 11). As a result of checking the nodes iteratively, if the er-neighbors of the current node is less than or equal to \( \rho \) of \( \text{Cur} \), we return \( \text{Cur} \) as the selected seed node of our algorithm (Lines 8-9). After selecting the best node, we update the node order based on the er-neighbors (Line 16). This process is repeated until we identify \( b \) seed nodes (Lines 4-16). Finally, it returns a set of selected seed nodes as a result (Line 17).

**Example 4.** Figure 5 depicts the procedure 2. The sample graph consists of seven nodes and ten edges. We set \( b = 2, k = 2 \), and \( r = 1 \).

First, we initialize er-neighbors. Next, we sort the nodes according to er-neighbors. Then, we iteratively compute SEGs in a descending order of er-neighbors. In the case of node \( c \), SEG is \( \{a, b, c\} \). When we compute SEG of node \( f \), it contains \( \{d, e, f, g\} \). Because er-neighbors of the next node \( g \) is 4, notice that all \( \rho \) values of the nodes to be computed later are less than 4. This implies that we do not need to compute SEG of the other nodes. Hence, we select the SEG of node \( f \) as a solution and delete node \( f \) from the candidate list. Then, we sort the list based on the size of er-neighbors. We repeatedly compute SEGs. Next, we select the SEG of node \( c \). Because the solution size is exactly the same as the number of seed nodes \( b \), we terminate the algorithm.

Note that if the inverted index is maintained to retain the information regarding which SEG contains a specific node \( u \), we can improve the efficiency. However, this is not preferred, because it requires considerable memory space.

Before we compute an SEG, we must check whether the SEG has already been computed or not. If computed, we do not need to recompute it because the SEG of a node is unique.

**Comparing with [60].** In [60], authors propose an accelerated greedy algorithm (AGA) to find a solution when the objective function is submodular. It iteratively maintains the \( \Delta(v) \) value by computing \( \rho(v) - \rho(\emptyset) \), \( v \in V \) then updates the \( \Delta \) value. Note that the high-level idea of AGA and ERA is similar. The two major differences compared with AGA is as follows: (1) ERA does not directly compute the value \( \rho(v) \) since computing the SEG is time-consuming. Thus, we utilize \( |N_r(v)| \) to improve the efficiency since \( |N_r(v)| \) can be utilized as an upper bound; (2) For every iteration, SEG utilize binary search to maintain the sorted values. Thus, it can improve the efficiency to find the node which has the largest gain.

**Time complexity.** The time complexity of ERA is the same as that of BA since it needs to compute all SEGs in the worst case. However, as discussed in section 5, we observe that ERA is much faster than BA in practice.

### 4 FAST CIRCLE ALGORITHM

In section 3, we discuss two algorithms for finding an approximate solution to the LUEM problem. Despite improving the update bottleneck of BA, ERA intrinsically suffers from memory consumption and initialization bottlenecks. Hence, in this section, we present an approach to improve the efficiency of ERA by improving the abovementioned issues while preserving its effectiveness. Our FCA algorithm is designed based on the assumption that a node with a large degree may engage many users. In Figure 6, we present the degree and size of engageable users when \( r = 1 \) and \( k = 3 \) in the Condmat dataset [49]. Observe that the degree of a node is correlated with the SEG size. In FCA, we utilize this characteristic to find a solution for LUEM.

First, we introduce HyperANF [10], which is a technique used to approximate the neighborhood function. Next, we propose an efficient algorithm that incorporates HyperANF, called the Fast Circle Algorithm (FCA).

#### 4.1 HyperANF

HyperANF [10] is a state-of-the-art algorithm to compute an approximation of the neighborhood function of a graph. It utilizes HyperLogLog counters [32] which are statistical counters requiring \( O(\log \log n) \) bits. Thus, HyperANF can approximate the number of reachable nodes within a specific distance from a node. Unfortunately, the time complexity of the HyperANF is unknown but its running time is expected to be approximately \( O((|V| + |E|)h) \), where \( h \) denotes the maximum distance. This is because HyperANF is an extension of ANF [64] that requires \( O((|V| + |E|)h) \).

#### 4.2 Algorithm description

By utilizing HyperANF, each node has a set of approximated neighborhood values (ANVs) for each distance, that is, given a radius \( r \), each node has \( 1\text{-ANV}, 2\text{-ANV}, \ldots, r\text{-ANV} \). First, we formally define the approximated neighborhood value (ANV) of a node.

**Definition 8.** (Approximated neighborhood value (ANV)).

Given a graph \( G \), node \( v \in V \), and threshold \( r \), the approximated neighborhood value (ANV) is a key-value structure, in which a key is a set of integers from 1 to \( r \) and a value is an approximated number of neighbors from node \( v \) within a specific distance.
Example 5. We illustrate the ANF value by reusing Figure 3. We assume that $r = 3$ and compute the ANF of node $a$. Figure 7 shows the ANF result of the node $a$. Note that ANF does not return the exact degree because it is an approximated value.

The intuition behind FCA is as follows. We focus only on $\text{ANF}_v^r$ to select the best node. After finding node $a$ in the first iteration, we update the ANF values of all nodes. If node $x$ is the $d$-hop neighbor of the selected node $a$, $\text{ANF}_v^r$ must be updated by negating $\text{ANF}_v^{r-d}$ because we assume that $\text{ANF}_v^{r-d}$ nodes are already engaged.

Procedure 3. At the initial stage, given a distance threshold $r$, a set of ANFs ($\text{ANF}^1 \cdots \text{ANF}^r$) for every node based on HyperANF is computed (Lines 1-4). Then, node $u$ that has the largest $\text{ANF}_u^r$ value is selected (Line 8). Intuitively, node $u$ have many $r$-neighbor nodes. Next, we compute $\text{SEG} \mathcal{K}(u)$ and the shortest distances from node $u$; then, construct a shortest-path tree based on the shortest path (Lines 8-10). Next, if node $v$ is in the $i$-th level of the shortest-path tree of $\mathcal{K}(u)$ owing to rooting node $u$, we obtain $\text{ANF}_v^{r-i}$. Then, the ANF values are updated as follows (Lines 11-14).

- $\text{ANF}_v^{r'} = \text{ANF}_v^r - \text{ANF}_v^{r-i}, \forall r' > r - i$
- $\text{ANF}_v^{r'} = 0, \forall r' \leq r - i$

This process is repeated for all nodes in $\mathcal{K}_{r}(u)$. After updating ANF value of a node, we relocate the position of the node to preserve the order in $D$ (Line 17). This procedure is repeated until the size of the solution is $b$.

Example 6. In Figure 8, we give an example of FCA. Suppose that $b = 2$, $r = 2$, and $k = 2$. At the initial stage, we compute ANFs of all nodes. In the Figure, the numbers indicate the ANF values of the nodes. Next, we choose the node that has the largest ANF value. We choose a blue-colored node in the first iteration. After choosing the node, we update the ANF values of its neighbor nodes based on the procedure 3. Among the updated ANF values, we choose the green-colored node that has the largest ANF value. Then, we choose two blue and green nodes as the seed nodes. Consequently, the algorithm is terminated. Note that these users are completely engaged.

Algorithm 5: Procedure of FCA

\begin{itemize}
\item \textbf{input} : $G = (V, E)$, $k$, $r$, $b$
\item \textbf{output} : A set of seed nodes $S$
\item $D \leftarrow \text{G[core}(G, k)], S \leftarrow \emptyset$;
\item initialize HyperANF on $D$;
\item \textbf{for} ($r' \leftarrow 1 : r$)
\item \hspace{1em} Setup ANF$^{r'}$ by computing HyperANF.$\text{next}(r')$;
\item \textbf{end for}
\item $D \leftarrow \text{sortByANF}(D)$;
\item \textbf{while} $|S| \neq b$
\item \hspace{1em} $U \leftarrow \emptyset$;
\item \hspace{2em} $u \leftarrow \text{argmax}_{v \in D} \text{ANF}_v^r$;
\item \hspace{2em} compute $\mathcal{K}_{r}(u)$;
\item \hspace{2em} compute shortest path tree $T(u)$ using $G[\mathcal{K}_{r}(u)]$ with root $u$;
\item \hspace{2em} $d_T \leftarrow \text{maxDistance}(T(u))$;
\item \hspace{2em} \textbf{for} ($y \in \mathcal{K}_{r}(u)$)
\item \hspace{3em} Update the ANF values by Strategy;
\item \hspace{3em} $U.\text{add}(y)$;
\item \hspace{2em} \textbf{end for}
\item \hspace{1em} $D.\text{remove}(u)$;
\item \hspace{1em} $S \leftarrow S \cup \{u\}$;
\item \hspace{1em} update nodes $U$ in $D$;
\item \textbf{end while}
\item \textbf{return} $S$;
\end{itemize}

Time complexity. FCA takes $O(H + br|V| + b(|V| + |E|) \log |V|))$. Time complexity of each component is as follows.

- Computing $\text{SEG}$ takes $O(b(|V| + |E|))$
Table 2: Summary of the real-world datasets

| Name       | # nodes | # edges | $c_{\text{max}}$ | $d_{\text{avg}}$ |
|------------|---------|---------|------------------|------------------|
| Amazon[79] | 334,863 | 925,872 | 6                | 5.52             |
| Brightkite[23] | 58,228 | 214,078 | 52               | 7.35             |
| CondMat[49] | 23,133 | 93,497  | 25               | 8.08             |
| DBLP[79]   | 317,080 | 1,049,866 | 113             | 6.62             |
| Enron[79]  | 36,692  | 183,831 | 43               | 10.02            |
| Hepth[49]  | 9,877   | 25,998  | 31               | 5.26             |
| LA[2, 75]  | 500,597 | 1,462,501 | 120             | 5.84             |
| NYC[2, 75] | 715,605 | 2,552,603 | 157             | 7.13             |
| Orkut[79]  | 3,072,441 | 117,185,083 | 253             | 76.28            |
| Youtube[79] | 1,134,890 | 2,987,624 | 51              | 5.27             |

- Computing HyperANF\(^1\) takes $O(H)$
- Updating $ANV^\ast$ takes $O(br|V|)$
- Computing the shortest path $O(b(|V| + |E| \log |V|))$

5 EXPERIMENTS

We evaluated the proposed algorithms using several real-world networks. All experiments were conducted on Ubuntu 14.04 with 64GB memory and 2.50GHz Xeon CPU E5-4627 v4. For the implementation, we used the Jgrapht library [59] and WebGraph [11]. Our code is publicly available \(^2\). Since the datasets are publicly available, we do not have any preprocessing.

Dataset. Table 2 lists the basic statistics of real-world datasets. All datasets are publicly available. $c_{\text{max}}$ and $d_{\text{avg}}$ denote the maximum core index and average degree, respectively.

Algorithms. As baseline algorithms, we used two graph centrality-based approaches\(^3\), named AC (alpha centrality), and CC (clustering-coefficient). Both approaches compute the centrality measures; then, iteratively pick the node that has the largest centrality until $b$ seed nodes are identified. If SEG of a selected node is null, the node is not selected. Owing to limited scalability, we only report the results of BA in the scalability test.

- Basic algorithm (BA)
- Effective $r$-neighbors-based approximation algorithm (ERA)
- Fast Circle Algorithm (FCA)
- Alpha Centrality-based approach (AC)
- Clustering Coefficient-based approach (CC)

Parameter setting. In our experiments, we vary the values of three parameters $k$, $r$, and $b$. In all the experiments, we use values $r \in [2, 3]$ since the larger values are not interesting since when $r$ becomes large, all the nodes are reachable from the seed nodes. We fix the value $b = 10$ since the larger values may cover all the possible nodes (See Figure 10). For the value $k$, it controlled the cohesiveness level. To the best of our knowledge, many previous works on minimum-degree based cohesive subgraph discovery [4, 29, 30, 44] take the minimum degree as an input parameter and did not shed light on strategies for setting its value. Intuitively, when a user selects a large value $k$, we expect the SEG might be more cohesive and smaller. Thus, we consider the value $k$ an additional degree of freedom available to users to specify the cohesiveness, and users can compare results with different parameters.

Effectiveness and efficiency evaluation in real-world networks. Figures 9a and 9b show effectiveness and efficient of the proposed algorithms on tests conducted using real-world datasets, respectively. Each column indicates different datasets and each row indicates different $r$ values. Each Figure reports the number of engaged users and running time with respect to varying $k$ values. For all experiments, we set $b = 10$. The effectiveness is verified in Figure 9a. Observe that for all cases, ERA algorithm returns the largest number of engaged users, and FCA returns comparable effective results. The two baseline algorithms return relatively low-quality results. In Figure 9b, we check the efficiency. Note that for large-sized datasets, such as LA, NYC, Orkut, and Youtube datasets with $r \geq 2$, ERA algorithm does not finish within 24 hours. Therefore, we did not report these results. We observed that our FCA outperformed the two baseline algorithms for all datasets. Next, we verified the efficiency of these algorithms. We observed that the proposed FCA algorithm is much faster than the two baseline algorithms because it does not need to compute many SEGs.

\(^1\)Note that the time complexity of HyperANF is unclear (See section 4.1)
\(^2\)https://bit.ly/3GyC8w1
\(^3\)In Table 4, we show several centrality measures and notice that AC and CC are relatively scalable to handle large-sized datasets. Hence, we choose both centrality measures as the baseline algorithms.
Measuring node influence. In this experiment, we use the Polbooks network [66] to verify the quality of our seed nodes using node influence measures. The Polbooks network is a well-known network with 105 nodes and 441 edges. The nodes and edges represent blogs on U.S. politics and web links, respectively. Because measuring the influence is time consuming, we used a small graph to check the tendency. In the experiments, we set \( k = 3, r = 2, b = 3 \). In addition, we used five measures to check the influence [68].

| Measures  | Mean of all nodes | ERA | FCA |
|-----------|-------------------|-----|-----|
| clusterRank | 48.0829           | 63.7308 | 63.7195 |
| hubness    | 28.5478           | 58.3913 | 56.9565 |
| IVI        | 12.6734           | 40.4316 | 40.4316 |
| SIRIR      | 1.4515            | 2.224  | 3.139  |
| Spreading score | 19.2765 | 48.6802 | 76.9068 |

Table 4: Centrality score

| Algorithm | Mean of all nodes | ERA | FCA |
|-----------|-------------------|-----|-----|
| AC        | 1.096             | 1.1739 | 1.171 |
| BC        | 104.2912          | 332.9455 | 488.5373 |
| CC        | 0.4902            | 0.3479 | 0.3283 |
| Close     | 0.3329            | 0.3716 | 0.407  |
| CN        | 4.9702            | 6      | 6     |
| IM        | 3.1193            | 5      | 5     |
| HC        | 0.4015            | 0.4671 | 0.4888 |

In Table 5, we verify the number of engaged users when a set of nodes is selected based on centrality measures. Observe that our ERA and FCA algorithms outperform the other algorithms because they aim to maximize the number of engaged users. This indicates that maximizing the number of engaged users cannot be achieved by identifying classic node importance measures.

![Figure 10: Varying parameter b](image1)

Effect on \( b \). To verify the effect on parameter \( b \), we used the Brightkite dataset by setting \( k = 3, b = 2, r = 2 \), and reported the number of distinctly engaged users. In Figure 10, when \( b \) value becomes relatively large, the difference between ERA and FCA algorithms increases. However, when \( b \) value becomes very large (e.g., 5120), the difference is negligible because most nodes are already engaged. Furthermore, we also observe that AC and CC return relatively few engaged users because the larger centrality scores do not indicate a larger number of engaged users.

![Figure 11: Scalability test on synthetic networks](image2)

Scalability test. Figure 11 demonstrate the efficiency of our algorithms with respect to a varying number of nodes in the LFR synthetic networks [48] to present scalability. Notice that the running time of ERA increases significantly when the number of nodes increases. Moreover, observe that the running time of FCA is much faster than that of ERA and increases almost linearly with node size.

Case study: Word-association network Figure 12 depicts the three resultant SEGs in the word-association network [61]. The word-association network consists of 10,617 nodes and 72,168 edges. Each node indicates an English word, and each edge indicates the association between two nodes. Please refer. To avoid
meaningless association, we retain the edges if the edge weight (occurrence) is greater than or equal to 10. By utilizing LUEM with $b = 3$, we identified three seed nodes: Water, Food, and Money. We deduced that the three seed nodes are keywords that represent high-level concepts (topic) to explain the words. For example, in the first SEG, many keywords such as Coast, Leak, Sink, Wet, Port, Gas, Vapor, Pump, and Puddle are related to the seed node Water. In the second SEG, we noticed that most of the keywords, such as Rib, Bread, Pizza, Pastry, Chips, and Spaghetti are related to the keyword Food. Notice that each SEG is a cluster of similar keywords.

**Case study : DBLP.** We use DBLP dataset [43] for our case study. We generate a co-authorship network where an edge of two authors is generated if they publish at least 3 papers together. We set $k = 3$ and $r = 2$. Figure 13 shows the result of the case study. We notice that the author 1621 is the center of SEG and all the nodes are reachable from 1621 within 2 hops. We observe that there are two major sub-communities: (1) two research groups of Academy; (2) a research group of an institute.

**Case study : Community Search.** In this experiment, we demonstrate the results when our LUEM approach is utilized on the community search problem [31, 73]. We use three representative community search models as baseline algorithms: the $k$-core model [73], $k$-truss model [38], and $k$-ecc model [19] among many community search models [4, 38, 39, 46, 77] since the implementation of the algorithms is publicly available [31]. To verify the accuracy, we use two representative metrics for the community detection problem: NMI (Normalized Mutual Information) [25] and ARI (Adjusted Rand Index) [40]. Because both measures are designed to identify the best partitions (communities), we consider the community search problem as a binary classification problem to utilize both measures. We use three datasets (Amazon, DBLP, and Youtube [79]) that are reported in Table 2, which have ground-truth communities. Because the ground-truth communities overlap, we compare the identified community with all ground-truth communities containing the query node; then, select the best accuracy value. For each experiment, we randomly picked 10 query nodes, set $k = 3$ and $r = 1$, and reported the average and standard deviation. Figure 14 reports the results of LUEM and two baseline algorithms. Observe that in the Amazon dataset, our model returns a better result than that of the $k$-core model and a result comparable to that of the $k$-truss model. In DBLP and Youtube datasets, observe that our model achieves better accuracy than those of the $k$-core, $k$-truss, and $k$-ecc models. This is because the size of ground-truth communities is relatively small and they normally have a small diameter.

![Figure 13: DBLP result when $k = 3$](image)

**Figure 13: DBLP result when $k = 3$**

![Figure 14: Community search experiment](image)

**Figure 14: Community search experiment**

![Figure 15: Diameter distribution](image)

**Figure 15: Diameter distribution**

Notice that approximately 80% of communities in DBLP dataset and 95% of communities in Youtube had a diameter of less than or equal to 5. Because our community model is a minimum-degree with a diameter-bounded community model, we can obtain a better accuracy score if the size of the ground-truth communities are very small.

## 6 RELATED WORK

### 6.1 Influence maximization

The influence maximization (IM) problem [50], which is a key problem in social network analysis, has recently gained much attention owing to its potential value. Given a graph $G = (V, E)$, the influence maximization problem aims to find a set of $k$ seed nodes while maximizing the expected number of users influenced by the seed nodes. A representative application of IM is viral marketing [21, 37]. Kempe et al. [42] first modeled the influence maximization problem for two fundamental information diffusion models: (1) an independent cascade model and (2) a linear threshold model. Most existing IM problems can be solved by applying the greedy hill-climbing algorithm. Because the objective function of the IM problem is non-negative monotone submodular, if the best node to maximize the
number of expected influenced users are chosen, a $1 - \frac{1}{e}$ approximation ratio is achieved.

### 6.2 Anchored and collapsed $k$-core

The anchored $k$-core problem was first proposed by Bhawalkar et al. [8]. Given a graph $G$, integer $k$, and budget $b$, the anchored $k$-core problem involves identifying $b$ anchor nodes to maximize the number of engaged users. Note that $b$ anchor nodes are fixed to be engaged, even if they do not have sufficient neighbor nodes to be engaged. In [8], the authors showed that when $k = 2$, polynomial-time algorithms could be designed. When $k \geq 3$, the inapproximability results were proved. Zhang et al. [83] proposed the OLAK algorithm for the anchored $k$-core problem. They proposed an onion layer structure to significantly reduce the search space for this problem. They also proposed early termination and pruning techniques to improve efficiency. Linghu et al. [51] proposed an anchored coreness problem. Instead of maximizing the number of engaged users, they focused on the coreness gain by anchoring the $b$ nodes. They presented that the anchored coreness problem was also NP-hard and proposed a pruning search space technique and method to reuse the intermediate results to improve efficiency. Cai et al. [18] proposed an attributed community engagement problem that considers the attributes of users and community cohesiveness. They aimed to identify $l$ anchored users that can induce a maximal expanded community, namely, Anchored Vertex set Exploration (AVE) problem. They demonstrated that, when $k \geq 3$, the AVE problem was NP-hard. To solve this problem, they proposed the filter-verify algorithm with early termination and pruning techniques. In [47], the authors proposed residual core maximization, an algorithm for the anchored $k$-core problem. They selected anchored nodes based on two strategies: residual degree and anchor score. Moreover, they showed that the result is close to the optimal solution.

The collapsed $k$-core problem is firstly proposed by Zhang et al. [84]. They focused on the problem of finding $b$ collapsers to minimize the number of engaged users in a social network when removing $b$ collapsers. The identified $b$ anchor nodes can be considered as important users for maintaining their engagement in a network. Luo et al. [54] proved that when $k \geq 3$, the collapsed $k$-core problem was W[1] hard. Zhang et al. [81] introduced a collapsed $k$-truss problem.

### 6.3 $k$-core Decomposition and Its Variation

The $k$-core is widely used for finding cohesive subgraphs in a graph. The definition of the $k$-core [72] is as follows. Given a graph $G = (V, E)$ and integer $k$, the $k$-core, denoted by $D_k$, is a set of nodes of which every node has at least $k$ neighbor nodes in $D_k$. This $k$-core is unique and has containment relationship, i.e., $D_k \subseteq D_{k+1}$ when $k-1 \geq 1$. The core index of a node is $k$ if it belongs to the $k$-core but not to the $(k+1)$-core. Note that $k$-core is a set of nodes of which its core index is larger than or equal to $k$. Batagelj et al. [5] propose an exact and efficient algorithm to find the core index. Sarıyuce et al. [69] focuses on incremental $k$-core problem in a dynamic graph. Bonchi et al. [15] formulate distance-generalized $k$-core named $(k, h)$-core. Zhang et al. [80] formulate $(k, p)$-core by considering the degree ratio in $k$-core. Zhang et al. [82] formulate $(k, s)$-core by unifying $k$-truss and $k$-core. There are several variations of $k$-core in signed networks [35], directed networks [36], weighted graph [28, 33], temporal graph [76], multi-layer graph [33], bipartite graph [27], and uncertain graph [14]. To get more details, please refer to this nice survey paper [55].

### 7 CONCLUSION

In this study, we formulate a novel problem called Local User Engagement Maximization (LUEM) by integrating the local user engagement with graph distance. We prove that the LUEN problem is NP-hard and its objective function is monotonic submodular. To solve this problem, we propose two approximation algorithms and an efficient heuristic algorithm that preserves effectiveness. To demonstrate the superiority of our algorithms, we conducted extensive experiments using real-world and synthetic networks. As a future research direction, we will consider a dynamic environment to find $b$ seed nodes. For example, the graph can be changed dynamically, or the end user may change the parameters online.

### A EFFICIENCY AND EFFECTIVENESS TEST WHEN $r = 1$

Figure 16 shows effectiveness and efficiency results for $r = 1$. Observe that the results have trends similar to those shown in Figures 9a and Figures 9b. One remarkable difference is that our FCA is slightly slower than the baseline algorithms because it requires some initialization steps for approximation. When $r \geq 2$, the initialization time is not significantly affected.

### B PROOF OF THEOREM 1

To show the NP-hard of a specific problem, it is required to show an example of an NP-hard problem that can be reduced to the problem that we want to verify the hardness. To show the hardness of LUEN problem, we utilize the set-cover problem [7]. The set-cover problem is defined as follows: given a set of elements $U = \{u_1, u_2, \ldots, u_p\}$, $z$ subsets $X = \{X_1, X_2, \ldots, X_z\}$ of $U$, a set of assignments $E_1$ from $X$ to $U$ where $X_i \in X$, and a parameter $b^*$, if $b^* > 0$, it returns YES if there exists a set of $b^*$ subsets whose union is the same with $U$. Otherwise, it returns NO. To present a reduction, we suppose that we have a solution of
LUEM. Next, we start with an arbitrary instance of the set-cover problem, and show that our LUEM problem can be utilized to solve the set-cover problem. Let suppose that we have an instance of the set-cover problem \( I_{SC} = (X, U, E, b') \). By utilizing the instance \( I_{SC} \), we can construct a new instance \( I_{LUEM} \). \( I_{LUEM} \) contains three key elements: (1) \( V_1' \) is the same with \( X \); (2) \( V_2' \) is the same with \( U \); and (3) \( V_3' \) is \( z \) sets of \( (y + z) \) entities. A subset \( X_j \) is connected to the element \( i \) if \( X_j \) contains the element \( i \). We denote this relationship as \( E_1 \). A subset \( X_j \) is connected to the \( j \)-th \( (y + z) \) entities in \( V_2' \). We set \( b = b', k = 1 \), and \( r = 1 \). The newly generated instance and graph \( G' \) can be checked in Figure 17.

For every entity in \( V_3' \), it can have at most \( z \) degree since the number of entities in \( V_3' \) is \( z \). All the entities in \( V_3' \) has exactly one neighbor entity in \( V_1' \). Interestingly, a node in \( V_1' \) has at least \( z + y \) degree since it is connected to the nodes in \( V_3' \). Since \( k = 1 \) and \( r = 1 \), the size of SEG of node \( v \) is the size of the ego-network of node \( v \) in \( G' \).

## C PROOF OF LEMMA 1

We denote \( OPT \setminus S = \{v_1, v_2, \ldots, v_l\} \) where \( l \leq b \). We then easily notice that

\[
\rho(OPT) \leq \rho(OPT \cup S)
\]

\[
\leq \rho(S) + \sum_{j=1}^{l} [\rho(S \cup \{v_1, \ldots, v_j\}) - \rho(S \cup \{v_1, \ldots, v_{j-1}\})]
\]

\[
\leq \rho(S) + \sum_{j=1}^{l} [\rho(S \cup \{v_j\}) - \rho(S)]
\]

\[
\leq \rho(S) + \sum_{x \in V} \max_{i=1}^{l} [\rho(S \cup \{x\}) - \rho(S)]
\]

\[
\leq \rho(S) + b' \max_{x \in V} [\rho(S \cup \{x\}) - \rho(S)]
\]

(3)

It indicates that the following inequality holds.

\[
\rho(OPT) \leq \rho(S) + b' \max_{x \in V} [\rho(S \cup \{x\}) - \rho(S)]
\]

\[
\frac{1}{b'} (\rho(OPT) - \rho(S)) \leq \max_{x \in V} [\rho(S \cup \{x\}) - \rho(S)]
\]

(4)

Hence, our Lemma 1 holds.

## D PROOF OF THEOREM 2

Let denote \( K' \) as the solution of our algorithm at the end of the iteration \( i \leq b \). Then, we reuse the Lemma 1.

\[
\rho(K') - \rho(K' - i) \geq \frac{1}{b'} (\rho(OPT) - \rho(K' - i))
\]

\[
\rho(K') - \rho(K' - i) \geq \frac{1}{b'} \rho(OPT) - \frac{1}{b'} \rho(K' - i)
\]

\[
\rho(K') \geq \frac{1}{b'} b \rho(OPT) + (1 - \frac{1}{b'}) \rho(K' - i)
\]

\[
\geq \frac{1}{b} b \rho(OPT) + \frac{1 - \frac{1}{b'}}{b'} \rho(OPT) + (1 - \frac{1}{b'}) \rho(K' - i)
\]

\[
\geq \frac{1}{b} (1 - \frac{1}{b'}) b \rho(OPT) = [1 - (1 - \frac{1}{b})] \rho(OPT)
\]

(5)

Therefore, we notice that \( \rho(K') \geq (1 - \frac{1}{b'}) \rho(OPT) \).

In \( G' \), we can find a solution of LUEM with \( b = b' \), \( k = 1 \), and \( r = 1 \). To find a solution, we must choose the nodes in \( V_3' \) since it can make at least \( 1 + y + z \) nodes to be engaged as we discussed. Thus, if we find a solution of LUEM in \( G' \) and it covers all the nodes in \( V_3' \), we can solve the set-cover problem with parameter \( b' \). Hence, LUEM problem is NP-hard.

## REFERENCES

[1] Lars Backstrom, Paolo Boldi, Marco Rosa, Johan Ugander, and Sebastiano Vigna. 2012. Four degrees of separation. In WWW.

[2] Jie Bai, Yu Zheng, and Mohamed F. Mokbel. 2012. Location-based and preference-aware recommendation using sparse geo-social networking data. In SIGSPATIAL.

[3] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. science 286, 5439 (1999), 509–512.

[4] Nicola Barbieri, Francesco Bonchi, Eduardo Galimberti, and Francesco Gullo. 2015. Efficient and effective community search. Data mining and knowledge discovery 29, 5 (2015), 1406–1433.

[5] Vladimir Batagelj and Matjaz Zaversnik. 2003. An O (m) algorithm for cores decomposition of networks. arXiv preprint cs/0310049 (2003).

[6] Alex Bavelas. 1950. Communication patterns in task-oriented groups. The journal of the acoustical society of America 22, 6 (1950), 725–730.

[7] KORTE Bernhard and JENS Vygen. 2008. Combinatorial optimization: Theory and algorithms. Springer, Third Edition, 2005. (2008).

[8] Kuhịa Bhawalkar, Jon Kleinberg, Kevin Lewi, Tim Roughgarden, and Aneesh Sharma. 2012. Preventing unranking in social networks: the anchored k-core problem. In ICALP. 440–451.

[9] Kuhịa Bhawalkar, Jon Kleinberg, Kevin Lewi, Tim Roughgarden, and Aneesh Sharma. 2015. Preventing unranking in social networks: the anchored k-core problem. SDM 29, 3 (2015), 1452–1475.

[10] Paolo Boldi and Sebastiano Vigna. 2004. The webgraph framework I: compression techniques. In WWW. 595–602.

[11] Phillip Bonacich. 1987. Power and centrality: A family of measures. American journal of sociology 92, 5 (1987), 1170–1182.

[12] Phillip Bonacich and Paulette Lloyd. 2001. Eigenvector-like measures of centrality for asymmetric relations. Social networks 23, 3 (2001), 191–201.

[13] Francesco Bonchi, Francesco Gullo, Andreas Kaltenbrunner, and Yana Volkovich. 2014. Core decomposition of uncertain graphs. In SIGKDD. 1316–1325.

[14] Francesco Bonchi, Arijit Khan, and Lorenzo Severini. 2019. Distance-generalized core decomposition. In SIGMOD. 1006–1023.

[15] Stephen P Borgatti and Martin G Everett. 2006. A graph-theoretic perspective on centrality. Social networks 28, 4 (2006), 464–484.

[16] Ulrik Brandes. 2001. A faster algorithm for betweenness centrality. Journal of mathematical sociology 25, 2 (2001), 163–177.

[17] Taotao Cai, Jianxin Li, Nur Al Hasan Haldar, Ajmal Mian, John Yearwood, and Tamos Sellis. 2020. Anchored Vertex Exploration for Community Engagement in Social Networks. In ICDE. IEEE, 409–420.
[80] Chen Zhang, Fan Zhang, Wenjie Zhang, Boge Liu, Ying Zhang, Lu Qin, and Xuemin Lin. 2020. Exploring finer granularity within the cores: Efficient (k, p)-core computation. In 2020 IEEE 36th International Conference on Data Engineering (ICDE). IEEE, 181–192.

[81] Fan Zhang, Conggai Li, Ying Zhang, Lu Qin, and Wenjie Zhang. 2018. Finding critical users in social communities: The collapsed core and truss problems. TKDE 32, 1 (2018), 78–91.

[82] Fan Zhang, Long Yuan, Ying Zhang, Lu Qin, Xuemin Lin, and Alexander Zhou. 2018. Discovering strong communities with user engagement and tie strength. In DASFAA. Springer, 425–441.

[83] Fan Zhang, Wenjie Zhang, Ying Zhang, Lu Qin, and Xuemin Lin. 2017. OLAK: an efficient algorithm to prevent unraveling in social networks. VLDB (2017), 649–660.

[84] Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2017. Finding critical users for social network engagement: The collapsed k-core problem. In AAAI. 245–251.

[85] Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2017. Finding critical users for social network engagement: The collapsed k-core problem. In AAAI (San Francisco, California, USA), Vol. 31. 245–251.

[86] Zhongxin Zhou, Fan Zhang, Xuemin Lin, Wenjie Zhang, and Chen Chen. 2019. K-Core Maximization: An Edge Addition Approach. In IJCAI. 4867–4873.

[87] Weijie Zhu, Chen Chen, Xiaoyang Wang, and Xuemin Lin. 2018. K-core minimization: An edge manipulation approach. In CIKM. 1667–1670.