OCSM : Finding Overlapping Cohesive Subgraphs with Minimum Degree

Junghoon Kim
Nanyang Technological University
Singapore
junghoon001@e.ntu.edu.sg

Sungsu Lim
Chungnam National University
South Korea
sungsu@cnu.ac.kr

Jungeun Kim
Kongju National University
South Korea
jekim@kongju.ac.kr

ABSTRACT
Cohesive subgraph discovery in a network is one of the fundamental problems and investigated for several decades. In this paper, we propose the Overlapping Cohesive Subgraphs with Minimum degree (OCSM) problem which combines three key concepts for OCSM: (i) edge-based overlapping, (ii) the minimum degree constraint, and (iii) the graph density. To the best of our knowledge, this is the first work to identify overlapping cohesive subgraphs with the minimum degree by incorporating the graph density. Since the OCSM problem is NP-hard, we propose two algorithms: advanced peeling algorithm and seed-based expansion algorithm. Finally, we show the experimental study with real-world networks to demonstrate the effectiveness and efficiency of our proposed algorithms.

ACM Reference Format:
Junghoon Kim, Sungsu Lim, and Jungeun Kim. 2018. OCSM : Finding Overlapping Cohesive Subgraphs with Minimum Degree. In Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX). ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXX.XXXXXXX

1 INTRODUCTION
1.1 Motivation
With recent rapid and important developments in mobile and IT technology, many people have started using Social Networking Services (SNSs) all the time and everywhere. Considering the vast number of social networks, the mining of cohesive subgraphs in a social network has been widely studied [12, 36] even if there is no formal definition. Normally, a cohesive subgraph is considered to be a group of users that are highly connected with each other. Recently, many cohesive subgraph models are proposed including k-core [36], \((\alpha, \beta)\)-core [21] k-clique [37], and k-truss [12, 46]. Among them, k-core [36] is the most popular and widely used model owing to its simple and intuitive structure. The definition of k-core [36] is as follows: given a graph G and a positive integer k, a k-core, denoted as D, is a maximal subgraph of which all nodes in the subgraph have at least k neighbor nodes in D. k-core has many applications, such as community search problem [13, 16, 25, 40], user engagement maximization problem [8, 30, 44]. Furthermore, it is known that the k-core can play a role as a subroutine for much harder problems [24, 47] and can be utilized in different networks [3].

Even if k-core is widely used and has many applications, k-core intrinsically suffers from several limitations due to its definition: (i) it returns a relatively large solution, especially when the value of k is small, i.e., it may contain loosely connected nodes; (ii) it always returns a disjoint result, i.e., it cannot reveal overlapping structures.

The reason for the large solution of k-core is its maximality constraint. In an Amazon dataset [42], when we apply 3-core, 98% of the nodes belong to a single giant connected component. Similarly, in a Youtube dataset, 100% of the nodes belong to a single giant connected component.

Many studies show that people in a real social network can be intrinsically characterized by multiple cluster memberships [14, 41]. Kelley et al. [23] shows that membership overlap is a significant characteristic of many real-world social networks. We can easily notice that a cohesive subgraph structure can overlap. In real life, people can belong to multiple groups, such as a dance club, table tennis club, family, graduate student association, and so on and can be engaged in all these groups. It indicates that the cohesive subgraphs can overlap. Therefore, in our paper, we focus on finding overlapping cohesive subgraphs.

1.2 OCSM Problem
To handle the problem of k-core, in this paper, we propose an Overlapping C cohesive Subgraph with Minimum degree (OCSM) problem by resolving the limitation of the k-core.

At first, to incorporate the overlapping structure into the cohesive subgraph discovery problem, we use a line-graph [15] which represents the adjacencies between edges of a network. This line-graph helps in identifying the latent structures by changing the perspective from the node-level to the edge-level. However, not only the original line-graph [15] but also its extension, called the link-space graph [29], suffer from efficiency and effectiveness problems. Thus, we propose a link-skein graph, which is a subset of the link-space graph with the edges which form high-order structures (e.g., triangles) in the original graph in order to preserve meaningful information, while significantly improving the efficiency.

Next, to avoid finding a large solution with loosely connected nodes, we incorporate the graph density [17] into the cohesive subgraph discovery problem. By maximizing the graph density of the cohesive subgraphs, we can achieve more cohesive subgraphs as a result. As we aim to find multiple overlapping cohesive subgraphs, we newly define a link-density, which is an extension of the graph density for link-skein graphs.
Table 1 briefly compares the result of OCSM with k-core and the densest subgraph (DS) discovery problem. Only OCSM can retrieve the top t overlapping subgraphs while satisfying the minimum degree constraint. Furthermore, the OCSM problem is not trivial as it is proven to be NP-hard.

Table 1: Comparison of the OCSM, k-core, and densest subgraph discovery (DS).

| Constraint | OCSM [this work] | k-core [36] | DS [17] |
|------------|------------------|------------|---------|
| Overlap    | Yes              | No         | No      |
| Result     | top t dense subgraphs | a set of nodes | the densest subgraph |
| Objective  | max. density     | max. size  | max. density |

To solve the OCSM problem, we propose two heuristic algorithms: (i) an advanced peeling algorithm (APA) and (ii) a seed-based expansion algorithm (SEA). The high-level idea of the proposed algorithms is as follows. The first is a top-down approach which iteratively deletes a set of nodes to maximize the link-density while satisfying the degree constraint. In contrast, the latter is a bottom-up approach which identifies the densely connected seed nodes, and then iteratively adds a set of nodes to satisfy the degree constraint. These procedures are iteratively repeated until the top t subgraphs are found.

Example 1. Figure 1 shows the subgraph mining results obtained by the OCSM problem and a k-core with two toy networks. To present the overall cohesive subgraph structure, we put the high-level community structure on the right-side of the figures. First, Figure 1a shows a simple network with 11 nodes and two cohesive subgraphs with one overlapping hub node. When we use k-core, it fails to find two dense cohesive subgraphs. This is because the hub node has a high degree and connects the two cohesive subgraphs, even if the edges of the hub node are not related to each other. Even though the density of each cohesive subgraph is larger than that of the whole graph, k-core always identifies the whole graph as a result. Thus, it also fails to find a hub (overlapping) node. Note that OCSM can retrieve two cohesive subgraphs that overlap at a node. Next, Figure 1b shows the case in which k-core cannot identify cohesive subgraphs and cannot specify the number of cohesive subgraphs. We notice that there are two bridge edges between the cohesive subgraphs. These bridge edges connect two cohesive subgraphs, then, k-core returns large subgraphs. Even if the user already knows the number of cohesive subgraphs in advance, k-core cannot incorporate this information. Note that the OCSM can identify three cohesive subgraphs since our approach lessens the influence of the bridge edges.

1.3 Key Contributions

- **Problem Significance:** We formally define the OCSM problem. To the best of our knowledge, this is the first work to find top t densely connected overlapping subgraphs discovery with a minimum degree constraint.
- **Solution:** We theoretically show that our problem is NP-hard and propose two heuristic algorithms for addressing the OCSM problem.
- **Extensive Evaluations:** We conduct extensive experiments on real-world datasets to check the efficiency and effectiveness. Furthermore, an interesting case study shows that our solution successfully discovers densely connected overlapping subgraphs.

2 PROBLEM STATEMENT

In this section, we formally introduce our problem and its hardness. We assume that all graphs considered in this work are simple and undirected. Given a subset of nodes $V' \subseteq V$, we denote $G[V'] = (V', E[V'])$ the subgraph of $G = (V, E)$ induced by $V'$, i.e., $E[V'] = \{i, j \in E | i, j \in V'\}$. The basic notations are summarized in Table 2.

| Notation                     | Definition                                                                 |
|------------------------------|----------------------------------------------------------------------------|
| $G = (V, E)$                 | an original graph                                                         |
| $L(G) = (V_L(G), E_L(G))$    | the link-skein graph of $G$                                                |
| $e_{i,j}$                    | an edge of the nodes $i$ and $j$ in $G$                                    |
| $v_{i,j}$                    | a node generated by $e_{i,j}$ in $L(G)$                                   |
| $R(H)$                       | a set of nodes in $G$ from $H \subseteq V_L(G)$                          |
| $E_L(G)$                     | edges of link-skein graph of $G$                                          |
| $N(v)$                       | a set of neighbor nodes of node $v$                                       |
| $w(v, u)$                    | the edge weight of $v$ and $u$ in $L(G)$                                  |
| $\delta(G)$                 | min. degree of graph $G$                                                  |
| $\beta(H)$                   | min. occurrence of $H \subseteq V_L(G)$                                  |
| $\gamma(H)$                  | link-density of $H \subseteq V_L(G)$                                    |

2.1 Link-Space and Link-Skein Graphs

We first introduce the link-space [26, 29] and link-skein graphs, which have several benefits for the overlapping cohesive subgraphs discovery.

**Definition 1.** Link-space graph [29]. Given a graph $G$, its corresponding link-space graph $L_S(G)$ is defined as follows:
Given a graph \( G \), \( \Gamma(i) \) represents the set of nodes \( i \) is adjacent to in \( G \). Two nodes \( i, j \) in \( G \) are adjacent if and only if \( \Gamma(i) \cap \Gamma(j) \neq \emptyset \). A node \( v \) in \( G \) is adjacent to \( i \) if \( \Gamma(i) \cap \Gamma(v) \neq \emptyset \). A set of nodes \( C \subseteq V(G) \) is a clique if and only if \( \Gamma(i) \cap \Gamma(j) = \emptyset \) for all \( i, j \) in \( C \) when \( i \neq j \). A clique \( C \) is maximal if any node in \( V(G) \) \( \setminus \) \( C \) is not adjacent to all nodes in \( C \). The clique number of \( G \) is \( \chi(G) \), the maximum size of a clique in \( G \). Given a graph \( G \), its link-space graph \( L(G) \) is defined as follows: \( L(G) = (V(L(G)), E(L(G))) \), where \( V(L(G)) = \{ \{i, j\}, \{j, k\}, \ldots \} \) if \( \Gamma(i) \cap \Gamma(j) \neq \emptyset \) and \( \Gamma(j) \cap \Gamma(k) \neq \emptyset \) for all \( i, j, k \) in \( V(G) \) when \( i \neq j \neq k \neq \ldots \). \( E(L(G)) = \{ \{i, j\}, \{j, k\}, \ldots \} \) if \( \{i, j\}, \{j, k\}, \ldots \) are adjacent in \( L(G) \).

**Definition 2. Link-skein graph.** Given a graph \( G \), its corresponding link-skein graph \( L(G) \) is defined as follows:

- A node \( v_{i,j} \) in \( L(G) \) represents the link \( e_{i,j} = \{i, j\} \) in \( G \).
- Two nodes \( v_{i,k} \) and \( v_{j,k} \) in \( L(G) \) are adjacent if and only if their corresponding links are contained in a triangle in \( G \).
- The weight \( w(v_{i,k}, v_{j,k}) \) on the link \( \{v_{i,k}, v_{j,k}\} \) in \( L(G) \) is assigned by similarity \( \sigma(e_{i,k}, e_{j,k}) \) calculated on \( G \).

We notice that a link-skein graph is a spanning subgraph of a link-space graph, or a graph sparsification due to the elimination of less important edges. Figure 2 contrasts the link-skein graph with the link-space graph. The link-space graph has 10 edges, whereas the link-skein graph has only 6 edges with the same weights. The edges with the smallest weights in the link-space graph do not appear in the link-skein graph.

### 2.2 Overlapping Cohesive Subgraph with Minimum Degree

We first introduce some basic definitions before introducing our problem.

**Definition 3.** \( k \)-core \([36]\). Given a graph \( G = (V, E) \) and positive integer \( k \), \( k \)-core of \( G \), denoted by \( D_k \), is a maximal subgraph consisting of a set of nodes which all the nodes in \( D_k \) have at least \( k \) neighbors.

There are two important characteristics of the \( k \)-core: (1) **Uniqueness**: given a graph \( G \) and integer \( k \), \( k \)-core is unique due to its maximality constraint; (2) **Hierarchical structure**: \( k \)-core has a hierarchical structure, i.e., \( (k + 1) \)-core \( \subseteq k \)-core \( \subseteq (k - 1) \)-core when \( k > 1 \). As \( k \)-core satisfies the minimum degree constraint, we can use a set of connected components of \( k \)-core as the baseline of our algorithm. We next discuss our objective function.

**Definition 4.** **Link-density.** Given sets of nodes \( C = \{c_1, c_2, \ldots, c_t\} \) where \( \forall c \in C \), \( c \subseteq V(L(G)) \), the link-density of \( C \) is defined as follows:

\[
\gamma(C) = \frac{\sum_{c \in C} w'(i, j)}{|V_c|},
\]

where \( V_c \) is a set of nodes \( c \subseteq V(L(G)) \) in the link-skein graph and \( E_c \) describes the edges in the link-skein graph, which is induced by a subgraph \( c \), and \( w'(i, j) = \frac{w(L_i, L_j)}{O(L_i, L_j)} \), where \( O(i, j) \) indicates the number of appearances of \( e_{i,j} \) in the subsets of \( C \).

**Example 3.** In Figure 2, suppose that \( t = 1 \) and we have two candidate subgraphs induced by the nodes \( \{1, 2, 3\} \) (small cohesive subgraph) and \( \{1, 2, 3, 4, 5\} \) (large cohesive subgraph). We then check the link-densities of the two candidate subgraphs in the link-space and link-skein graphs, respectively. Figure 3 reports the link-densities of the candidate subgraphs. We can notice that the link-space graph prefers large-sized subgraphs in terms of the link-density. However, our link-skein graph does not prefer large subgraphs. This helps us in identifying more densely connected overlapping cohesive subgraphs.

Instead of directly using traditional graph density, we develop a new graph density measure for finding overlapping cohesive subgraphs. The rationale behind adding the occurrence term \( O(i, j) \) is to prevent finding dense subgraphs, which commonly share nested dense subgraphs. For example, given a graph \( G \) and \( k = 1 \), suppose that there is a clique \( C \subseteq V \). Then, finding top \( t \) subgraphs is equal to finding the clique \( C \) \( t \) times as this can maximize the graph density. It
indicates that the traditional graph density measure is not proper for finding overlapping dense subgraphs. To handle this problem, some studies adopted additional hyper-parameters, such as the overlapping ratio threshold \([4]\) or distance \([18]\). In this paper, we do not use any thresholds or constraints to control the overlapping ratio. Instead of controlling the overlapping ratio, we use the link-density, which has the effect of lessening the edge weight if the edge has already been selected. When \(t = 1\), the link-density is the same as the traditional graph density.

**Property 1.** Link-density is less sensitive than graph density when handling nested subgraphs.

Let us suppose that we have two subgraphs \(C_1\) and \(C_2 = C_1 \cup \{u\}\) in a link-skein graph. When we calculate the graph density, it is\[
\frac{\sum_{(u,v) \in E_C} w(u,v)}{|V_C|} + \frac{\sum_{(u,v) \in E_{C'}} w(u,v)}{|V_{C'}|}.
\]
If \(C_1\) is densely connected, it can easily be that \(C_2\) may have a high graph density since \(C_2\) contains \(C_1\). It indicates that when we would like to find \(t\) subgraphs and \(C_1\) is a clique, we may find \(t\) subgraphs of which each contains \(C_1\).

In section 2.3, we discuss the differences between link-space and link-skein graphs for small-sized datasets.

![Graph G and Link-skein graph L(G)](image)

**Figure 4: A toy network containing 6 nodes**

**Example 4.** We next present an example for computing the link-density when \(t > 1\). Suppose that \(t = 2\) and we have two candidate solutions.

- \(S_1 = \{v_{1,2}, v_{1,6}, v_{2,6}\}, \{v_{1,2}, v_{1,3}, v_{2,3}\}\)
- \(S_2 = \{v_{1,2}, v_{1,6}, v_{2,6}\}, \{v_{1,2}, v_{1,3}, v_{2,3}, v_{1,3}, v_{2,3}\}\)

The link-densities of two solutions are as follows.

\[
\gamma(S_1) = \frac{0.5 + 0.5 + 1}{3} + \frac{0.5 + 0.5 + 1}{3} = 1.3333
\]

\[
\gamma(S_2) = \frac{0.25 + 0.25 + 0.5}{3} + \frac{0.25 + 0.25 + 0.5 + 0.5 + 0.5 + 1}{5} = 0.9333
\]

We can notice that we prefer \(S_1\) to \(S_2\) without requiring any specific parameters.

**Definition 5.** Minimum occurrence. Given a subgraph \(H\) of link-skein graph \(L(G)\), the minimum occurrence of \(H\), denoted \(\beta(H)\), is the minimum number of node occurrences when the link-skein graph \(H\) is translated back to the original graph \(R\).

**Example 5.** In Figure 2, \(\beta(\{v_{1,2}, v_{1,3}, v_{2,3}\}) = 2\) as nodes 1, 2, and 3 appeared twice. \(\beta(\{v_{1,2}, v_{1,3}, v_{2,3}, v_{1,4}\}) = 1\) as node 4 appeared once.

Note that the minimum degree is closely related to the minimum occurrence. We notice that the following property always holds.

**Property 2.** If a subgraph \(H \in V_t(G)\) satisfies the minimum occurrence, it indicates that a subgraph \(R(H)\) satisfies the minimum degree constraint.

**Proof.** The proof is trivial.

Given a link-skein graph \(H\) of \(L(G)\), \(R(H)\) is a set of nodes in \(G\) which are translated back from the link-skein graph \(H\). Now, we are ready to introduce our OCSM problem.

**Problem Definition 1.** (Overlapping Cohesive Subgraphs with Minimum degree (OCSM)). Given a graph \(G = (V, E)\), a positive integer \(k\), and the desired number of subgraphs \(t\), OCSM aims for finding a set of subgraphs \(H = \{H_1, H_2, \cdots, H_t\}\) where \(\forall H_i \in H, H_i \subseteq V_t(G)\) such that

- \(\forall H_i \in H, R(H_i)\) is connected.
- \(\forall H_i \in H, \delta(R(H_i)) \geq k\).
- \(\gamma(H)\) is maximized.

We call \(\forall H_i \in H, \delta(R(H_i)) \geq k\) as the degree constraint. Note that satisfying the degree constraint does not guarantee that the corresponding link-skein graph is connected. In Figure 2, the link-skein graph has two connected components, even if the original graph is connected.

**Theorem 1.** The OCSM problem is NP-hard.

**Proof.** We prove that our problem is NP-hard by reducing an instance of the \(pDSS\) [4] problem to our problem. \(pDSS\) problem is defined as follows: Given a graph \(G\) and a positive integer \(p\), and \(\alpha \in [0, 1]\), \(pDSS\) aims to find at most \(p\) overlapping subsets \(S_1, S_2, \cdots, S_p\) of the nodes, such that \(\sum \alpha \rho(S_j)\) is maximized such that \(1 - \alpha \rho(S_j) \leq \alpha, \forall S_i, S_j \in S\). They show that \(pDSS\) problem is NP-hard.

Our reduction procedure is as follows. Suppose that we have an instance of \(pDSS\): \(pDSS = (G, p, \alpha = 1)\). We can easily create an instance of our problem: \(I_{OCSM} = (G', k = 1, t = p)\) where \(L(G') = G\). Then, finding the top \(p\) densest subgraphs in \(pDSS\) is exactly the same as finding a solution of \(I_{OCSM}\) since \(\alpha = 1\). Therefore, we can guarantee that our problem is NP-hard.

**Theorem 2.** Given an optimal solution \(OPT\) of \(L(G)\) when \(t = 1\), \(\gamma(OPT) \leq w_{\text{max}}\) where \(w_{\text{max}}\) is the maximum node weight in a link-skein graph.

**Proof.** We notice the link-density of the optimal solution is less than or equal to \(w_{\text{max}}\).

\[
\gamma(OPT) = \frac{\sum_{e(u,v) \in OPT} w(u,v)}{|OPT|} \leq w_{\text{max}} \tag{3}
\]

\[
\sum_{e(u,v) \in OPT} w(u,v) \leq w_{\text{max}} \cdot |OPT| \tag{4}
\]

We can easily notice that \(\gamma(OPT) \leq w_{\text{max}}\) holds since \(w_{\text{max}} \cdot |OPT|\) is the maximum possible number of internal edges. It also indicates that given identified solution \(C\), \(\gamma(C) \leq \frac{w_{\text{max}}}{|OPT|} \cdot |OPT|\), as \(\gamma(C)\) is always positive. We notice that the approximation ratio depends on the link-density of our result.
we use the link-skein graph, we can mitigate the free-rider effect [40]. Our objective function (link-density) also considers that usage of the link-skein graph prunes relatively less patterns. However, when the similarity is very small such as 0.2, there are many edges in the link-space graph that rarely appear. This is because the link-skein graph has the effect of pruning inessential edges which appear in the link-space graph. Therefore, we can consider that usage of the link-skein graph prunes relatively less important edges and keeps the important edges.

2.3.2 Effectiveness. We first introduce the free-rider effect problem [40]. Let us denote $C_{OPT}$ as the optimal solution, whose goodness value $f(C_{OPT}) \geq f(C), \forall C \subseteq V$ for the maximization problem. Note that there are two types of free-rider effects: (1) the global free-rider effect and; (2) the local free-rider effect. In this section, we do not consider the local free-rider effect since it is defined for a community search problem with query nodes [40]. Thus, we only discuss the global free-rider effect. To check the global free-rider effect, we consider our problem as finding a single community (densely connected cohesive subgraph) without any query nodes.

Definition 6. Global free-rider effect [40]. A goodness function $f$ suffers from global free-rider effect if for any $C \subseteq V$, $f(C) \leq f(C \cup C_{OPT})$.

It is known that many metrics, including minimum degree, graph density, modularity, and external conductance measures suffer from the free-rider effect [40]. Our objective function (link-density) also suffers from the global free-rider effect, but we show that when we use the link-skein graph, we can mitigate the free-rider effect as opposed to when using the link-space graph. Let $f$ be the link-density in the link-space graph and $g$ be the link-density in the link-skein graph.

Theorem 3. The link-skein graph mitigates the global free-rider effect compared with the link-space graph.

Proof. Suppose that $C$ is a solution of $f$ and $g$, and $C_{OPT}$ as the optimal solution. From Definition 6, we can derive the following inequalities.

\[ \forall C \subseteq V, f(C) \leq f(C \cup C_{OPT}) \] \[
\forall C \subseteq V, g(C) \leq g(C \cup C_{OPT}) \]

Let us denote $f(C)$ as the link-density of the link-space graph, $g(C)$ as the link-density of the link-skein graph, and $x_C = f(C) - g(C)$. Note that $x_C$ is always positive. This is because the denominator is the same, but the numerator of $f(C)$ is larger than $g(C)$. Similarly, let us denote $x_{C_{OPT}}$ as $f(C_{OPT}) - g(C_{OPT})$, and $x_{C,C_{OPT}}$ as $f(C \cup C_{OPT}) - g(C \cup C_{OPT})$. $x_{C_{OPT}}$ is the link-density gain due to additional edges in the link-space graph, and $x_{C,C_{OPT}}$ is the link-density gain due to additional edges between $C$ and $C_{OPT}$.

We then check $f(C \cup C_{OPT}) - f(C)$ for a comparison with $g(C \cup C_{OPT}) - f(C)$.

\[ f(C \cup C_{OPT}) - f(C) \]
\[ \Leftrightarrow g(C \cup C_{OPT}) + x_{C,C_{OPT}} - (g(C) + x_C) \] \[ \Leftrightarrow g(C \cup C_{OPT}) + (x_{C,C_{OPT}} - x_C) - g(C) \]

From Equations 6 and 7, we derive the following inequality since $x_{C,C_{OPT}} > x_C$.

\[ f(C \cup C_{OPT}) - f(C) \geq g(C \cup C_{OPT}) - g(C) \]

Equation 8 implies that the link-space graph is more vulnerable with regards to the free-rider effects than the link-skein graph when we calculate link-density, as the link-space graph has additional edges with positive weights.

3 ALGORITHMS

In this section, we introduce how we generate the link-skein graph and propose two algorithms to solve the OCSM problem. Each algorithm has different strategies to solve the problem: (1) APA is an advanced peeling algorithm which is a top-down approach by iteratively deleting a set of nodes based on the link-density contribution. It first focuses on the degree constraints as the major concern and then aims to maximize link-density; (2) SEA is a seed-based expansion algorithm which is a bottom-up approach by iteratively adding a set of nodes based on the criteria. Its main concern is for maximizing the link-density, then try to satisfy the degree constraints by expanding a set of nodes.

3.1 Generating the link-skein graph

Algorithm 1 shows the procedure for generating the link-skein graph. It first calculates the similarity in the original graph to avoid duplicate computations, then assigns the weight in edges of the link-skein graph.

Time complexity. Time complexity to generate the link-skein graph is $O(|E||V|)$ since $|E|$ is for calculating the similarity, and $|V|$ is to find common neighbor nodes. Due to the power-law distribution of the degree in a graph, the practical running time is reasonably faster than the theoretical time complexity.
We first introduce the Peeling Algorithm (PA) which uses a straightforward approach. This algorithm is to use the k-core and minimum occurrence for finding a solution. Let denote a subgraph is feasible when the subgraph in G is connected and satisfies the minimum degree constraint, or a subgraph in L(G) satisfies the minimum occurrence constraint. In PA, k-core is used to find a maximal feasible solution in G. The high-level idea of PA is as follows. It firstly computes k-core Dk to find feasible subgraphs in G. It then converts Dk on the link-skein graph. Note that each subgraph in Dk might be divided into multiple connected components in L(Dk). For each connected component in L(Dk), we check whether the subgraph is feasible, i.e., the subgraph satisfies the minimum occurrence. If the subgraph is not feasible, we iteratively delete a set of nodes whose occurrence is less than k in a cascading manner. Finally, we pick the top t subgraphs as a result. We notice that PA may return large-sized subgraphs as a result since the peeling procedure is a kind of finding maximal feasible subgraphs in the link-skein graph.

To overcome the limitation of PA, we propose an advanced peeling algorithm (APA) by considering the link-density in the peeling procedure of PA. The procedure of the APA is described in Algorithm 2.

1. We firstly pick a feasible solution T1 from PA having the largest link-density (Line 4);
2. Next, for the selected connected component T1, we apply a density-based peeling strategy. We first pick a node u having the smallest average edge weight then delete it. Next, we apply an occurrence-based peeling approach to guarantee the minimum degree constraint. If the link-density is improved, we keep the result. This process is repeated until the connected component becomes empty (Lines 5-10);
3. Among the intermediate subgraphs T1, T2, . . . , Ti−1, we pick a subgraph which has the largest link-density when it is added to the current solution and adds it to the current solution (Lines 11-12). Next, we change the edge weight of the link-skein graph G (Line 13);

### Algorithm 1: Generating link-skein graph

**input** : $G = (V, E)$

**output** : Link-skein graph $L(G) = (V_{L(G)}, E_{L(G)})$

1. $H \leftarrow \emptyset$;
2. for $(e \in E)$ {
   3. $H$.addNode($e$);
   }
3. for $(e \in E)$ {
   4. $u \leftarrow$ from($e$), $v \leftarrow$ to($e$);
   5. $sim \leftarrow$ jaccard($N(u) \cup u, N(v) \cup v$);
   6. $W \leftarrow$ intersect($N(u), N(v)$);
   7. for $(w \in W)$ {
      8. $l_1 \leftarrow$ $H$.get($u, w$);
      9. $l_2 \leftarrow$ $H$.get($v, w$);
      10. $sim(l_1, l_2) \leftarrow$ $sim$;
   }
4. return $H$;

### Algorithm 2: Advanced Peeling algorithm (APA)

**input** : $G = (V, E)$, k, and t

**output** : $OCSM C \subseteq V$

1. $C \leftarrow \emptyset$;
2. while $|C| \neq t$ do
   3. $i \leftarrow 1$;
   4. $T_i \leftarrow L(PA(G, 1))$;
   5. while $|T_i| = 0$ do
      6. $v \leftarrow$ smallestAvgEdgeWeight($T_i$);
      7. $T_{i+1} \leftarrow T_i \setminus v$;
      8. $T_{i+1} \leftarrow T_{i+1} \setminus$ notSatisfying($T_{i+1}, k$);
      9. $i \leftarrow i + 1$;
   10. end
   11. $T^* \leftarrow$ pickBest($T_1, T_2, \ldots, T_{i-1}$);
   12. $C \leftarrow C \cup R(T^*)$;
   13. Change the edge weight in $L(G)$;
14. end
15. return $C$;

(4) Repeat steps 1 through 3 until finding the top t subgraphs.

**Example 6.** We utilize Figure 6 to explain the procedure of APA. Suppose that $t = 1$ and $k = 2$. APA firstly finds a solution of PA. There are two connected components in the link-skein graph. We choose the larger one since its link-density is larger than the smaller one. Next, for every node, we compute the average node weight. For example, the node weight of $v_{1,3}$ is 0.519. Since the node $v_{1,3}$ has the smallest node weight, we remove it. We then notice that the node 3 does not satisfy the minimum degree constraint. Thus, the node $v_{1,3}$ is deleted together. This process is repeated until there is no node in the current subgraph. Finally, we return a subgraph $v_{1,6}, v_{1,2}, v_{2,6}$ as a result since its link-density is larger than other intermediate subgraphs. Since $t = 1$, we do not need to update the edge weight. Whenever $t \geq 2$, it is required to update the edge weight based to avoid finding nested subgraphs.

**Limitation.** One issue is that after removing a node based on the average edge weights, a set of nodes can be deleted together cascadingly since the occurrence of some nodes can be decreased. This set of nodes is changed dynamically when we remove any node. Ideally, for every node, we can compute a set of nodes to be...
deleted together, then delete them which have the smallest link-density. However, this approach is prohibitive since computing all the sets in each iteration takes \(O(|V_1||E_1|)\), and it cannot be utilized to handle a large-scale dataset. Thus, in APA, we designed that the node deletion is done independently to improve the running time even if we might lose additional accuracy.

**Time complexity.** Time complexity of APA is as follows.

- \(O(|V| + |E|)\) to get k-core and a set of connected components
- \(O(|E_{L(G)}|)\) to apply the peeling approach for each iteration
- \(O(|V_{L(G)}|)\) is the maximum number of iterations
- \(O(|V||E|)\) is to compute the link-skein graph (See Algorithm 1)

Therefore, the time complexity of APA is \(O(|E_{L(G)}| + |V_{L(G)}| + |V||E|)\) since normally \(|E_{L(G)}| + |V_{L(G)}| > |V| + |E|\). Note that the time complexity of APA is the same as PA since the additional peeling step takes the same computational cost of the peeling approach in APA. Note that it does not take much time normally to apply the peeling approach.

### 3.3 Seed-Based Expansion Algorithm (SEA)

In this section, we introduce the Seed-based Expansion algorithm (SEA) which is a bottom-up manner. SEA algorithm uses expansion approaches by combining Goldberg’s densest subgraph algorithm [19] and a local expansion approach [13] with a reweighting scheme. Instead of finding a solution by iteratively removing a set of nodes, this algorithm aims to find the densest subgraph and then iteratively expand the solution while satisfying the constraint with two criteria. There are three main operations: (1) finding the densest subgraph in \(L(G)\); (2) applying local expansion; (3) reweighting. These operations are applied iteratively until finding top \(t\) subgraphs. The detailed explanation of each operation is as follows.

**Goldberg’s densest subgraph.** Goldberg [19] proposes a polynomial time algorithm to find the densest subgraph by using the max flow. Goldberg’s algorithm iteratively computes the minimum \(s-t\) cut based on the binary search procedure. One limitation of the Goldberg’s algorithm is that it can fail to find a solution in a large-scale dataset due to its computational cost. In our problem, we use Goldberg’s algorithm in the link-skein graph to find seed nodes.

**Local expansion.** In [13], authors propose two greedy strategies to find a community satisfying the minimum degree constraints from a seed node: (1) largest increment of goodness (lg). This approach is to choose a node having the largest \(\delta(G[C \cup v]) - \delta(G[C])\) in the expansion stage; (2) largest number of incidence (li). It chooses the node with the largest number of connections to the current node in the expansion stage, i.e., \(f(v) = deg_{G[C \cup v]}(v)\). We use both strategies in the local expansion process. Note that our operation is in the link-skein graph. Therefore, we use \(\beta(L(G)[C \cup v]) - \beta(L(G)[C])\) for lg and \(f(v) = deg_{L(G)[C \cup v]}(v)\) for li.

**Reweighting scheme.** Since we aim to find the top \(t\) subgraphs, it is required to have additional operations. Suppose that we have identified the top 1 subgraph. The simple way is to just remove the subgraph in the link-skein graph, then find other subgraphs. However, this approach has a flaw. Let assume that there are two cliques \(C_1\) and \(C_2\) which are overlapped partially, i.e., half nodes of each clique are overlapped. Suppose that we have identified \(C_1\) as the top 1 subgraph and have removed it. Then \(C_2\) may not be considered since some nodes in \(C_2\) are already removed. Thus, we change the edge weight of the selected subgraphs in the solution to 0 in the link-skein graph. It makes the Goldberg’s algorithm return meaningful results to find the top \(t\) subgraphs.

#### Algorithm 3: Seed-based Expansion Algorithm (SEA)

```
input : G = (V, E), k, and t
output : OCSM C ⊆ V_{L(G)}
1 C, T ← ∅;
2 T ← k-core(G);
3 LT ← L(T);
4 while |C| ≠ t do
5 S ← goldberg(LT);
6 S ← expansion(S, LT);
7 if S = ∅ then
8   next;
9 end
10 C.add(S);
11 reweighting(S, LT);
12 end
13 return C;
```

The pseudo description of SEA is described in Algorithm 3. Initially, we compute \(k\)-core and then convert the result of \(k\)-core to the subgraph of the link-skein graph (lines 2-3). Until finding the top \(t\) subgraphs, we firstly find the densest subgraph which can be the seed nodes (line 5). Next, we use the local expansion manner [13] to expand the seed nodes to guarantee the degree constraint (line 6). If we identify a subgraph satisfying the degree constraint, we add it to the solution and change the weight of the link-skein graph (lines 9-10). Finally, we return the resulted subgraph as a result.

**Example 7.** We reuse Figure 6 to explain the procedure of SEA. Suppose that \(t = 1\) and \(k = 2\). It firstly applies Goldberg’s densest subgraph algorithm to find an initial subgraph. When we apply the algorithm, it returns \(\{v_{1,6}, v_{1,2}, v_{2,6}\}\). Luckily, all the nodes satisfy the minimum degree constraint. Thus, we return the result directly.

**Time complexity.** Time complexity of SEA is as follows.

- \(O(|V_{L(G)}|^3)\) to compute Goldberg’s densest subgraph [19].
- \(O(|V||E|)\) is to compute the link-skein graph (See Algorithm 1).
- \(O(|X^*|)\) as the time complexity for local expansion. \(lg\) takes \(O(|V_{L(G)}|^3 + |E_{L(G)}| |log |V_{L(G)}|)\) and \(li\) takes \(O(|V_{L(G)}|^3 + |E_{L(G)}|)\)

Therefore, the time complexity of SEA is \(O(t(|V_{L(G)}|^3 + |X^*| + |V||E|))\).

### 4 EXPERIMENTS

In this section, we evaluate the proposed algorithms using real-world datasets. All experiments were conducted on Ubuntu 14.04 with a 32GB memory and 2.50GHz Xeon CPU E5-4627 v4. We used JgraphT library [31] in our implementation.
4.1 Experimental Setup

Dataset. Table 3 shows the statistics of 6 datasets in our experiments. All datasets are publicly available. We denote CI as the maximum core index, AD as the average degree, and $\# \Delta$ as the number of triangles.

Algorithms. To the best of our knowledge, our OCSM does not have direct competitors in previous literature due to the overlapping and minimum degree constraints. Thus, we compare the proposed algorithms with the several cohesive subgraph discovery problems including k-core, k-peak, k-truss, and (3, 4)-nucleus in our experiments. As we aim to find the top $k$ subgraphs, we use a greedy manner for post-processing. The list of the algorithms is as follows:

- k-core \[^{[36]}\]
- k-peak \[^{[20]}\]
- (k + 1)-truss \[^{[12]}\]
- (3, 4)-nucleus \[^{[35]}\]
- PA: Peeling Algorithm (in Algorithm 3.2)
- APA: Advanced Peeling Algorithm (in Algorithm 2)
- SEA: Seed-based Expansion Algorithm (in Algorithm 3)

Parameter setting. We use a different $k$ value based on the maximum core index. When $CI$ is less than 50, we vary $k$ between 3 and 6. To test the effect of $t$, we fix $k = 3$. When $CI$ is larger than 50, we vary $k$ between 5 and 8 and set $k = 3$ to test the effect of $t$. For selecting a proper $k$, we follow previous studies, which used the minimum degree threshold \[^{[16, 25]}\]. Finally, the link-density is chosen to measure the quality of the output subgraphs while the running time is used to measure the efficiency of our algorithms.

4.2 Experimental Results

Effect of $k$. Figure 7 shows the resultant link-density when we change the value of $k$. We observe that our SEA algorithm has the largest link-density for all cases. When the $k$ value increases, the gap between our algorithms and other algorithms decreases. This is because the input graph size becomes small. Thus, there is not sufficient room for improving the link-density. For an Amazon dataset, the (3, 4)-nucleus does not return a subgraph with $k \geq 6$. Thus, the (3, 4)-nucleus does not return a solution for $k = 6$. We can notice that the (3, 4)-nucleus achieves comparable results for some datasets. However, we point out that the (3, 4)-nucleus cannot return a result satisfying our desired $k$ minimum degree.

Effect of $t$. Figure 8 shows the resulting link-density when we change the value of $t$. From among the three algorithms, the proposed SEA algorithm is considered as the best algorithm. When $t$ becomes large, we observe that the link-density increases and that the gap between ours and the existing algorithms becomes larger.

Scalability test. To demonstrate the scalability of our algorithms, we vary the number of nodes between 1K and 1M in the LFR benchmark network \[^{[27]}\] using default parameters. For the scalability test, we fixed the number of the subgraph parameter $t$ to 20 and the minimum degree threshold $k$ to 3. Figure 9 shows the log-scaled running time of our algorithms and existing algorithms. For existing algorithms including k-core, k-truss, k-peak, and (3, 4)-nucleus, there are two parts: algorithm running time and post-processing time. Algorithm running time is the time necessary to compute cohesive subgraphs and post-processing time is the time necessary to find top $t$ subgraphs based on link-density in a greedy manner. We observe that the running times of the proposed PA and APA are not significantly different and are comparable with that of k-core, which is the fastest algorithm from among the existing algorithms. However, SEA takes much longer since it needs to find the densest subgraph to find seed nodes. However, we notice that our SEA is more scalable than k-peak and nucleus decomposition, as our algorithms do not require the enumeration of all the possible...
solutions. In $k$-peak or nucleus, a huge number of subgraphs are enumerated as a result. Thus, it naturally takes a long time to pick top $t$ subgraphs.

Table 4: Effectiveness test

| Algorithm | Exact solution | $k$-core | $(k+1)$-truss | $k$-peak |
|-----------|----------------|----------|---------------|----------|
| Link-density | 3.21            | 0.75     | 1.12          | 0.701    |
| Algorithm  | PA              | APA      | SEA(li)       | SEA(lg)  |
| Link-density | 1.67            | 2.37     | 2.56          | 2.91     |

**Effectiveness.** We use the Karate network [43], which contains 34 nodes and 78 edges to check the effectiveness of our algorithm. We first enumerate all the connected subgraphs [2], then filter them out if the minimum degree of a subgraph is smaller than or equal to 3. In total, there are 3, 431 connected subgraphs satisfying the minimum degree constraint. Next, we use a brute-force approach to find the top 4 subgraphs. In the experiment, we do not include the $(3,4)$-nucleus as it cannot be guaranteed to return any subgraph which has at least 3 minimum degree. Table 4 reports the results of the link-density. We notice that SEA with lg has a similar result as the exact solution and that APA also returns comparable results compared with the exact solution. In contrast, all competitors have low link-density.

**Case study (Polbooks).** Figure 10 shows a case study for the Polbooks network [33]. We set the parameters $k = 3$ and $t = 5$. Note that our result is translated back to the original graph for visualization. We observe that $k$-core and $k$-peak return a single giant connected component, which is loosely connected. This phenomenon occurs more frequently when the value of $k$ is small. We also observe that $k$-truss returns three connected components which are not sufficiently cohesively connected. $(3,4)$-nucleus returns 2 connected components which are densely connected components. We notice that our SEA returns only 5 clear cohesive connected components for which each subgraph satisfies the minimum degree constraint. Note that each connected component is densely connected and we can notice that several nodes overlap.

**Comparing with different measures.** Figure 11 reports on the four different measures of the resulting cohesive subgraphs for the Brightkite dataset. Note that a larger score indicates a better result for all measures. When we use the traditional graph density as an evaluation measure, we notice that the trend of the graph density is quite close to that of the link-density. However, as we have mentioned before, if our objective function is the traditional graph density, we cannot identify the overlapping structures. We also use graph modularity [32] to measure the quality of the identified cohesive subgraphs. We notice that our SEA algorithm returns the largest modularity. For 1-graph conductance, we observe that $k$-core and $k$-peak have large 1-conductance as they return isolated connected components as a result. We also check that $(k+1)$-truss has the lowest 1-conductance, which implies that the identified cohesive subgraph contains many external edges. In contrast, we use the link-skein graph, which helps consider external edges. Thus, our algorithms return high-quality dense cohesive subgraphs as a result.

**Approximation ratio.** Figure 12 shows the result of the approximation ratio of the proposed algorithm for $k = 5$ and $t = 1$. We compute the approximation ratio from Theorem 2. We notice that the approximation ratio of the SEA algorithm is more reasonable than those of other algorithms. The range of the approximation ratio of SEA is between 2 and 7.

5 RELATED WORK

**5.1 $k$-core and Its Variations**

$k$-core is widely used to find cohesive subgraphs. The definition of the $k$-core [36] is as follows: given a network $G = (V,E)$ and the positive integer $k$, the $k$-core of $G$, denoted by $H_k$, consists of a set of nodes of which all the nodes in $H_k$ have at least $k$ neighbor nodes in $H_k$. Batagelj et al. [6, 7] proposed an efficient $O(|E|)$ algorithm for finding the $k$-core. Sariyuce et al. [34] studied an incremental $k$-core problem in a dynamic graph. Instead of finding the whole $k$-core for every insertion and deletion in a dynamic graph, they proposed efficient algorithms to avoid duplicate operations.

$k$-truss [12] has recently been proposed for finding a cohesive subgraph. The definition is as follows: given a graph $G$ and the positive integer $k \geq 2$, the $k$-truss of $G$ is a maximal subgraph in which all edges are contained in at least $(k - 2)$ triangles within the subgraph. It is known that the time complexity of $k$-truss is $O(|E|^3)$. Even if $k$-truss returns more cohesive subgraph, it is hard to find an appropriate parameter $k$.

In [20], the authors claimed that when the graph contains multiple distinct regions with different edge densities, $k$-core cannot handle the sparser regions. To handle this problem, they formulated $k$-peak decomposition problem, which aims to find the centers of distinct regions in the graph. They proposed an efficient $k$-peak decomposition algorithm with a rigorous theoretical analysis.

There are several extensions of the $k$-core. Zhang et al. [45] proposed $(k,r)$-core for an attributed social network. They denoted a connected subgraph $S \subseteq V$ where $S$ is a $(k,r)$-core if $S$ satisfies both structure constraint ($\delta(S) \geq k$) and the similarity constraint ($DP(S) = 0$ where $DP(S)$ indicates that the number of dissimilar pairs in subgraph $S$). In the paper, they focused on two fundamental problems: enumerating all maximal $(k,r)$-cores and finding the maximum $(k,r)$-core. Recently, Bonchi et al. [9] introduced $(k, h)$-core which considers the $k$-core with graph distance. Given a distance threshold $h$ and the positive integer $k$, $(k, h)$-core of $G$ is a maximal subgraph such that every node in $(k, h)$-core has at least $k$ $h$-neighbors.

Bhawalkar et al. [8] propose the anchored $k$-core problem. The problem is to find a set of anchor nodes to maximize the size of the $k$-core. An anchor node indicates that a selected node, which does not belong to $k$-core, but is forced to belong to the $k$-core. They show that finding $b$ anchor nodes is NP-hard when $k \geq 3$. For a special case ($k = 2$), they proposed the exact algorithm, which has polynomial time complexity. Zhang et al. [44] proposed the practical OLAK (onion layer based anchored $k$-core) algorithm to solve the anchored $k$-core problem efficiently by reducing the search space significantly.

Sariyuce et al. [35] proposed the graph nucleus decomposition: given two positive integers $r < s$, the $(r,s)$-nucleus is defined as a maximal union of $s$-cliques, in which every $r$-clique is present...
5.2 Finding the Densest Subgraph

Finding the densest subgraph is one of the fundamental problems in the data mining field [38]. Given a graph $G = (V, E)$, the goal is to find the subgraph of $G$ which has the highest density. One of the popular density metrics is defined as $\frac{|E|}{|V|^2}$, where $|E|$ is the number of edges [22].

5.3 Line Graph Analysis

A node in real-world networks, especially in social networks, typically belongs to multiple communities, so communities overlap at a node. Overlapping community detection is known as a more generalized problem compared with disjoint community detection [29]. To tackle the overlapping community detection problem, an intuitive way is to identify a partition of links (i.e., relationships) rather than a partition of nodes (i.e., individuals). Evans and Lambiotte [15] and Ahn et al. [1] introduced the line graph model for this purpose. The line graph of the original graph is constructed as follows: each node is mapped from a link in the original graph that two nodes are adjacent if the corresponding links in the original graph share a common node. The link-space graph [29], which is a variant of the line graph, is used for the state-of-the-art algorithms dealing with the overlapping community detection problem. It is also a transformed graph of the original graph, whereby its topological structure is the line graph, and the weight is calculated on the original graph and carried over into the transformed graph. The weighting scheme helps avoid unnecessarily large communities containing weak ties.

6 CONCLUSION

In this paper, we propose Overlapping Cohesive Subgraphs with Minimum degree problem, which is called OCSM. We proved that the OCSM is NP-hard by showing a polynomial-time reduction. To solve the problem, we propose two efficient and effective heuristic algorithms called APA and SEA. APA is a top-down approach...
and the SEA is a bottom-up approach. Finally, we report on exten-
sive experiments using real-world datasets for demonstrating the
superiority of our algorithms.

REFERENCES

[1] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. 2010. Link communities
reveal multiscale complexity in networks. Nature 466, 7307 (2010), 761–764.
[2] Mohammed Aloshlhi, Saeed Salem, and Fidaa Abed. 2018. A Linear Delay
Linear Space Algorithm for Enumeration of All Connected Induced Subgraphs. In Proceedings of the 2018 ACM International Conference on Bioinformatics,
Computational Biology, and Health Informatics. 607–607.
[3] Wen Bai, Yadi Chen, and Di Wu. 2020. Efficient temporal core maintenance of
massive graphs. Information Sciences 513 (2020), 324–340.
[4] Oana Denisa Balalau, Francesco Bonchi, TH Hubert Chan, Francesco Gullo, and
Mauro Sozio. 2015. Finding subgraphs with maximum total duration and limited
overlap. In Proceedings of the Eighth ACM International Conference on Web Search
and Data Mining. 379–388.
[5] Jie Bao, Yu Zheng, and Mohamed F Mokbel. 2012. Location-based and preference-
aware recommendation using sparse geo-social networking data. In Proceedings of the 20th international conference on advances in geographic information systems.
Association for Computing Machinery, New York, NY, USA, 199–208.
[6] V. Batagelj and M. Zaversnik. 2003. An O(m) Algorithm for Cores Decomposition
of Networks. Advances in Data Analysis and Classification, 2011. Volume 5, Number 2, 129–145. arXiv:arXiv:cs/0310049
[7] V. Batagelj and M. Zaversnik. 2011. Fast Algorithms for Determining (Generalized)
Core Groups in Social Networks. Adv. Data Anal. Classif. 5, 2 (2011), 129–145.
[8] Kshipra Bhawalkar, Jon Kleinberg, Kevin Lewi, Tim Roughgarden, and Anesh Sharma. 2015. Preventing unravelling in social networks: the anchored k-core problem. SIAM Journal on Discrete Mathematics 29, 3 (2015), 1452–1475.
[9] Francesco Bonchi, Arjith Khan, and Lorenzo Severini. 2019. Distance-generalized
core decomposition. In Proceedings of the 2019 International Conference on Man-
agement of Data. 1006–1023.
[10] Moses Charikar. 2008. Greedy approximation algorithms for finding dense
components in a graph. In International Workshop on Approximation Algorithms for
Combinatorial Optimization. Springer, 84–95.
[11] Eunjoo Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility:
user movement in location-based social networks. In Proceedings of the 17th ACM
SIGKDD international conference on Knowledge discovery and data mining. ACM,
Association for Computing Machinery, New York, NY, USA, 1082–1090.
[12] Jonathan Cohen. 2008. Trusses: Cohesive subgraphs for social network analysis.
National security agency technical report 16 (2008), 3–1.
[13] Wanyun Cui, Yanghua Xiao, Haixun Wang, and Wei Wang. 2014. Local search of
communities in large graphs. In Proceedings of the 2014 ACM SIGMOD inter-
national conference on Management of data. ACM, Association for Computing
Machinery, New York, NY, USA, 1001–1002.
[14] Xiaoyu Ding, Haifu Yang, Jianpei Zhang, Jing Yang, and Xiaohong Xiang. 2022. CEO:
Identifying Overlapping Communities via Construction, Expansion and
Optimization. Information Sciences 596 (2022), 93–118.
[15] T.S. Evans and R. Lambiotte. 2009. Line graphs, link partitions, and overlapping
communities. Physical Review E 80, 1 (2009), 016105.
[16] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and
Xuemian Lin. 2020. A survey of community detection algorithms. Physical Review E 87, 4 (2018), 046110.
[17] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution:
Densification and shrinking diameters. ACM Transactions on Knowledge Discovery
from Data (TKDD) 1, 1 (2007), 2.
[18] Yuxun Liu, Sengwoog Ryu, Sejeong Kwon, Kyomin Jung, and Jae-Gil Lee. 2014. LinkSICAN*: Overlapping community detection using the link-space transfor-
mation. In 2014 IEEE 30th international conference on data engineering. IEEE,
292–303.
[19] Qingyuan Litng, Fan Zhang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. 2020. Global
Reinforcement of Social Networks: The Anchored Coreness Problem. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data.
Association for Computing Machinery, New York, NY, USA, 2211–2226.
[20] Dimitrios Michail, Jorí Jinkale, Barak Naveh, and John V. Sichi. 2020. JGraphT—A
Java Library for Graph Data Structures and Algorithms. ACM Transactions on
Evaluating Social Software (TOMS) 46, 2 (2020), 1–29.
[21] Mark EJ Newman. 2006. Modularity and community structure in networks.
Proceedings of the national academy of sciences 103, 23 (2006), 8577–8582.
[22] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with
Interactive Graph Analytics and Visualization. In Proceedings of the Twenty-Ninth
AAAI Conference on Artificial Intelligence (Austin, Texas) (AAAI’15). AAAI Press,
California, USA, 4292–4293.
[23] Ahmet Erdem Sanyurie, Bugra Gedik, Gabriela Jacques-Silva, Kim-Lung Wu, and
Unit V Cyatularev. 2016. Incremental k-core decomposition: algorithms and
evaluation. The VLDB Journal—The International Journal on Very Large Databases
25, 3 (2016), 425–447.
[24] Ahmet Erdem Sanyurie, C. Seshadhri, Ali Pinar, and Unit V Cyatularev. 2015. Finding the hierarchy of dense subgraphs using nucleus decompositions. In Proceedings of the 24th International Conference on World Wide Web. 927–937.
[25] Stephen B Seidman. 1983. Network structure and minimum degree. Social
networks 5, 3 (1983), 269–287.
[26] Charalampos Tsourakakis. 2015. The k-clique densest subgraph problem. In
Proceedings of the 24th international conference on world wide web. 1122–1132.
[27] Jiawen Wang, Rongjie Wang, Jia Wei, Qianli Ma, and Guihua Wen. 2020. Finding
dense subgraphs with maximum weighted triangle density. Information Sciences 539 (2020), 96–48.
[28] Ling-Yin Wei, Yu Zheng, and Wen-Chih Peng. 2012. Constructing popular routes
from uncertain trajectories. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. Association for Computing
Machinery, New York, NY, USA, 195–203.
[29] Yuhao Wu, Ruoming Jin, Jing Li, and Xiang Zhang. 2015. Robust local community
detection: on free rider effect and its elimination. Proceedings of the VLDB
Endowment 8, 7 (2015), 798–809.
[30] Junges Xie and Boledaw K Syrjanski. 2012. Towards linear time overlapping
community detection in social networks. In Pacific-Asia Conference on Knowledge
Discovery and Data Mining. Springer, 25–36.
[31] Jaewon Yang and Jure Leskovec. 2015. Defining and evaluating network commu-
nities based on ground-truth. Knowledge and Information Systems 42, 1 (2015)
181–213.
[32] Wayne W Zachary. 1977. An information flow model for conflict and fission in
small groups. Journal of anthropological research 33, 4 (1977), 452–473.
[33] Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2017. OLAK:
an efficient algorithm to prevent unravelling in social networks. Proceedings of the VLDB Endowment 10, 6 (2017), 649–660.
[34] Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2017. When
game meets similarity: efficient (k, r)-core computation on social networks. Proceedings of the VLDB Endowment 10, 10 (2017), 998–1009.
[35] Zhih Ren, Fanghua Ye, Rong-Hua Li, Gushui Lang, and Tan Jin. 2017. Finding
weighted k-truss communities in large networks. Information Sciences 417 (2017),
344–360.
[36] Jiming Zhu, Bilian Chen, and Yifeng Zeng. 2020. Community detection based on
modularity and k-plexes. Information Sciences 513 (2020), 127–142.

Search in Location-Based Social Networks. In Proceedings of the 2020 ACM SIG-
MOD International Conference on Management of Data. Association for Computing
Machinery, New York, NY, USA, 2199–2209.

Jungeun Kim, Sungsu Lim, Jae-Gil Lee, and Byung_Lee Lee. 2018. LinkBlackHole:
Robust Overlapping Community Detection Using Link Embedding. IEEE Trans-
actions on Knowledge and Data Engineering 31, 11 (2018), 2138–2150.

Brenda Lanchinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark
graphs for testing community detection algorithms. Physical review E, 78, 4 (2008),
046110.