Information Sciences 634 (2023) 290–307

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Effective and efficient core computation in signed networks

Junghoon Kim [a], Hyun Ji Jeong [b], Sungsu Lim [c,*], Jungeun Kim [d,*]

[a] Department of Computer Science and Engineering, Ulsan National Institute of Science & Technology, Ulsan 44191, South Korea
[b] Department of Artificial Intelligence, Kongju National University, Cheonan 31080, South Korea
[c] Department of Computer Science and Engineering, Chungnam National University, Daejeon 34134, South Korea
[d] Department of Software and CMPSI, Kongju National University, Cheonan 31080, South Korea

ARTICLE INFO

Keywords:
Cohesive subgraph discovery
Signed network analysis
Minimum degree constraint

ABSTRACT

With the proliferation of mobile technology and IT development, people can use social network services anywhere and anytime. Among many social network mining problems, identifying cohesive subgraphs has attracted extensive attention from different fields due to its numerous applications. The most widely used among many cohesive subgraph models is k-core, because of its simple and intuitive structure. In this paper, we formulate (p,n)-core in signed networks by extending the k-core model. (p,n)-core is designed to guarantee sufficient internal positive edges and deficient internal negative edges. We formally prove that finding an exact (p,n)-core is NP-hard. Hence, we propose three efficient and effective algorithms to find a solution. We demonstrate the superiority of our proposed algorithms using real-world and synthetic networks.

1. Introduction

With the recent rapid and prolific development of mobile and communication technology, people can use online social networking services at anywhere and anytime. Understanding online social networks helps us understand complex human relationships; and therefore, many researchers are trying to analyse social networks by capturing their characteristics [36]. There are several interesting properties in social networks including degree distribution [1], community structure [15], node influence [41], small diameter [43] and so on. Among them, mining of *cohesive subgraphs* has recently received much attention from different fields due to its attractive applications, even though there is no formal definition of a cohesive subgraph.

Nowadays, many social networks consist of attributes associated with node and link information. Such social networks with attribute information are called *attributed social networks* [8]. Attributed social networks include location-based social networks, keyword-based social networks, event-based social networks, multi-layer graphs, and uncertainty graphs, etc. In this study, we focus on a signed network [40] that consists of a set of nodes and edges, with each edge having either a positive sign ('+') or a negative ('-') sign. In social networks, this sign can be interpreted as the relationship between two users, where a positive edge indicates that two users (or entities) are close, while a negative edge implies that they are not in a good relationship. These signed networks have many applications [40,33,31] including social psychology, physics, correlation clustering, network robustness, and link recommendation.

In this paper, we investigate the problem of discovering cohesive subgraphs [37] in signed networks. Intuitively, we may consider applying the classical k-core algorithm to a signed network for this purpose. The k-core is a representative cohesive subgraph model and is widely used for social network analysis [29]. It is utilised for graph clustering, network visualisation, community search,

* Corresponding authors.
E-mail addresses: junghoon.kim@unist.ac.kr (J. Kim), hjjeong@kongju.ac.kr (H.J. Jeong), sungsu@cnu.ac.kr (S. Lim), jekim@kongju.ac.kr (J. Kim).

https://doi.org/10.1016/j.ins.2023.03.097
Received 19 March 2022; Received in revised form 12 March 2023; Accepted 17 March 2023

Available online 21 March 2023

and identifying influential nodes. There are many variations [29] of the k-core in different domains such as distance-generalised k-core [4], bipartite (α, β)-core [9], overlapping cohesive subgraph discovery [22] and so on. Recently, Giatsidis et al. [13] extended classic k-core for directed signed networks. However, the proposed problem only guarantees sufficient incoming internal positive edges and sufficient outgoing external negative edges, and does not consider negative internal edges in the resultant subgraphs. Hence, [13] cannot be directly utilised for some applications such as team formation or group recommendation because the identified subgraphs may contain many negative edges among users. This implies that the proposed method in [13] cannot guarantee the quality of the identified subgraph.

Instead of extending the k-core, several approaches [44,26] for finding cohesive subgraphs in signed networks have been proposed, including k-truss based [44] and clique based [26] approaches. Even if both approaches offer higher levels of cohesiveness compared to k-core [10], selecting the proper parameter can be challenging for end-users. For example, to determine an appropriate parameter k for a k-truss model, the end-users may need to understand the concept of the triangle and adjacent triangles. Similarly, the clique-based method also requires users to understand the model in order to select appropriate parameters. Conversely, our extended k-core based model is simple and intuitive, making it easier for end-users to select appropriate parameters.

In this paper, we propose a new core computation problem for signed networks named (p, n)-core by preserving the simple and intuitive structure of the k-core in networks.

The (p, n)-core inherits the structure of the traditional k-core. More specifically, (p, n)-core computation aims to find a maximal subgraph while guaranteeing sufficient internal positive edges and deficient internal negative edges of a resultant subgraph. Our proposed model is easier for end-users to select appropriate parameters compared to other approaches.

Our problem has many applications by directly extending the applications of the classic k-core such as *network analysis*, *community search*, *visualisation*, *similarity*, and *anomaly detection*.

We list several real-life motivating applications:

- **Community Search:** (p, n)-core problem can be utilised for solving the community search problem [42]. The community search problem aims to find query-centric community in which every node is densely connected to each other while containing all the query nodes.

  By finding a connected component of (p, n)-core that contains all query nodes in a signed network, we can expect to find a high-quality community since it guarantees sufficient positive cohesiveness and deficient negative cohesiveness. In Section 4, we present the result of the community search by utilising our proposed method.
- **Network Analysis:** (p, n)-core can be utilised in network analysis. For example, consider that we have a signed social network of computer science and engineering department. By varying constraints on positive and negative edges, we can identify high-level latent structure of students in the department. It can also be utilised to form teams or groups for various activities.
- **Recommendation System:** (p, n)-core can be utilised for recommendation system. Note that positive relationship in a social network implies that two users might have similar interests. Thus, identifying a cohesive subgraph can be directly utilised for the recommendation system, e.g., edge recommendation in a connected component of (p, n)-core.

In this work, we focus on the k-core problem in signed networks which involves unifying two key concepts. Each component can be described as follows:

1. *Considering positive and negative edge constraints*: To compute (p, n)-core, we simultaneously consider both positive and negative edge constraints. This approach ensures sufficient internal positive edges and deficient internal negative edges.
2. *Maximality*: We find a maximal cohesive subgraph as a result.

Note that this work is an extension of our previous study [21], in which we introduced (p, n)-core problem and presented two solutions. In this work, we have made three major changes: (1) we have provided additional explanations such as examples, pseudo descriptions, many figures, and motivating examples to improve understanding; (2) we present a new algorithm named FCA to significantly improve the efficiency for finding a solution; and (3) we have conducted extensive experiments to evaluate the performance of the three algorithms. We have included three additional large-sized real-world networks, and conducted scalability test. Additionally, we have studied the effect on the radius parameter, the effectiveness of the pruning strategy, and new case studies.

1.1. Challenges

Since finding an exact (p, n)-core is NP-hard (See Section 2), we have two primary challenges:

1. *Challenge 1: effectiveness*: how do we identify effective solutions?
2. *Challenge 2: efficiency*: how do we identify scalable (efficient) solutions?

To address both challenges, we propose three algorithms which are summarised in Table 1. Note that the term *"followers"* implies a set of nodes that would be deleted together due to the positive edge constraint when we remove a node.

A follower-based algorithm (FBA) is designed to compute the exact size of the followers after computing the candidate nodes. Even though FBA focuses on effectiveness, it fails to preserve effectiveness and efficiency since it does not consider the importance of the nodes in the negative graph and is required to compute all the followers of the nodes.

Table 1
Summary of proposed algorithms.

	FBA	DFBA	FCA
effectiveness	★	★★★★	★
efficiency	★	★★	★★★★

Next, we propose an improved FBA called a disgruntled follower-based algorithm (DFBA) to enhance both efficiency and effectiveness. For DFBA, we introduce two strategies: (1) *Propose pruning strategy*: to improve efficiency, We propose a new pruning strategy that avoid computing all follower sizes; and (2) *Set a new objective function*: to satisfy the negative edge constraint, We define a new objective function that considers both follower sizes and the node importance by selecting a proper node to be removed. Even if DFBA improves both effectiveness and efficiency, it still struggles with large-scale networks since it intrinsically computes many followers to pick the best node.

Lastly, to overcome the limitation of FBA and DFBA, we introduce a fast heuristic algorithm named FCA that empirically observes the relationship between the number of h-degree and the size of followers. By utilising an existing approximate neighbourhood estimation technique, we significantly reduce the running time without computing all followers for each iteration.

1.2. Contributions

The contributions of this work are summarised as follows:

- *Problem definition:* To the best of our knowledge, this is the first work for finding k-core in signed networks considering sufficient internal positive edges and deficient internal negative edges;
- *Theoretical analysis:* We prove that finding an exact solution of (p, n)-core problem is NP-hard and the solution is not unique;
- *Designing new algorithms:* We propose three algorithms to identify the high-quality cohesive subgraphs in signed networks;
- *Extensive experimental study:* We conduct extensive experiments on both real-world and synthetic networks to demonstrate the superiority of our proposed algorithms.

2. (p, n)-core in signed networks

A signed network is modelled as a graph $G = (V, E^+, E^-)$ with a set of nodes V, a set of positive edges E^+, and a set of negative edges E^-. We denote a positive graph (or a negative graph) G^+ (or G^-) to represent the induced subgraph consisting of positive (or negative) edges, and we denote V^+ (or V^-) to represent a set of nodes in the positive or negative graph. In this paper, we consider that the graph G is unweighted and undirected. Given a set of nodes $C \subseteq V$, we denote $G[C]$ as the induced subgraph of G which takes C as its node set and $E[C] = \{(u, v) \in E | u, v \in C\}$ as its edge set. Note that we allow any pair of nodes to have both positive and negative edges, as positive edges may indicate a public or visible relationship, while negative edges may indicate a private or invisible relationship. For example, in many online social networking services, users can choose to hide friends' posting or comments if the user explicitly click 'hide user' tab. This indicates that positive and negative edges may coexist in real-life networks. To introduce our problem, we present some basic definitions, which are explained in Table 2.

Table 2
Notations.

Notation	Description
$G = (V, E^+, E^-)$	a signed network
p	the positive edge threshold
n	the negative edge threshold
$H = (V_H, E_H^+, E_H^-)$	a subgraph H of G
H^+, H^-	a positive (or negative) graph H of G
$D(u)$	the disgruntlement of node u
$F(u)$	followers of node u
$\tilde{F}(u)$	an approximated number of followers of node u
$LB(u)$	the lower-bound of node u's followers
$UB(u)$	the upper-bound of node u's followers

Definition 1 *(Signed network).* A signed network $G = (V, E^+, E^-)$ consists of V nodes, a set of positive edges E^+ and a set of negative edges E^-.

Fig. 1 depicts an example of a signed network. Note that the blue-coloured edges indicate positive edges and the red-coloured edges indicate negative edges. This signed network contains 9 nodes, 12 positive edges, and 4 negative edges. To define (p, n)-core problem, we introduce two constraints.

Definition 2 *(Positive edge constraint).* Given a signed network $G = (V, E^+, E^-)$ and a positive integer p, a subgraph $H \subseteq V$ satisfies the positive edge constraint if any nodes in H have at least p positive edges in H, i.e., $\delta(H) \geq p$.

Definition 3 *(Negative edge constraint).* Given a signed network $G = (V, E^+, E^-)$ and integer n, a subgraph $H \subseteq V$ satisfies the negative edge constraint if any nodes in H have less than n negative edges in H, i.e., $\gamma(H) < n$.

Note that both edge constraints are utilised to define our problems. Next, we present k-core computation which is closely related to the positive edge constraint.

Problem definition 1 *(k-core [37]).* Given a graph $G = (V, E)$ and a positive integer k, k-core, denoted as H, is a maximal subgraph of which each node has at least k neighbours in H.

It is widely accepted that finding k-core in a network can be achieved in a polynomial time [2]. In the following, we use a notation p-core graph of a signed network, denoted by $H = G[D]$, where D is a p-core. Now, (p, n)-core problem is ready to be formally formulated.

Problem definition 2 *((p, n)-core).* Given a signed network G, positive integers p, and n, (p, n)-core, denoted as H, is a subgraph of G where every node satisfies the positive and negative edge constraints in H, i.e., we aim to find a subgraph H such that

- $|H|$ is maximised;
- $\delta(H) \geq p$;
- $\gamma(H) < n$

Note that the definition of (p, n)-core is an extension of k-core by additionally considering the negative edge constraint.

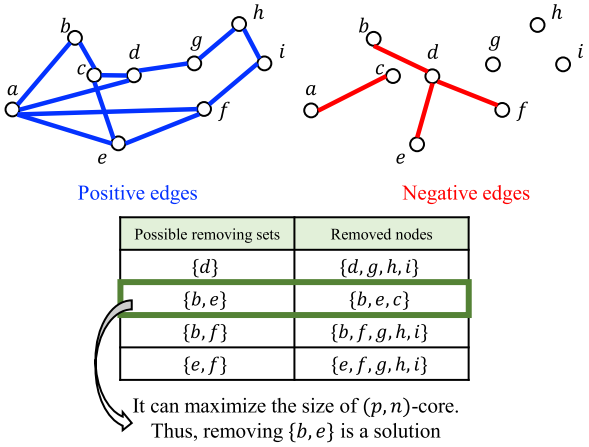

Fig. 1. Example of (p, n)-core.

Example 1. Fig. 1 shows an example of a signed network. Assuming that $p = 2$ and $n = 2$, the entire graph does not satisfy the negative edge constraint since node 'd' has three negative edges. To satisfy the negative edge constraint There are four options to satisfy the negative edge constraint. The first option is to remove node d, which leads to satisfy the constraint However, it results in the removal of nodes $\{d, g, h, i\}$ due to the positive edge constraint. Therefore, the remaining graph size is 5.

The second option is to remove nodes $\{b, e\}$. This results in the removal of node c due to the positive edge constraint; therefore, the size of the remaining graph is 6.

The third option is to remove nodes $\{b, f\}$. This leads to the cascading removal of nodes $\{b, f, g, i, h\}$, resulting in the remaining graph size of 4.

The last option is to remove nodes $\{e, f\}$. This results in the removal of nodes $\{b, f, g, i, h\}$ due to the positive edge constraint, resulting in a remaining graph size of 4. Since the objective is to maximise the size of (p, n)-core, the best option is to remove nodes $\{b, e\}$

2.1. Properties of (p, n)-core

Property 1. *(p, n)-core is not unique.*

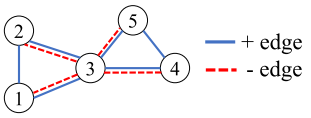

Fig. 2. Counter example.

Proof. In this proof, we present a simple counter example to demonstrate that (p,n)-core is not unique. Suppose that we have a simple graph shown in Fig. 2 with $p=2$ and $n=3$. Let D_1 as a set of nodes $\{1,2,3\}$ and D_2 as a set of nodes $\{3,4,5\}$. We know that two induced subgraphs $G[D_1]$ and $G[D_2]$ are feasible solutions. However, we notice that due to the negative edges of node 3, $G[D_1 \cup D_2]$ cannot be a feasible solution. As our goal is to maximise the number of nodes in (p,n)-core, either D_1 or D_2 can be a solution. □

Theorem 1. *Finding a solution of (p,n)-core is NP-hard.*

Proof. To prove the hardness of (p,n)-core computation, we utilise the well-known k-clique which is NP-hard. First, suppose that we have an instance of k-clique: $I_{KC}=(G=(V,E),k)$. We then show a reduction from I_{KC} to an instance of our problem. We first construct a new signed network $S=(V,E^+,E^-)$ where $E^+=E$, and we generate $|V|(|V|-1)/2$ negative edges in E^-, and then n and p are set to $k+1$ and $k-1$, respectively. It implies that the size of the solution must be larger than or equal to k, and less than $k+1$, i.e., the size must be k and its minimum degree is $k-1$. This indicates that finding a solution of $I_{pncore}=(S,k-1,k+1)$ is exactly same with finding a k-clique in I_{KC}. As mentioned above, a reduction from an instance I_{KC} to the instance I_{pncore} can be done in polynomial time. Therefore, we conclude that finding an exact solution to (p,n)-core is NP-hard. Fig. 3 shows a reduction from the k-clique problem to our (p,n)-core problem. □

Property 2. *If any solution of (p,n)-core where $p\geq 2$ and $n\geq 2$ contains a node u, any solution of $(p-1,n)$-core must contain the node u and any solution of $(p,n+1)$-core must contain the node u.*

Proof. One useful property of (p,n)-core is that any solution of this problem can also be a solution of both the $(p-1,n)$-core and the $(p,n+1)$-core. A node of a solution in (p,n)-core guarantees that it has at least p positive edges. This directly implies that it has at least $p-1$ positive edges. Similarly, a node of a solution in (p,n)-core indicates that it has less than n negative edge. Thus, the node belongs to $(p,n+1)$-core. □

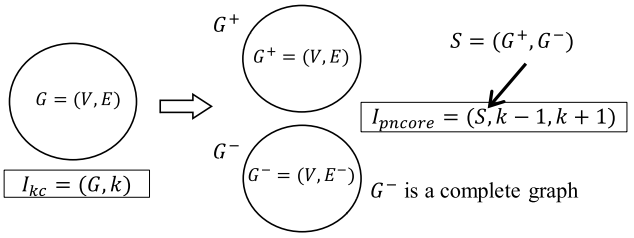

Fig. 3. Reduction procedure.

In this paper, we adopt a two-layered graph model to represent a signed network. As illustrated in Fig. 1, it comprises a positive graph layer and a negative graph layer, which facilitates a clear and concise visualisation.

3. Algorithms for (p,n)-core

In this section, we present three algorithms with different strategies for finding a solution of (p,n)-core. First, we propose a follower-based algorithm named FBA which prioritises a positive edge constraint as the primary concern. Next, we propose DFBA which puts both positive and negative edge constraints as a major concern. However, both approaches must compute a set of nodes which is removed together (named *followers*) for every iteration to find the best nodes to be removed. To address this, we propose a fast-circle algorithm, named FCA, by approximately estimating the size of the followers to select the best node to be removed. In this section, to avoid confusion, we interpret a signed network as a two-layered graph (the positive graph layer and negative graph layer can be seen in Fig. 1).

3.1. Follower-based algorithm (FBA)

We propose an algorithm named FBA which focuses on the positive edge constraint as the major concern. As our goal is to identify a set of nodes satisfying the constraints, it is reasonable to remove a set of nodes to satisfy the constraints 2. When a node is removed, a set of nodes can be removed together in a cascading manner due to the positive edge constraint (minimum degree constraint). This is because when a node is removed, the degree of the neighbour nodes decreases, which can make the nodes to not satisfy the positive edge constraint. The high-level idea of FBA is as follows: First, it runs p-core by considering positive edges only. This indicates that the current graph satisfies the positive edge constraint. If the current graph does not satisfy the negative edge constraint, we iteratively pick a node to satisfy the negative edge constraint while maximising the size of the remaining graph after removing its corresponding followers. This process is repeated until the remaining subgraph satisfies the positive and negative edge constraints. To formally define a set of nodes that can be removed together, we formulate a concept named *follower*.

Definition 4 *(Followers).* Given an integer p, positive graph H with $\delta(H) \geq p$, and node $v \in H$, followers of node v, denoted as, $F(v)$, consist of (1) node v and (2) the nodes that are deleted cascadingly owing to the positive edge constraint with p when node v is removed.

An example and properties are described as follows.

Example 2. In Fig. 1, suppose that $p = 2$. The followers of some nodes are as follows: $F(a) = \{a, b\}$, $F(b) = \{b\}$, $F(c) = \{b, c\}$, $F(d) = \{d, g, h, i\}$, $\cdots$, and $F(i) = \{g, h, i\}$. Note that followers of a specific node v contain the node v itself based on Definition 4.

Property 3. *Given a p-core graph H, when we remove a node $u \in H^+$ with $|F(u)| = 1$, the only node u in the negative graph is removed, i.e., no other nodes are additionally removed due to the negative edge constraint.*

Property 3 is reasonable since the negative edge constraint does not require any cascading removal.

Property 4. *When we remove a node u in the negative graph, a set of nodes which contains the node u can be removed together due to the positive edge constraint.*

Due to Property 3 and Property 4, removing multiple nodes is only determined in the positive graph. Thus, in FBA, it is necessary to compute the followers of the candidate nodes to identify the node to be removed. The strategy is as follows.

Strategy 1. *We identify a set of nodes in the negative graph, each of which each has at least n negative neighbour nodes. We refer to these nodes key nodes. Next, we find a set of candidate nodes to be removed. The candidate nodes are the union of the key nodes and neighbours of the key nodes. All the followers are computed for every candidate node, and then a node which has the minimum size of the followers is found. We then remove the selected node and its followers from the graph. This process is repeated until the remaining graph satisfies the positive and negative constraints simultaneously.*

In the following, we present a pseudo description of the FBA in Algorithm 1. The proposed FBA is to iteratively remove a set of nodes to satisfy the positive and negative edge constraints. It considers an union of key node and its neighbour nodes as candidate nodes. Then, it selects a node with the smallest number of followers since we aim to maximise the size of (p, n)-core. This process follows a greedy manner to find a solution. It is repeated until there is no node that violates the negative edge constraint.

Initially, p-core and a set of candidate nodes are computed (lines 1-3). Next, all the followers of the candidate nodes are identified, and then the node having the smallest followers is selected and then removed (lines 5-11). This process is repeated until the negative edge constraint of the remaining graph is satisfied. Finally, a set of nodes in the current subgraph is returned as a result (line 12).

Algorithm 1: Follower-Based Algorithm (FBA).

input : Signed graph G, and positive integers p and n
output: (p, n)-core

1 $H \leftarrow G[p\text{-core}(\underline{G})]$;
2 $H^+ \leftarrow p\text{Graph}(\underline{G})$ /* induced subgraph by positive edges */
3 $H^- \leftarrow n\text{Graph}(\underline{G})$ /* induced subgraph by negative edges */
4 **while** $\underline{\gamma(H^-) < n}$ **do**
5 $\quad S \leftarrow \bigcup_{v \in H^-} v$ if $N(v, H^-) \geq n$;
6 $\quad \mathsf{T} \leftarrow S \cup \bigcup_{s \in S} N(s, H^-)$;
7 $\quad F(v) \leftarrow \texttt{computeFollower}(\underline{v, H^+}), \forall v \in \mathsf{T}$;
8 $\quad u \leftarrow \text{argmin}_{v \in \mathsf{T}} |F(v)|$;
9 $\quad H^+ \leftarrow \texttt{removeNode}(\underline{H^+, F(u)})$;
10 $\quad H^- \leftarrow \texttt{removeNode}(\underline{H^-, F(u)})$;
11 $\quad T \leftarrow T \setminus F(u)$;
12 **return** V_H;

Complexity. Time complexity of the components in FBA is as follows.

- $O(|V|)$ is the number of iterations.
- $O(|V|+|E^+|)$ is to compute the follower of a specific node.
- $O(|V|)$ is total number of nodes.

Therefore, the time complexity of FBA is $O(|V|^2(|V|+|E^+|))$.

Limitation of FBA. Although FBA is a straightforward approach, it does not consider the negative edges when selecting the best node to be removed. It only utilises the negative edges to select the candidate nodes. As a result, it may remove many undesired nodes to satisfy the constraint, that is, it suffers from the effectiveness issue. In addition, for every iteration, it must compute all the followers in the remaining graph, which results in an efficiency issue.

3.2. Disgruntled follower-based algorithm (DFBA)

The previous section presents FBA and highlighted its limitations. This is because FBA only focuses on the followers of the identified candidates to find proper nodes to be removed and requires computing the exact followers in every iteration. In this section, we present DFBA to mitigate the limitation of FBA. The high-level idea of DFBA is to set a new objective function called *disgruntlement* which considers both negative and positive edges simultaneously, and propose the upper-bound and lower-bound by utilising the characteristics of p-core.

To incorporate the negative edge constraint, we define *disgruntlement* which is an indicator of how helpful it is to satisfy the negative edge constraint when we remove a node. Note that a node having a large disgruntlement is preferred to be removed since it helps satisfy the negative edge constraint. Therefore, in this section, we simultaneously consider both followers and disgruntlement to maximise the size of the resultant (p,n)-core. Since a large disgruntlement and a small number of followers are preferred, we aim at maximising the following function.

$$O(.) = \frac{D(.)}{|F(.)|} \tag{1}$$

In this section, we present an algorithm by iteratively removing a node which has the maximum $O(.)$. However, when computing Equation (1), a significant concern arises: *How can we efficiently compute* $|F(.)|$*?* Given a node v, computing $F(v)$ requires $O(|V|+|E^+|)$. We notice that computing $F(.)$ is required for all the nodes and every iteration. Note that the maximum iteration is $|V|$. To handle this issue, we develop a lower-bound (Section 3.2.2) and an upper-bound (Section 3.2.3). Finally, we present an algorithmic procedure (Section 3.2.4) and time complexity.

3.2.1. Considering negative edges

We provide a formal definition of the term 'disgruntlement' and an example of computing a disgruntlement is shown. Given a negative graph $G^- = (V^-, E^-)$ and a negative edge threshold n, a node $u \in V^-$ is denoted as a key node if its degree is larger than or equal to n.

Definition 5 *(Disgruntlement).* Given a negative graph G^-, a node $u \in V^-$, and a negative edge threshold n, the disgruntlement of node u is defined as

$$\begin{aligned} D(u) &= D^{self}(u) + D^{neib}(u) \\ D^{self}(u) &= \begin{cases} 0, & \text{if } u \text{ is not a key node,} \\ |N(u,G^-)| - n + 1, & \text{if } u \text{ is a key node} \end{cases} \\ D^{neib}(u) &= \sum_{w \in N(u,G^-)} 1, \forall N(w,G^-) \geq n \end{aligned} \tag{2}$$

Example 3. In Fig. 1, suppose that $n = 2$. We can compute the disgruntlement of the following nodes can be computed. For node a, node a is not a key node ($D^{self}(a) = 0$) and none of its neighbours is a key node ($D^{neib}(a) = 0$); thus, $D(a) = 0$. Similarly, $D(c) = 0$. Node b is known not to be a key node ($D^{self}(b) = 0$), but its neighbour node d is a key node; thus, $D^{neib}(b) = 1$. Since node d is a key node, we notice that $D^{self}(d) = 3 - 2 + 1 = 2$ and $D^{neib} = 0$; thus, $D(d) = 2$.

3.2.2. Computing a lower-bound

In this section, we introduce an approach to find a lower-bound of the followers. Let us recall the definition of the followers (Definition 4). When a node in the positive graph is removed, the followers indicate a union of the node and a set of nodes that is removed together due to the positive edge constraint.

We focus on finding a lower-bound for a node $v \in V^+$, denoted as $LB(v)$, which is always smaller than $|F(v)|$.

To identify the lower-bound of the followers, we first introduce the concept of a VD-node (Verge of Death).

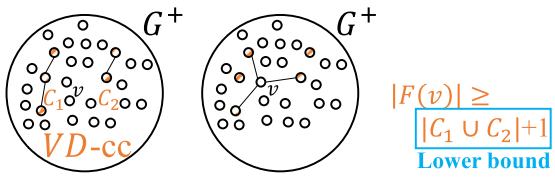

Fig. 4. Computing a lower-bound of node v.

Definition 6 *(VD-node).* Given a graph G, and an integer p, a node $v \in V$ is named a VD-node if its degree and coreness[1] is exactly p.

Note that a VD-node belongs to the p-core and its degree is exactly p. The VD-node represents a critical node in the positive graph, implying that the node will be immediately deleted after any neighbour nodes of the VD-node are removed. This is because the degree of the VD-node is exactly p, making it vulnerable to the removal of any of its neighbours.

Definition 7 *(VD-cc).* Given a graph G, and a set of VD-nodes, a set of connected components of VD-nodes are denoted as VD-ccs.

If any node in a VD-cc is removed, all the nodes in the "*associated connected components*" are removed together. Therefore, we can find the lower-bound of the followers by performing the following steps. First, all the VD-nodes ($O(|E^+|)$) are computed and VD-ccs ($O(|V|+|E^+|)$) are found. Next, given a node $v \in V$, $N(v, G^+)$ can be computed. If any neighbour node $w \in N(v, G)$ is in VD-nodes, the VD-cc of the node w can be obtained. After combining all the VD-ccs of the neighbour nodes, the minimum size of the followers of the node v can be computed by summarising all the sizes of VD-ccs. An advantage of using VD-ccs is that to find the lower-bounds of all the nodes, repeatedly finding VD-ccs is not required. Figure 4 presents an example to compute a lower bound.

3.2.3. Computing an upper-bound

In this section, we present an approach for finding an upper-bound of the number of followers by leveraging the hierarchical structure of the k-core [37]. To describe the idea of identifying the upper-bound, we first need to introduce some definitions.

Definition 8 *(CCNode).* Given a positive graph G^+, and integer k, we define a set of connected subgraphs of induced subgraphs by the k-core as a set of CCNodes, i.e., each connected subgraph of k-core is a CCNode.

Given an integer k, we observe that any pair of two CCNodes is not connected and the number of maximal CCNodes is $\frac{|V^+|}{(k+1)}$. This is because each CCNodes must contain at least $k+1$ nodes.

By utilising the CCNode, we define CCTree.

Definition 9 *(CCTree).* CCTree is a tree consisting of a dummy root node and a set of CCNodes. CCNodes in the same tree level imply that they belong to the same x-core, and a parent and a children pair of the CCNodes which are connected in the CCTree implies that the parent CCNode (in x'-core) contains the children CCNode (in $(x'+1)$-core).

The height of the CCTree is the $c^{max} - p + 1$ where c^{max} is the maximum coreness in the positive graph. Identifying the upper-bound is described as follows.

Strategy 2. *A CCTree is first constructed based on the input graph G^+ and positive edge threshold p by making a dummy node and adding it to the CCTree as a root node. Next, p-core is computed and all the connected components induced by p-core are found. The connected subgraphs can be the CCNodes and they can be the children of the root node with level 1 of the CCTree. Then, for each CCNode C_i, we check whether C_i contains p'-core with $p' = p + l$ where l is the level of the C_i. For example, if a CCNode is in level 2, it indicates that the CCNode is a subgraph induced by $(p+1)$-core. Thus, we check whether $(p+2)$-core exists in the nodes of the CCNode. If p'-core exists, a new CCNode C_j is added by computing the connected components of p'-core. Then, C_i is made to be the parent of C_j since $C_j \subseteq C_i$. This process is repeated.*

To compute the upper-bound for a given node v, we need to compute two CCNodes named cur' and cur. A CCNode cur' is a child of the root in the CCTree which contains the node v. A CCNode cur which contains the node v while the level of the cur in the CCTree is the largest among the CCNodes which contain the node v. The children immutable size is then computed. As the name implies, the size of the node sets is found which are not deleted after removing the node v since they are already sufficiently connected (children immutable).

Children immutable size $CI(.)$**.** *Given cur and node v, the children immutable size is the sum of the children's size of cur. Since the children of cur do not contain the node v, and their coreness value is larger than coreness of node v, i.e., the nodes in the children of the cur are already sufficiently connected. Thus, they will not be removed after removing the node v.*

[1] Given a graph G and a node v, the coreness of the node v is q if the node v belongs to the q-core but not to $(q+1)$-core.

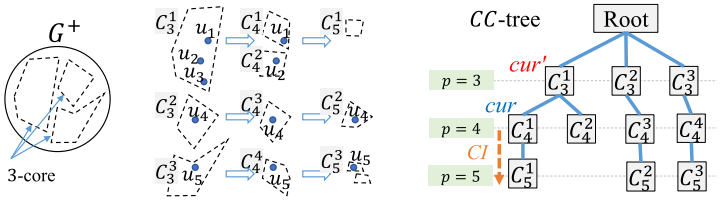

Fig. 5. Computing an upper-bound.

By negating $CI(.)$ from $|cur'|$, we can find upper bound.

$$UB(u) = |cur'| - CI(u) \tag{3}$$

Example 4. Fig. 5 shows a CCTree and CCNodes when $p = 3$. We assume that the input graph is G^+. Let find an upper-bound of the followers of the node u_1. We notice that the cur' is C_3^1, and cur is C_4^1 since C_4^1 is deeply located than C_3^1. We then compute $CI(u_1)$ by summarising the children's size. Hence, $CI(u_1) = |C_5^1|$. Therefore, the upper-bound of the node u_1 is $|C_3^1| - |C_5^1|$.

Algorithm 2: Disgruntled follower-based algorithm (DFBA).

input : Signed graph G, and positive integers p and n
output: (p,n)-core

$H \leftarrow G[p\text{-core}(\underline{G})]$;
$H^+ \leftarrow$ pGraph$(\underline{G})$ /* induced subgraph by positive edges */
$H^- \leftarrow$ nGraph$(\underline{G})$ /* induced subgraph by negative edges */
$c[v] \leftarrow$ computeCoreness$(\underline{H^+}), \forall v \in V_H$;
$ct \leftarrow$ constructCoreTree();
$S \leftarrow \bigcup_{v \in H^-} v$ if $N(v, H^-) \geq n$;
$\mathsf{T} \leftarrow S \cup \bigcup_{s \in S} N(s, H^-)$;
while $\underline{\gamma(H^-) < n}$ **do**
  $D(v) \leftarrow$ computeD$(\underline{v, H^-}), \forall v \in T$ and $D(v) \neq 0$;
  $M \leftarrow$ computingVDcc$(\underline{H^+, c[]})$;
  $LB^* \leftarrow$ computeLowerBound$(\underline{v, ct})$;
  $UB^* \leftarrow$ computeUpperBound$(\underline{v, M})$;
  $Cand \leftarrow$ computeCandidates$(\underline{LB^*, UB^*})$;
  $u \leftarrow \text{argmax}_{v \in Cand} O(v)$;
  $F(u) \leftarrow$ computeFollower$(\underline{u, H^+})$;
  c.remove$(\underline{c, F(u)})$;
  $H^+ \leftarrow$ removeNode$(\underline{H^+, F(u)})$;
  $H^- \leftarrow$ removeNode$(\underline{H^-, F(u)})$;
  $T \leftarrow T \setminus F(u)$;
  updateCoreness$(\underline{c})$ /* apply Purecore [35] */
  updateCT$(\underline{ct, F(u)})$;
return V_H;

3.2.4. Algorithmic procedure

We introduce our proposed method named DFBA, which combines the lower-bound and upper-bound with the concept of disgruntlement. First, we discuss overall strategy and the objective function of DFBA.

Strategy 3. *Instead of computing all the nodes' followers, we aim to compute only a few followers for efficiency. We first compute the following three values: (1) Disgruntlement score $D(.)$; (2) lower-bound $LB(.)$; and (3) upper-bound $UB(.)$. By utilising disgruntlement and the $LB(.)$ and $UB(.)$, we can set the new lower-bound and upper-bound. Note that the following equation always holds.*

$$\begin{aligned} LB(.) \leq |F(.)| \leq UB(.) \Rightarrow \frac{D(.)}{LB(.)} \geq \frac{D(.)}{|F(.)|} \geq \frac{D(.)}{UB(.)} \\ \Rightarrow UB^*(.) \geq O(.) \geq LB^*(.) \end{aligned} \tag{4}$$

In our algorithm, we utilise Equation (4) in the pruning strategy. If the lower-bound of node v is greater than the upper-bound of node u, then computing the followers of node u is unnecessary. Therefore, we first compute the node having the largest $LB^*(.)$ and then find a set of candidate nodes to compute the followers. The detailed procedure of DFBA is described in Algorithm 2.

Complexity. Time complexity of the components in DFBA is as follows.

- $O(|V|)$ is the number of iterations.

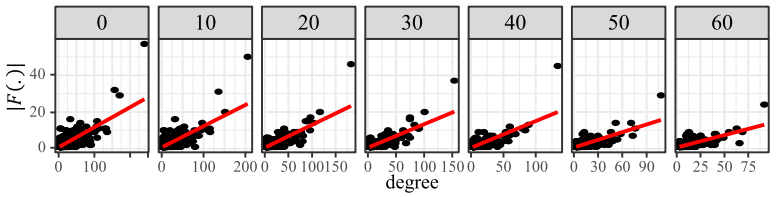

Fig. 6. Node degree and the size of followers.

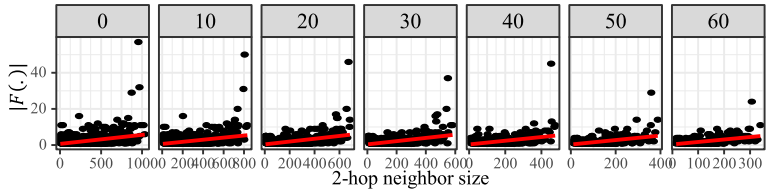

Fig. 7. Two-hop degree and the size of followers.

- $O(|V|(|V|+|E^+|))$ is to compute all the followers of the nodes V.
- $O(|E^-|)$ is to compute disgruntlement of the nodes.

Since for every iteration, we need to compute the followers of the remaining nodes, it takes $O(|V|^2(|V|+|E^+|)+|E^-|)$.

3.3. Fast-circle algorithm (FCA)

In this section, we propose a novel and efficient heuristic algorithm by utilising HyperANF technique. The high-level idea is as follows. By observing the real-world datasets, we observe that the size of followers is correlated to the x-hop node degree. Thus, we use HyperANF to find the best node to be removed based on its approximated value. After removing process, we update the approximated values to select the next nodes to be removed. This process is repeated until the remaining graph satisfies the positive and negative edge constraints.

3.3.1. Motivation

In Sections 3.1 and 3.2, we compute the exact followers to determine the best node to be removed. In this section, we propose a new approach that does not require computing the exact followers of all candidate nodes. To estimate the size of the followers, the characteristics of the followers in a signed network are utilised. Upon observation, we find that the degree of nodes correlates with the size of their followers, as depicted in Fig. 6. When the regression line (red-coloured) to fit the data distribution is plotted, we observe that the degree and the size of followers are correlated. Thus, selecting a node having a large degree tend to a large number of followers. Fig. 7 shows the correlation between the two hop degree of nodes and the size of followers, which is observed to still have a correlation. This observation indicates that when a node having a small degree is selected, it may have few followers. The challenge of this approach is in *efficiently computing the r-hop degree*. To handle the challenge, we utilise the HyperANF technique.

3.3.2. HyperANF

HyperANF [3] is a state-of-the-art algorithm for computing the approximated neighbourhood cardinality of a network. It utilises the HyperLogLog counter [11] which is a statistical counter requiring $O(\log\log n)$ bits. Therefore, HyperANF can efficiently estimate the number of reachable nodes within a specific distance from a node. Although the time complexity of the HyperANF is not revealed; however, its running time is expected to be approximately $O(r(|V|+|E|))$, where r denotes a maximum distance since HyperANF is a kind of extension of ANF [34] requiring $O((|V|+|E|)h)$.

3.3.3. Algorithmic procedure

Strategy 4. *The strategy of FCA is to use the approximated objective function in Equation (1) to select the best node to be removed by not computing the exact value $F(.)$ to find the best node, the estimated number of followers $\tilde{F}(.)$ is computed. To estimate the number of followers, we utilise HyperANF [3]. After selecting the best node, the exact followers of the node are computed. Next, $\tilde{F}(.)$ values of the neighbour nodes of the selected node are updated by considering the distance r.*

Algorithm 3 provides a pseudo description of FCA. Since it requires computing a follower for each iteration, its time complexity is much better than other algorithms.

Time complexity. Let denote H as a time complexity for HyperANF. Then, the time complexity of our FCA is as follows.

Algorithm 3: Fast Circle Algorithm (FCA).

input : Signed graph G, and positive integers p, n, and r
output: (p,n)-core

$H \leftarrow G[\texttt{pcore}(\underline{G})]$;
$\forall v \in H$, compute $D(v)$;
$\forall v \in H$, compute $\tilde{F}(v)$;
$\forall v \in H$, compute $O^*(v) = \frac{D(v)}{|\tilde{F}(v)|}$;
while $\underline{\gamma(H^-) < n}$ **do**
  $u \leftarrow \texttt{getBest}(O^*)$;
  $F(u) \leftarrow \texttt{follower}(\underline{u, H^+})$;
  update O^* and $\tilde{F}(.)$;
  $H^+ \leftarrow \texttt{removeNode}(\underline{H^+, F(u)})$;
  $H^- \leftarrow \texttt{removeNode}(\underline{H^-, F(u)})$;
return V_H;

- Maximum number of iterations is $O(|V|)$.
- Computing HyperANF takes $O(H)$.[2]
- Computing followers takes $O(|V| + |E|)$.
- Dijkstra's shortest path algorithm takes $O(|V| + |E| \log |V|)$.

Hence, the time complexity of FCA algorithm is $O(|V|(|V| + |E| \log |V| + |V| + |E|) + H)$.

4. Experiments

In this section, we evaluate the performance of our algorithms using eight real-world networks and three synthetic networks. We conducted all experiments on a machine running Ubuntu 18.04 with 128 GB memory and a 2.50 GHz Xeon CPU E5-4627 v4. For implementation, we implemented our algorithms using the JgraphT library [32].

Experiment setup. In our experimental section, we design experiments to answer the following questions.

- **Effectiveness**: How do our algorithms work on real-world / synthetic networks?
- **Efficiency of the pruning strategy**: How much does the pruning strategy improve efficiency?
- **Sensitivity of the negative edge ratio**: What trend does the cohesive subgraph return when the negative edge ratio changes?
- **Scalability**: How do our algorithms scale with the graph size?
- **Sensitivity of radius r of FCA**: How does changing radius r affect the result?
- **Case study**: Does (p,n)-core return meaningful results?

Table 3
Real-world datasets.

Dataset	$\|V\|$	$\|+\|$	$\|-\|$	MC	# Δ
Alpha [23]	3,783	12,759	1,365	18	16,838
OTC [23]	5,881	18,250	3,242	19	23,019
Epinions (EP) [25]	131,828	590,466	120,744	120	3,960,165
SD0211 [25]	82,140	382,167	118,314	54	418,832
SD0216 [25]	81,867	380,078	117,594	54	414,903
SD1106 [25]	77,350	354,073	114,481	53	395,289
Wiki-edit (WE) [5]	116,836	774,785	1,253,086	88	4,450,048
Wiki-interaction (WI) [30]	138,587	629,523	86,360	53	2,599,698

Dataset. Table 3 reports the statistics of the eight real-world networks. In our work, we do not consider the weighted temporal networks; thus, we convert Alpha and OTC datasets [23] as signed networks by ignoring the edge weights, and keeping the most recent edges with their signs. All the datasets used in this paper are publicly available. In Table 3, $|V|$ is the number of nodes, $|+|$ (or $|-|$) is the number of positive (or negative) edges, MC is the maximum positive coreness value of a network, and # Δ is the number of triangles in a positive graph.

Since the real-world networks we analysed are relatively small in size, we synthetically generate signed networks by utilising existing non-signed networks [45]. Specifically, we randomly insert $2|E|$ negative edges into original networks while considering the original edges as positive edges. The statistics for the synthetic networks can be found in Table 4.

[2] Unfortunately, the time complexity of the HyperANF seems not revealed [20], but its time complexity is around $O((|V| + |E|)h)$ where h is a maximum distance because it is an extension of ANF [34] that takes $O((|V| + |E|)h)$.

Table 4
Synthetic datasets.

Dataset	$\|V\|$	$\|+\|$	$\|-\|$	MC	# Δ
Amazon [45]	334,863	925,872	1,851,744	6	667,108
DBLP [45]	317,080	1,049,866	2,099,732	113	2,224,650
Youtube [45]	1,134,890	2,987,624	5,975,248	51	3,056,379

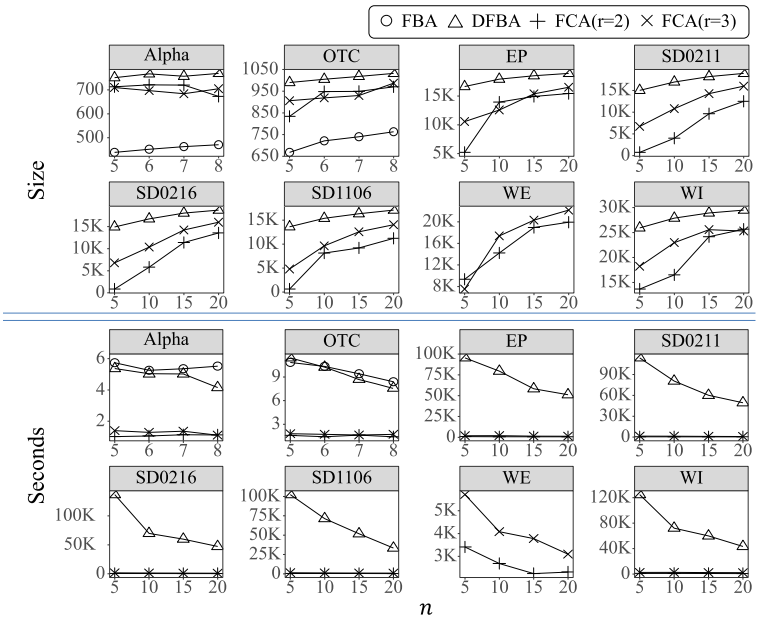

Fig. 8. Varying the variable n.

Algorithms. To the best of our knowledge, our (p,n)-core does not have any direct competitor in the previous literature. Prior works have focused on the different objective functions with different problems, directly comparing the size of the resultant subgraph is unfair. Thus, we report the results of our proposed algorithms in these experiments. Since both FBA and DFBA are not scalable due to their high time complexity, we only report the results of both algorithms if the algorithms are terminated within 24 hours. FBA is only applicable to OTC and Alpha datasets, and DFBA cannot be applicable in WE.

4.1. Experiments on real-world networks

In this section, we observe the impact of varying the user parameters p and n on the size and running time of proposed algorithms. We expect that as p increases, the size of the result will become smaller. Similarly, as n increases, the size of the result will be larger.

Effect on n. In Fig. 8, we fix the parameter $p=5$ then change the negative edge threshold n from 5 to 20 (for Alpha and OTC datasets, n is varied from 5 to 8 since the size of both networks is small).

When n becomes larger, our algorithms return larger (p,n)-cores since the large n indicates that we allow more negative edges in (p,n)-core. In addition, we observe that when n becomes larger, the algorithm will be finished earlier since we do not need to remove many nodes in the removing procedure.

For all the cases, we observed that DFBA consistently returns the best result. We verified that FBA returns small-sized (p,n)-core since it does not consider the negative edges during node removal. We verified that both FCAs return reasonable solutions, but their effectiveness is limited compared with DFBA since it uses the approximations of the follower size. We observe that FCA($r=3$) normally returns better results compared with FCA($r=2$) since it considers much more structural information.

Effect on p. In Fig. 9, we fix the parameter n at 5 and vary the positive edge threshold p from 5 to 20, except for the Alpha and OTC datasets, where we change p from 5 to 8. We identified that when p is large, the three proposed algorithms consistently return small-sized (p,n)-cores as a result since the larger p implies that the resultant subgraphs are more cohesive through the positive edges. We also observed that when p becomes larger, the algorithms are terminated earlier since p-core returns many small-sized results, i.e., the nodes to be removed are comparably limited.

These results are consistent with the trends observed in Fig. 8. Specifically, we observe that DFBA returns the best result and FCA($r=3$) returns comparable results. Additionally, the FCA algorithms with different radius parameters are much faster than other algorithms since it does not compute the exact number of followers for every iteration.

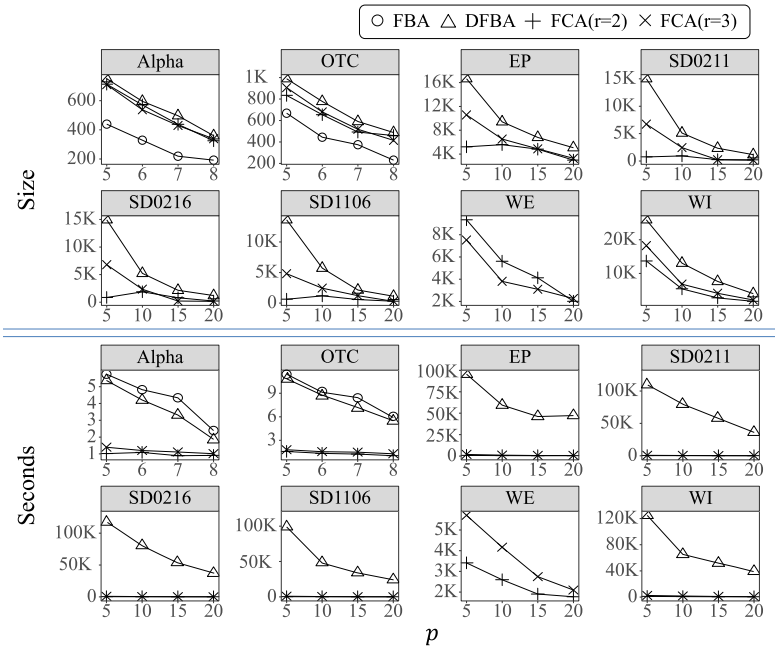

Fig. 9. Varying the variable p (same legend with Fig. 8).

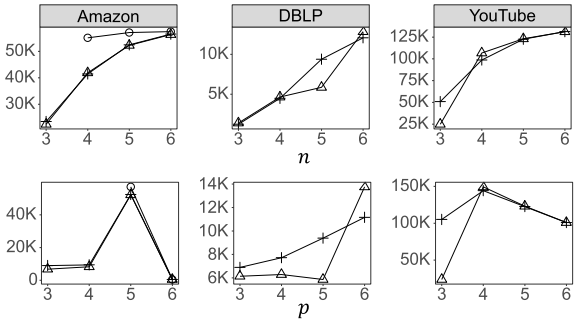

Fig. 10. Varying the variables p and n.

4.2. Experiments on synthetic networks

In this section, we conduct the same experiments on synthetic networks by varying parameters. We expect that as p increases, the size of the result will become smaller. Similarly, as n increases, the size of the result will be larger.

In Fig. 10, we present two set of experimental results obtained by varying p and n. For each parameter, we vary one from 3 to 6 while the other is fixed at 5 to observe the trends in synthetic networks. We found that the trends in the results are consistent with those observed in the experiments on real-world networks. Specifically, as the p value becomes larger, the resultant subgraph becomes smaller due to the stricter constraint, and as the n value becomes larger, the resultant subgraph becomes larger due to the relaxed constraint. Note that in some cases shown in Fig. 10, the resultant subgraph size becomes larger as p becomes larger. This is because removing several nodes is beneficial in finding large-sized subgraphs since the negative edges are injected randomly ($p = 5$ in Amazon, $p = 6$ in DBLP, and $p = 4$ in Youtube). However, if p-core returns a very small-sized solution, it may return a very small-sized result ($p = 6$ in Amazon).

4.3. Experiments on the pruning strategy

In this section, we first present the efficiency of our pruning strategy. To check the efficiency, we check the number of iterations and the ratio of nodes required to compute followers. We expect that our DFBA has fewer iterations than FBA and has fewer nodes which are required to compute followers. Next, we check the statistics of real-world networks to rationalise our pruning strategy in DFBA. We mainly check the number of connected components and the ratio of the number of nodes.

Analysis of the pruning strategy. Fig. 11 shows the number of connected components (in blue) and the ratio of the nodes (in red) when varying the parameter p from 1 to the maximum coreness value in real-world networks. We observe that there is a single large

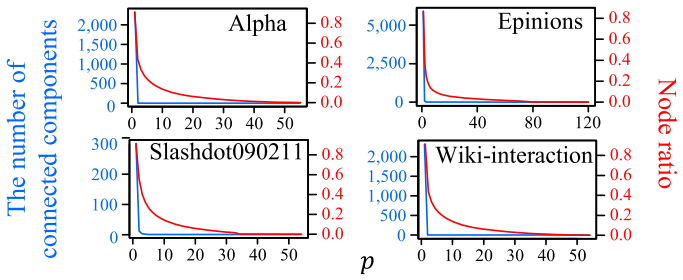

Fig. 11. # of connected components and the node ratio.

connected component containing all the nodes when p is larger than or equal to 3. We also observe that the difference between any adjacent p values is not significant. That implies that the children immutable size (CI) may be helpful to find the proper upper-bound to improve efficiency.

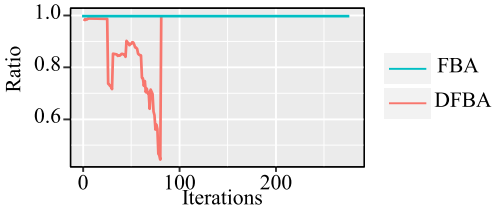

Fig. 12. Ratio of computing followers in Alpha and OTC networks ($p = 5, n = 5$).

Effect on pruning strategy. We also verified that our proposed DFBA approach can avoid computing followers in OTC networks. Fig. 12 reports the ratio of the nodes to compute followers among all the remaining nodes in DFBA and FBA. The DFBA is verified to have fewer iterations than FBA. More specifically, in OTC network, DFBA has less than 30% iterations compared with the number of iterations of FBA. We also observed that DFBA compute fewer followers compared with FBA since the pruning strategy can affect the avoidance of computing all the followers.

4.4. Effect on the negative edge ratio

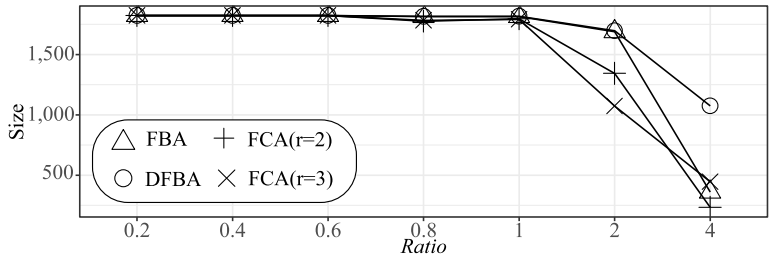

Fig. 13. Varying the ratio.

In this section, we vary the negative edge ratio to study its effect on the size of (p, n)-core. We expect that when there are many negative edges, finding large-sized (p, n)-core becomes hard.

By using the LFR benchmark dataset [24], we varied the number of injected negative edges. The value $\alpha = \frac{|-|}{|+|}$ is varied from 0.2 to 4.0 where $|+|$ is the number of positive edges and $|-|$ is the number of negative edges. In Fig. 13, we observe that when the ratio is smaller than 1, it has no significant difference in the size of the identified solution. However, when the values are larger than or equal to 2, the size of the identified solution is significantly decreased. When $\alpha = 4$, the FBA contains only 19.7% of the nodes compared with $\alpha = 0.2$.

4.5. Scalability test

In this section, by varying the size of synthetic networks, we check the running time of our proposed algorithms. We can expect that our proposed FCA may return the best result.

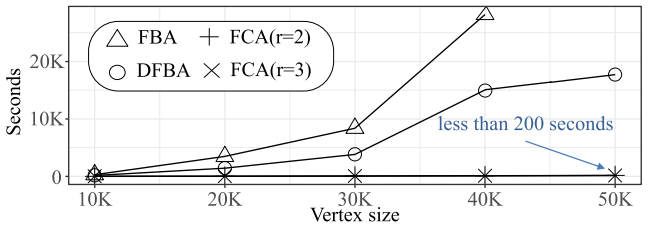

Fig. 14. Scalability test ($p = 3, n = 3$).

We used the LFR benchmark datasets [24] to check the scalability of our proposed algorithms by varying the size of the nodes from $10K$ to $50K$. The result is presented in Fig. 14. In the LFR benchmark networks, we set the average degree as 5, maximum degree as 50, minimum community size as 10, maximum community size as 500, and community mixing parameter as 0.1. We also injected $\frac{|+|}{2}$ negative edges. We found that the proposed FBA is slower than DFBA since it must compute all the followers in every iteration. Even if the time complexity of DFBA is the same as that of FBA, our pruning strategy makes DFBA much faster than FBA. Notably, FCA is over 100 times faster than DFBA when $|V| = 50,000$ since it does not require computing the exact number of followers to find the best nodes to be removed.

4.6. Varying the radius r of FCA

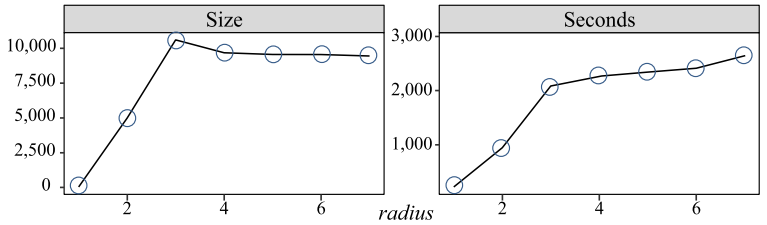

Fig. 15. Varying radius r.

In this section, we show the results of FCA by varying the radius r. We expect that when r becomes very large, it may not return desired results since it is easy to find the local optimum.

In FCA, the radius r is a user specified parameter. In Fig. 15, we check the result of (p, n)-core by varying the radius r in the EP dataset. As r becomes larger, the running time increases since the time complexity of HyperANF depends on the number of hops. Interestingly, we find that the size of the solution changes with r. We observe that when $r = 3$, the resultant subgraph returns the largest subgraph as a result. This implies that checking too much information may not be useful for estimating the number of followers. Therefore, in this paper, we set $r = 2$ for an efficiency-focus mode or $r = 3$ for an effectiveness-focus mode.

4.7. Case study: community search

Lastly, we show a case study to check the usefulness of our problem. One famous community search model is to maximise the minimum degree [38]. To test our algorithm, we fixed $n = 5$ and applied DFBA and FCA with $r = 3$ to the OTC dataset, using v_1 as the query node. In Fig. 16, the positive edges are grey-coloured, and the negative edges are red-coloured. We observe that the query nodes are densely connected to the other nodes in the community and there are few negative edges, and the degree of negative edges of the nodes is less than 5. The statistics of the identified community is as follows. In Table 5, CC indicates the global clustering coefficient and Δ indicates triangles. Note that we use the global clustering coefficient which is the # of closed triplets (or $3 \times$ triangles) over the total # of triplets. A large clustering coefficient indicates the presence of dense modules.

Table 5
Statistics of the identified communities.

	$\|V\|$	$\|E\|$	CC	# Δ	Edge ratio ($\|-\|/\|+\|$)
DFBA	95	1,341	0.41	5,693	0.015
FCA($r = 3$)	91	1,217	0.409	4,833	0.011

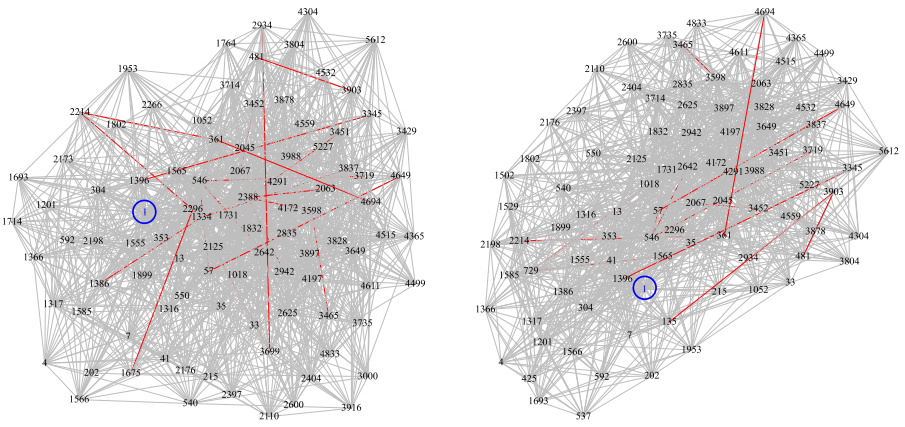

Fig. 16. Identifying a community of v_1.

5. Related work

5.1. Structural pattern discovery in networks

Identifying structural patterns in networks gains many attentions due to its various applications [38,27,7,10]. In this paper, we study signed cohesive subgraphs discovery problem. Thus, we focus on discussing representative cohesive subgraph models.

The k-core is the most widely used cohesive model for cohesive subgraph discovery problem. Given a graph and user parameter k, it aims to find a set of nodes of which every node has at least k neighbour nodes in the induced subgraph. The k-core has two main characteristics; (1) uniqueness; and (2) hierarchical structure. Computing an exact k-core takes $O(|E|)$ time. There are several variations of k-core in signed networks [13], directed networks [14], streaming network [35], distance-generalised network [4], weighted graph [12], multi-layer graph [12], and bipartite graph [9]. To get more details, readers may refer to this survey paper [29].

The k-truss is another cohesive subgraph model that incorporates graph triangles to define a cohesive subgraph. Given a graph and positive integers $k \geq 2$, it aims to find a maximal subgraph in which all edges are contained in at least $(k-2)$-triangles in the subgraph. Computing an exact k-truss takes $O(|E|^{1.5})$ time. Although k-truss takes more time compared with k-core, it finds more cohesive subgraphs, i.e., cohesiveness level of k-truss is higher than k-core.

5.2. Signed network mining

In the 1940s and 50s, Heider [17] and Cartwright and Harary [6] presented structural balance theory in signed networks. Since then, signed network analysis gained much attention in social psychology, computer science, physics, and so on. In a signed network, positive links are usually considered friendship and trust, while negative edges imply distrust and foes and so on. In the computer science area, signed networks have been used to solve various research problems including node ranking [19], community detection [39], sign prediction [28], opinion maximisation [16], and influence maximisation [18].

In this section, we introduce two representative theories named balance theory and status theory. First, the structural balance theory is as follows.

- **Balance theory** is firstly proposed by Heider [17] and Cartwright and Harary [6]. It focuses on the triangles in signed networks. The high-level idea of the balance theory is as follows [25]: *the friend of my friend is my friend*. We can also imagine various scenarios. For example, when there is only a single positive edge with two negative edges, it indicates that (1) *the friend of my enemy is my enemy*; (2) *the enemy of my friend is my enemy*; and (3) *the enemy of my enemy is my friend*.
- **Status theory** is developed by [25]. In the directed signed network, a positive edge from A to B can be interpreted as "B has a higher status than A". Similarly, a negative edge from A to B can be considered as "B has lower status than A".

For more detailed explanations with a comparison, readers may refer to the paper [46].

5.3. Cohesive subgraphs in signed networks

There are several works for finding cohesive subgraphs in signed networks. We present three works. First, in [44], Wu et al. study a signed (k,r)-truss problem. They present balanced and unbalanced triangles to model (k,r)-truss. Specifically, given a signed network G and two positive integers k and r, a signed (k,r)-truss is a subgraph S of G which satisfies (1) $sup^{+}(e,S) \geq k$; (2) $sup^{-}(e,S) \leq r$; and (3) maximality constraint. Support $sup^{+}(e,S)$ (or $sup^{-}(e,S)$) indicates that the number of balanced (or unbalanced) triangles contain

the edge e in S. They define that a triangle is balanced if it contains an odd number of positive edges; otherwise, the triangle is unbalanced. In this paper, they show that the proposed (k,r)-truss is NP-hard to find an exact solutions and show that the different edge deletion orders may lead to different (k,r)-trusses. To find a solution, Wu et al. propose three algorithms: (1) the trivial approach which removes the edges based on the edge id; (2) the greedy edge removing approach which iteratively chooses an edge with the least followers; and (3) the triangle-based approach, which is a revised greedy algorithm by selecting the best edge which can remove many unbalanced triangles. Since this problem focuses on balanced and unbalanced triangles by considering the truss, it is different from our (p,n)-core computation problem. Instead of considering the balance theory, we directly consider the number of internal edges and external edges to guarantee high-level positive edge cohesiveness and low-level negative edge cohesiveness.

Next, Giatsidis et al. [13] study the signed (l^t,k^s)-core problem in signed directed networks. Specifically, given a signed directed network G, and two parameters k and l, it aims to find (l^t,k^s)-core which is a maximal subgraph H of G, where each node $v \in H$ has $deg^s_{in}(v,H) \geq k$ and $deg^t_{out}(v,H) \geq l$. Note that $s,t \in \{+,-\}$. As we have discussed before, this study does not consider the internal negative edges, so the resultant (l^t,k^s)-core may contain many negative internal edges which leads to meaningless results. In our work, by pursuing deficient internal negative edges, we achieve high-quality cohesive graphs in signed networks.

Finally, Li et al. [26] investigate the (α,k)-clique problem. Specifically, given α, k, and r, it aims to enumerate all maximal (α,k)-cliques and find the top r maximal cliques where (α,k)-clique is a clique that satisfies negative and positive constraints. The authors prove that maximal (α,k)-clique problem is NP-hard. In the paper, the authors present a branch and bound enumeration algorithm with several pruning rules to enumerate all maximal (α,k)-cliques. Since (α,k)-clique is a clique, the problem returns different results compared with our model.

6. Conclusion

In this paper, we investigate a novel core computation problem in signed networks by taking into account positive and negative edges simultaneously. We demonstrate that a solution of (p,n)-core is not unique and NP-hard to find an exact solution; thus, we suggest three algorithms to solve the problem. Through extensive experiments on both real-world and synthetic networks, we demonstrate the superiority of our proposed algorithms. As future work, we plan to explore the dynamic signed networks to find (p,n)-core by updating the existing solutions. In addition, designing an approximation algorithm which balances effectiveness and efficiency remains a key challenge in this work.

CRediT authorship contribution statement

Junghoon Kim: Conceptualization, Data curation, Formal analysis, Investigation, Methodology/Study design, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Hyun Ji Jeong:** Methodology/Study design, Software, Supervision, Validation, Writing – original draft, Writing – review & editing. **Sungsu Lim:** Data curation, Formal analysis, Funding acquisition, Project administration, Resources, Supervision, Validation, Writing – original draft, Writing – review & editing. **Jungeun Kim:** Data curation, Formal analysis, Funding acquisition, Project administration, Resources, Supervision, Validation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2020-0-01336, Artificial Intelligence Graduate School Program (UNIST)) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2022-00155857, Artificial Intelligence Convergence Innovation Human Resources Development (Chungnam National University)). Also, this research was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A4A1031509).

References

[1] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.
[2] V. Batagelj, M. Zaversnik, An O(m) algorithm for cores decomposition of networks, arXiv preprint, arXiv:cs/0310049.
[3] P. Boldi, M. Rosa, S. Vigna, HyperANF: approximating the neighbourhood function of very large graphs on a budget, in: WWW, 2011, pp. 625–634.
[4] F. Bonchi, A. Khan, L. Severini, Distance-generalized core decomposition, in: SIGMOD, 2019, pp. 1006–1023.
[5] U. Brandes, P. Kenis, J. Lerner, D. Van Raaij, Network analysis of collaboration structure in Wikipedia, in: WWW, 2009, pp. 731–740.

[6] D. Cartwright, F. Harary, Structural balance: a generalization of Heider's theory, Psychol. Rev. 63 (5) (1956) 277.
[7] Y. Chen, D. Mo, Community detection for multilayer weighted networks, Inf. Sci. 595 (2022) 119–141.
[8] P. Chunaev, Community detection in node-attributed social networks: a survey, Comput. Sci. Rev. 37 (2020) 100286.
[9] D. Ding, H. Li, Z. Huang, N. Mamoulis, Efficient fault-tolerant group recommendation using alpha-beta-core, in: CIKM, 2017, pp. 2047–2050.
[10] Y. Fang, X. Huang, L. Qin, Y. Zhang, W. Zhang, R. Cheng, X. Lin, A survey of community search over big graphs, VLDB J. 29 (1) (2020) 353–392.
[11] P. Flajolet, É. Fusy, O. Gandouet, F. Meunier, Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm, in: Discrete Mathematics and Theoretical Computer Science, 2007, pp. 137–156.
[12] E. Galimberti, F. Bonchi, F. Gullo, Core decomposition and densest subgraph in multilayer networks, in: CIKM, 2017, pp. 1807–1816.
[13] C. Giatsidis, B. Cautis, S. Maniu, D.M. Thilikos, M. Vazirgiannis, Quantifying trust dynamics in signed graphs, the S-Cores approach, in: SDM, 2014, pp. 668–676.
[14] C. Giatsidis, D.M. Thilikos, M. Vazirgiannis, D-cores: measuring collaboration of directed graphs based on degeneracy, Knowl. Inf. Syst. 35 (2) (2013) 311–343.
[15] M. Girvan, M.E. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. 99 (12) (2002) 7821–7826.
[16] Q. He, L. Sun, X. Wang, Z. Wang, M. Huang, B. Yi, Y. Wang, L. Ma, Positive opinion maximization in signed social networks, Inf. Sci. 558 (2021) 34–49.
[17] F. Heider, Attitudes and cognitive organization, J. Psychol. 21 (1) (1946) 107–112.
[18] W. Ju, L. Chen, B. Li, W. Liu, J. Sheng, Y. Wang, A new algorithm for positive influence maximization in signed networks, Inf. Sci. 512 (2020) 1571–1591.
[19] J. Jung, W. Jin, L. Sael, U. Kang, Personalized ranking in signed networks using signed random walk with restart, in: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE, 2016, pp. 973–978.
[20] J. Kim, J. Kim, H.J. Jeong, S. Lim, LUEM: local user engagement maximization in networks, Knowl.-Based Syst. 255 (2022) 109788.
[21] J. Kim, S. Lim, (p,n)-core: core decomposition in signed networks, in: International Conference on Database Systems for Advanced Applications, Springer, 2022, pp. 543–551.
[22] J. Kim, S. Lim, J. Kim, OCSM: finding overlapping cohesive subgraphs with minimum degree, Inf. Sci. 607 (2022) 585–602.
[23] S. Kumar, F. Spezzano, V. Subrahmanian, C. Faloutsos, Edge weight prediction in weighted signed networks, in: ICDM, IEEE, 2016, pp. 221–230.
[24] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (4) (2008) 046110.
[25] J. Leskovec, D. Huttenlocher, J. Kleinberg, Signed networks in social media, in: SIGCHI, 2010, pp. 1361–1370.
[26] R.-H. Li, Q. Dai, L. Qin, G. Wang, X. Xiao, J.X. Yu, S. Qiao, Efficient signed clique search in signed networks, in: ICDE, IEEE, 2018, pp. 245–256.
[27] D. Liu, Z. Chang, G. Yang, E. Chen, Hiding ourselves from community detection through genetic algorithms, Inf. Sci. 614 (2022) 123–137.
[28] S.-Y. Liu, J. Xiao, X.-K. Xu, Sign prediction by motif naive Bayes model in social networks, Inf. Sci. 541 (2020) 316–331.
[29] F.D. Malliaros, C. Giatsidis, A.N. Papadopoulos, M. Vazirgiannis, The core decomposition of networks: theory, algorithms and applications, VLDB J. 29 (1) (2020) 61–92.
[30] S. Maniu, T. Abdessalem, B. Cautis, Casting a web of trust over Wikipedia: an interaction-based approach, in: WWW, 2011, pp. 87–88.
[31] F. Meng, M. Medo, B. Buechel, Whom to trust in a signed network? Optimal solution and two heuristic rules, Inf. Sci. 606 (2022) 742–762.
[32] D. Michail, J. Kinable, B. Naveh, J.V. Sichi, JGraphT—a Java library for graph data structures and algorithms, ACM Trans. Math. Softw. 46 (2) (2020) 1–29.
[33] Y.-W. Niu, C.-Q. Qu, G.-H. Wang, J.-L. Wu, G.-Y. Yan, Information spreading with relative attributes on signed networks, Inf. Sci. 551 (2021) 54–66.
[34] C.R. Palmer, P.B. Gibbons, C. Faloutsos, ANF: a fast and scalable tool for data mining in massive graphs, in: SIGKDD, 2002, pp. 81–90.
[35] A.E. Sariyüce, B. Gedik, G. Jacques-Silva, K.-L. Wu, Ü.V. Çatalyürek, Streaming algorithms for k-core decomposition, Proc. VLDB Endow. 6 (6) (2013) 433–444.
[36] J. Scott, Social network analysis, Sociology 22 (1) (1988) 109–127.
[37] S.B. Seidman, Network structure and minimum degree, Soc. Netw. 5 (3) (1983) 269–287.
[38] M. Sozio, A. Gionis, The community-search problem and how to plan a successful cocktail party, in: KDD, 2010, pp. 939–948.
[39] R. Sun, C. Chen, X. Wang, Y. Zhang, X. Wang, Stable community detection in signed social networks, IEEE Trans. Knowl. Data Eng. 34 (10) (2020) 5051–5055.
[40] J. Tang, Y. Chang, C. Aggarwal, H. Liu, A survey of signed network mining in social media, ACM Comput. Surv. 49 (3) (2016) 1–37.
[41] B.A.N. Travençolo, L.d.F. Costa, Accessibility in complex networks, Phys. Lett. A 373 (1) (2008) 89–95.
[42] C. Wang, H. Wang, H. Chen, D. Li, Attributed community search based on effective scoring function and elastic greedy method, Inf. Sci. 562 (2021) 78–93.
[43] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 440–442.
[44] Y. Wu, R. Sun, C. Chen, X. Wang, Q. Zhu, Maximum signed (k, r)-truss identification in signed networks, in: CIKM, 2020, pp. 3337–3340.
[45] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, Knowl. Inf. Syst. 42 (1) (2015) 181–213.
[46] J. Yap, N. Harrigan, Why does everybody hate me? Balance, status, and homophily: the triumvirate of signed tie formation, Soc. Netw. 40 (2015) 103–122.