Efficient Influential Community Search in Large Uncertain Graphs

Wensheng Luo, Xu Zhou, Kenli Li, *Member, IEEE*, Yunjun Gao, *Member, IEEE*, and Keqin Li, *Fellow, IEEE*

Abstract—Influential community search aims to find cohesive subgraphs (communities) with considerable influence. It is a fundamental graph management operator that can play a crucial role in biological network analysis, activity organization, and other real-life applications. Existing research on influential community search is mainly focused on deterministic graphs with the assumption that influences between entities are certain. This assumption is invalid in many cases because it ignores the uncertainty which is an inherent property of influence. Against this backdrop, in this paper, we introduce an uncertain influential community model, namely (k, η)-influential community, based on which the influential community search problem over uncertain graphs is formulated. Furthermore, we propose an online approach by integrating a peeling-pruning strategy that can progressively refine the given uncertain graph to find the (k, η)-influential communities. To further improve the search performance, two novel indexes, ICU-Index and FICU-Index, are developed to organize the (k, η)-influential communities at different probabilistic intervals. The indexes decompose the probabilistic interval into multiple subintervals and based on this, the (k, η)-influential communities are divided into different groups in turn. Compared with ICU-Index, FICU-Index requires considerably less space with the introduction of two optimization strategies. These indexes help obtain results of an influential community search problem more efficiently. Extensive experiments on large real and synthetic datasets demonstrate the efficiency and effectiveness of our proposed algorithms.

Index Terms—Graph queries, influential community search, network analysis, uncertain graphs

1 Introduction

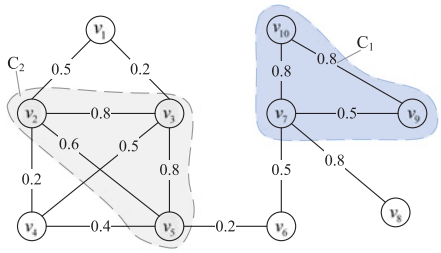
Community search, a fundamental problem in network analytics, has received extensive research interest. At present, studies on the community search problem fall broadly into two categories. The first category focuses on cohesion of the community structure including k-core [1], [2], [3], k-truss [4], [5], and k-clique [6], to name just a few. The second category looks into different attributes of vertices. Through examining the influence of vertices, influential community search is presented in [7]. The influential community search has been found to be instrumental in biological networks [8], protein-protein interaction networks [9], social networks [10], paper citation networks [7], activity organization [11], and other real-life applications.

There have been a number of studies on influential community search [7], [10], [11], [12], which, however, are primarily centered on deterministic graphs with the assumption that the influences between entities are fixed. This assumption is invalid in many cases as it overlooks the uncertainty which is an inherent property of the influences. In many applications, such as measuring social contagion of structural diversity [13], [14], describing biological functions of proteins [15], [16], [17], analyzing properties of a network structure [18], and selecting sentences in text summarization [19], data can be uncertain due to the possible influence of noise, precision of measurement, accuracy of prediction, protection of privacy, and other reasons [20]. Uncertain graphs are widely used to describe these data, in which vertices represent entities, and the probabilities of edges indicate the closeness of relations between entities. For example, in social networks, the probability represents the trust between users [21]; in protein interaction networks, the reaction between two substances is marked with confidence probability due to the accuracy of experiments [22]; likewise, the herd behavior among people can also be expressed by probability [23]. Therefore, calculating influential communities in uncertain graphs helps organize activities in a social network, identify research groups that are closely related in co-authorship networks, and discover the most critical structures of the receptor in PPI networks, to name a few.

In this paper, we attempt to investigate the issue of influential community search over uncertain graphs based on a new influential community model, namely (k, η)-influential community, for the first time. Specifically, for a given uncertain graph, a (k, η)-influential community is a connected subgraph of $\mathcal{G}$ such that for each vertex in the subgraph, its

- *Wensheng Luo, Xu Zhou, and Kenli Li are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410000, China. E-mail: {luowensheng, zhxu, lkl}@hnu.edu.cn.*
- *Yunjun Gao is with the College of Computer Science and Technology, Zhejiang University, Hangzhou, Zhejiang 310027, China. E-mail: gaoyj@zju.edu.cn.*
- *Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12651 USA. E-mail: lik@newpaltz.edu.*

Manuscript received 1 Nov. 2020; revised 6 Oct. 2021; accepted 21 Nov. 2021. Date of publication 30 Nov. 2021; date of current version 7 Mar. 2023.
This work was supported in part by the NSFC under Grants 61772182, 61802032, 62172146, and 62172157, in part by the Key Area Research Program of Hunan under Grant 2019GK2091, in part by the Open Research Projects of Zhejiang Lab under Grant 2021KD0AB02, and in part by the Project of Hunan Science and Technology Innovation Plan under Grant 2020RC2032.
(Corresponding author: Xu Zhou.)
Recommended for acceptance by A. Khan.
Digital Object Identifier no. 10.1109/TKDE.2021.3131611



(a)

Vertex	Influence
v_1	4
v_2	2
v_3	1
v_4	3
v_5	7
v_6	5
v_7	9
v_8	6
v_9	10
v_{10}	8

(b)

Fig. 1. A vertex-weighted uncertain graph.

probability of owning at least k neighbors (k-probability) is no less than a specified threshold η. Moreover, a subgraph C is not a (k,η)-influential community if there exists a subgraph C' containing C and has the same influence with C. The influence of a community is the minimum weight of all the vertices in it. This is because using the minimum weight of all vertices as the influence of the community guarantees that all community members have significant influence. Note that we follow the assumption that each vertex in $\mathcal{G}$ has a distinct weight [7].

Example 1. Fig. 1 shows an uncertain graph $\mathcal{G}$ from a co-authorship network. It consists of ten vertices that represent different authors. The influence (weights) of the vertices is the h-index of authors, and the probabilities of edges denote co-authorship relationships between different authors. Without loss of generality, a large probability means close collaboration between authors. The (k,η)-influential communities investigated in this paper are helpful to identify research groups where authors have strong co-authorship relationships.

With $k=1$ and $\eta=0.8$, the $(1,0.8)$-influential communities of $\mathcal{G}$ are $C_1=\{v_7,v_9,v_{10}\}$, $C_2=\{v_2,v_3,v_5\}$, and $C_3=\{v_7,v_8,v_9,v_{10}\}$, in which the probability of each vertex owning at least 1 neighbor is no less than 0.8. The influence of these communities are $\omega(v_{10})=8$, $\omega(v_3)=1$, and $\omega(v_8)=6$, respectively. Given $k=1$ and $\eta=0.92$, there is only one $(1,0.92)$-influential community of $\mathcal{G}$, which is $C_2=\{v_2,v_3,v_5\}$ whose influence is $\omega(v_3)=1$. Compared to C_2, $C_1=\{v_7,v_9,v_{10}\}$ has a higher influence, which means that the authors in C_1 have higher h-indexes. The community C_2 has a larger value of η, which indicates that the authors in C_2 have a closer cooperative relationship. It is noted that C_2 is both a $(1,0.8)$-influential community and a $(1,0.92)$-influential community. This can be suggestive of an inclusion relationship between communities with different probability thresholds η under a specified k.

Why (k,η)-core? There are mainly two models, (k,η)-core and (k,θ)-core, proposed to assess the cohesion of uncertain graphs. The (k,η)-core [24] is proposed based on the probabilistic degree. It focuses on the local degree distribution and the support among k-core members. In contrast, (k,θ)-core in [25] is more concerned with the k-core probability of each individual vertex, i.e., the probability of each vertex appearing in all possible k-cores. Compared with (k,θ)-core, (k,η)-core is more in line with the influential community model over uncertain graphs. This is because (k,θ)-core is more appropriate for applications where k-core members are used individually, while (k,η)-core concentrates on the relationship of community members, which could integrate the influence between them.

In this paper, the influential community search problem is formulated over uncertain graphs. The smallest weight in the community is regarded as the influence of the community. Due to this property, an online algorithm obtaining all the (k,η)-influential communities is devised using a peeling strategy where the smallest weight vertex of the maximal (k,η)-core is iteratively deleted. Although this online algorithm can process the proposed problem effectively, it still faces several challenges.

Challenges. The online algorithm is time-consuming, especially for large-scale graphs. The time complexity of the online algorithm depends on the calculation of k-probabilities for vertices in a given graph. Even with the state-of-the-art dynamic programming (DP) in [24], it still costs $O(\sum_{v\in V} k{\cdot}d_v)$ time to compute all the k-probabilities where d_v is the maximum number of neighbors of vertex v in the graph. Accordingly, for large-scale graphs, the method is inefficient. For instance, our experiments in Section 8 show that for LiveJournal, a dataset with 10^8 edges, the query time of the online approach reaches 10^5 seconds.

This prompts us to propose an index, namely ICU-Index, to organize (k,η)-influential communities at different probabilistic intervals. Specifically, we first compute the range of k based on the topological structure of the given uncertain graph, that is, $k\in(0,k_{max}]$, and then save all the communities corresponding to η under each k. It is difficult to build an index for all possible communities corresponding to η because the probability threshold η is a real number between 0 and 1, which means there are infinite possible communities. To fix this problem, we decompose the probabilistic interval into multiple sub-intervals, and further divide the (k,η)-influential communities into different groups in turn. This is based on the observation that the corresponding communities of any η in a certain interval are consistent. The space complexity of ICU-Index is $O(k_{max}{\cdot}t{\cdot}m)$, where m is the number of edges in $\mathcal{G}$, and t is the number of intervals of the index. Moreover, we devise two optimization strategies and present an advanced forest-based index, namely FICU-Index, to further boost the query performance and reduce the space overhead of ICU-Index. It is proven that the space complexity of FICU-Index is $O(k_{max}{\cdot}t{\cdot}n)$. Based on these indexes, we can obtain influential communities efficiently.

Contributions. Our contributions are summarized as follows.

- We investigate the (k,η)-influential community search problem over uncertain graphs for the first time.
- We propose an online algorithm based on a peeling pruning strategy to compute the (k,η)-influential communities.
- We devise an index called ICU-Index to improve query efficiency by dividing the range of η into several intervals and storing the corresponding communities of each interval.
- We present two optimization strategies and devise an advanced index, namely FICU-Index, to optimize the query efficiency and lower the space overhead of the ICU-Index.

The rest of the paper is summarized as follows. Section 2 offers an overview of related research on the influential community search problem. We provide the background information and definition of the problem in Section 3. Our proposed online search algorithm is described in Section 4. Section 5 presents the index-based algorithm for searching the influential communities in uncertain graphs. In Section 6, two optimizations of the index are proposed and discussed, while Section 7 introduces the maintenance of the index. Our extensive experiment results are detailed in Section 8, before Section 9 concludes the paper.

2 Related Work

In this section, we review related work of the (k,η)-core in uncertain graphs and influential community search over deterministic graphs.

2.1 (k,η)-Core in Uncertain Graphs

k-core proposed in [1] is recognized as one of the most popular cohesive subgraphs, and there have been a significant number of studies on k-core over deterministic graphs [26], [27], [28], [29], [30]. Hence, representative studies of the k-core computation problem over uncertain graphs, which are closely related to our research in this paper, are reviewed.

Bonchi *et al.* [24] first identified the k-core problem in uncertain graphs, namely (k,η)-core. They also presented an η-core decomposition algorithm that can calculate the (k,η)-core effectively. Recently, Esfahani *et al.* [31] introduced the Lyapulov's central limit theorem to compute η-degrees. In this way, approximate results of the η-degree can be gained. A peeling algorithm and a sequential algorithm were further proposed for computing the (k,η)-cores. Besides, Yang *et al.* [32] developed an index-based solution which achieves better performance when computing the (k,η)-cores.

All the algorithms as mentioned above tend to focus on the cohesion of vertices while ignoring the attributes, such as influence, of vertices. Consequently, they cannot be directly adapted to obtain the influential communities in uncertain graphs.

2.2 Influential Community Search

For a given graph, an influential community is a cohesive subgraph with an influence. There are a varied definitions of cohesive subgraphs, e.g., k-core [1], k-truss [4], edge density [33], [34], [35], and edge connectivity [36], [37]. Among these studies, k-core-based community is widely used because it is easy to represent and calculate. Li *et al.* first defined the k-influential community based on the notion of k-core, and designed an index-based method to effectively retrieve influential communities in a graph [7]; Chen *et al.* [28] presented two global algorithms, forward and backward, to search the top-r influential communities in web-scale networks. Bi *et al.* [11] improved the influential community search performance by refining the original graph.

Admittedly, these approaches are very effective in the influential community search over deterministic graphs. It has to be pointed out that they cannot be directly utilized to process the influential community search over uncertain graphs investigated in this paper.

TABLE 1
Frequently-Used Notations

Notation	Definition
$G{=}(V,E,\omega)$	A deterministic graph with a vertex set V, an edge set E, and a function ω assigns a weight to each vertex
$\mathcal{G}{=}(V,E,p,\omega)$	An uncertain graph with a vertex set V, an edge set E, a function p that assigns a probability to each edge, and a function ω assigns a weight to each vertex
n,m	The number of vertices and edges, respectively
$deg(v)$	The degree of v in $\mathcal{G}$
d_v	The maximum number of neighbors of v in $\mathcal{G}$
$f(H)$	The influence of subgraph H
$\mathcal{F}(\mathcal{H})$	The influence of subgraph $\mathcal{H}$
p_e	The existence probability of the edge $e{\in}E$
η-$deg(v)$	The maximum k that satisfies the probability that v has at least k neighbors is not less than η
$H{=}(V_H,E_H,\omega)$	An induced subgraph of G where $V_H{\in}V, E_H{=}\{(u,v)\|u,v{\in}V_H,(u,v){\in}E\}$
$\mathcal{H}{=}(V_H,E_H,p,\omega)$	An induced subgraph of $\mathcal{G}$

3 Problem Statement

In this section, we formulate the influential community search problem over uncertain graphs. Table 1 summarizes the notations and their meanings in this paper.

Let $G=(V,E,\omega)$ be an undirected graph, where V and E denote a vertex set and an edge set, respectively, and ω is a function that assigns a weight to each vertex in V. In other words, $\omega(v)$ represents the weight of vertex v. For each vertex $v\in V$, the degree of v is denoted as $deg(v)$.

For an induced subgraph $H=(V_H,E_H,\omega)$ of G where $V_H{\subseteq}V, E_H=\{(u,v)|u,v\in V_H,(u,v)\in E\}$, the influence of H is defined as the minimum weight of vertices in V_H, i.e., $f(H)=\min_{v\in V_H}\{\omega(v)\}$. Specifically, the k-influential community over deterministic graphs is defined as follows.

Definition 1 (k-influential community [7]). *Given a graph G and $k\in\mathbb{N}$, $H\subseteq G$ is a k-influential community if it satisfies all the following constraints.*

- Connection*: H is a connected graph;*
- Cohesion*: $\forall v\in V_H$, $deg(v)\geq k$;*
- Maximality*: There is no other induced subgraph H' containing H that satisfies the two conditions above and $f(H')=f(H)$.*

Given an uncertain graph $\mathcal{G}=(V,E,p,\omega)$, where p is a function that assigns an existence probability to each edge in $\mathcal{G}$. The existence probability of an edge $e\in E$ is p_e. Without loss of generality, the existence probability of each edge is assumed to be independent.

The concept of possible world semantics [24] is adopted here, wherein a possible world represents a deterministic instance of an uncertain graph. Thus, there are $2^{|E|}$ instances of an uncertain graph. In this paper, an instance of $\mathcal{G}$ is denoted by $G(V,E_G){\sqsubseteq}\mathcal{G}$, where $E_G{\subseteq}E$. The probability of $G(V,E_G)$ is

$$Pr(G)=\prod_{e\in E_G}p_e\prod_{e\in E\setminus E_G}(1{-}p_e). \tag{1}$$

According to Eq. (1), the probability of $v \in \mathcal{G}$ owning at least k neighbors is

$$Pr[deg(v, \mathcal{G}) \geq k] = \sum_{G \sqsubseteq \mathcal{G}} Pr(G), \tag{2}$$

where each G has $deg(v, G) \geq k$.

Definition 2 ((k, η)-core [24]). *Given an uncertain graph $\mathcal{G}$ and two parameters $k \in \mathbb{N}$ and $\eta \in [0, 1]$, a maximal induced subgraph $\mathcal{H} = (V_H, E_H, p, \omega)$ of $\mathcal{G}$ is a (k, η)-core if $\forall v \in V_H, Pr[deg(v, \mathcal{H}) \geq k] \geq \eta$.*

Based on (k, η)-core, the (k, η)-influential community search over uncertain graphs is formulated as follows.

Definition 3 ((k, η)-influential community). *Given an uncertain graph $\mathcal{G}$, two parameters k and η, a (k, η)-influential community of $\mathcal{G}$ is a maximal connected subgraph $\mathcal{H}$ satisfying all the following constraints.*

- Connection*: $\mathcal{H}$ is a connected graph;*
- Cohesion*: $\forall v \in V_{\mathcal{H}}$, η-$deg(v) \geq k$;*
- Maximality*: There is no other induced subgraph $\mathcal{H}'$ containing $\mathcal{H}$ which satisfies the two conditions above and $\mathcal{F}(\mathcal{H}') = \mathcal{F}(\mathcal{H})$.*

Problem Statement. Given an uncertain graph $\mathcal{G}(V, E, p, \omega)$ and two parameters $k \in \mathbb{N}$ and $\eta \in [0, 1]$, the problem of influential community search aims to find all the (k, η)-influential communities of $\mathcal{G}$.

4 An Online Search Algorithm

In this section, we propose an online search approach for calculating the (k, η)-influential community. According to Definition 3, the maximal connected (k, η)-core of $\mathcal{G}$ is also a (k, η)-influential community for a given weighted uncertain graph $\mathcal{G}$. Therefore, our proposed online search approach consists of two stages: calculating the connected components of (k, η)-core of $\mathcal{G}$ and computing (k, η)-influential communities. In what follows, we will discuss the two stages in depth.

4.1 (k, η)-Core Calculation

Let $\mathcal{G}(V, E, p, \omega)$ be an uncertain graph, and the maximum degree of each vertex $v \in V$ be denoted as d_v, the number of all the neighbors of the vertex v in $\mathcal{G}$. For a vertex v, its probability of owning at least k neighbors is computed as

$$\begin{aligned} Pr[deg(v) \geq k] &= \sum_{i=k}^{d_v} Pr[deg(v) = i] \\ &= 1 - \sum_{i=0}^{k-1} Pr[deg(v) = i]. \end{aligned} \tag{3}$$

In the following, $Pr[deg(v) \geq k]$ is denoted as k-$prob(v)$ for simplicity. In case $deg(v) < k$, it holds that k-$prob(v) = 0$. For each vertex in a (k, η)-core, its k-$prob$ is not less than η.

Given the properties of (k, η)-core, the calculation of (k, η)-core is depicted in Algorithm 1. The main idea of our algorithm is consistent with the (k, η)-core decomposition proposed in [24]. A major difference lies in that (k, η)-core does not require all vertices to be connected, while (k, η)-influential community is a connected subgraph of $\mathcal{G}$. Therefore, we calculated all the connected components in (k, η)-core.

Algorithm. Algorithm 1 takes an uncertain graph $\mathcal{G}$, two parameters k and η as inputs. Given an uncertain graph $\mathcal{G}$, for each (k, η)-core C, there exists a k-core C' that contains C. Accordingly, $\mathcal{G}$ is first refined by calculating the k-core in the deterministic graph of $\mathcal{G}$ (Line 1). Line 2 initializes a core set $CoreS$ including all the (k, η)-cores of the given uncertain graph $\mathcal{G}$. Then we compute k-$prob(v)$ of each vertex $v \in \mathcal{G}$ (Line 3). After that, (k, η)-cores are computed by iteratively deleting the vertices whose k-probabilities are less than η (Lines 4-6). Moreover, k-probabilities of all the vertices that are neighbors of the vertex v need to be updated. The for-loop (Lines 4-6) cannot be stopped until the k-probability of every vertex left is larger than the probabilistic threshold η. This means that after the for-loop, the vertices in the given graph $\mathcal{G}$ are retained if their k-probabilities are no less than η. Based on this refined graph $\mathcal{G}$, Lines 7 and 8 are executed to find all the connected subgraphs C of $\mathcal{G}$, which will be further added to the result set $CoreS$.

Algorithm 1. (k, η)-Core Calculation

Input: An uncertain graph $\mathcal{G}$, two parameters k and η
Output: Connected components of (k, η)-core
1 $\mathcal{G} \leftarrow$ the k-core of $\mathcal{G}$;
2 Initialize a (k, η)-core set $CoreS \leftarrow \varnothing$;
3 Calculate k-$prob(v)$ for all the vertices $v \in \mathcal{G}$;
4 **foreach** *v in $\mathcal{G}$ with k-$prob(v) < \eta$* **do**
5 Update $\mathcal{G}$ by removing the vertex v ;
6 Recompute the k-$probs$ for all the neighbors of v;
7 **foreach** *Connected subgraph $\mathcal{G}'$ in $\mathcal{G}$* **do**
8 Add $\mathcal{G}'$ to $CoreS$ as a connected component of (k, η)-core;
9 **return** $CoreS$

As shown in Algorithm 1, the processing time largely depends on the calculation and update cost of the k-probabilities of the vertices. A direct way to calculate k-$prob(v)$ is to compute $Pr[deg(v) = i]$ with i increasing from 0 to $k-1$. However, the recalculation of k-$prob(v)$ is inefficient in this way, especially when the edges related to v are removed.

To address this concern, we adopted a dynamic programming approach to compute the k-probability of each vertex [24]. Let $Pr[deg(v|\{e_1, \ldots, e_h\}) = j]$ be $D(h, j)$, where $h \in [1, d_v]$ and $j \in [0, h]$. If the vertex v has k neighbors, then there will be two situations, (1) the addition of e_h makes the degree of v equal to k; (2) prior to the addition of e_h, the degree of v is already k, that is

$$D(h, j) = p_{e_h} \cdot D(h-1, j-1) + (1 - p_{e_h}) \cdot D(h-1, j), \tag{4}$$

where p_{e_h} is the existence probability of the edge e_h. The base case of Eq. (4) is

$$\begin{cases} D(0, 0) = 1, \\ D(h, -1) = 0, & h \in [0, d_v], \\ D(h, j) = 0, & h \in [0, d_v],\ j > h. \end{cases} \tag{5}$$

Similarly, k-$prob$ can be updated by

$$Pr[deg(v|E(v) \setminus \{e\}) = i] = \frac{1}{1-p_e} \cdot \Big(Pr[deg(v) = i] - p_e \cdot Pr[deg(v|E(v) \setminus \{e\}) = i-1] \Big), \tag{6}$$

where $i \in [1, k]$, $p_e \neq 1$, and if $i = 0$

$$Pr[deg(v|E(v) \setminus \{e\}) = 0] = \frac{1}{1-p_e} \cdot Pr[deg(v) = 0]. \tag{7}$$

Theorem 1. *The time complexity of Algorithm 1 is $O(k{\cdot}m)$.*

Proof. The (k, η)-core calculation algorithm first calculates the k-core of $\mathcal{G}$, and then calculates k-$prob$ of each vertex. It needs $O(n{+}m)$ time to calculate the k-core and requires $O(k{\cdot}d_v)$ time to compute k-$prob$ of a vertex v [32]. As a result, it costs $O(\sum_{v \in V} k{\cdot}d_v)$ time to initialize k-probabilities of all the vertices.

For a vertex, the time complexity of updating the k-$prob$ based on Eq. (4.1) is $O(k)$, and then it requires $O(k{\cdot}d_v)$ time to update the k-probabilities of its neighbors. Accordingly, for all the vertices in $\mathcal{G}$, this updating operator (Lines 3 to 5) costs $O(\sum_{v \in V} k{\cdot}d_v)$ time.

To summarise, the total time cost of the (k, η)-core calculation algorithm is

$$O\left(\sum\nolimits_{v \in V} k{\cdot}d_v\right) + O\left(\sum\nolimits_{v \in V} k{\cdot}d_v\right) + O(m+n) = O\left(\sum\nolimits_{v \in V} k{\cdot}d_v\right) = O(k{\cdot}m).$$

Therefore, the theorem holds. □

4.2 (k, η)-Influential Community Search

This subsection detailed our (k, η)-influential community search algorithm.

On the basis of Definition 3, we have the following observation.

Observation 1. *Given a (k, η)-influential community of $\mathcal{G}$, there may exist (k, η)-influential communities in it with higher influence.*

Example 2. As shown in Fig. 1, given $k = 1$ and $\eta = 0.8$, $\{v_7, v_8, v_9, v_{10}\}$ is a (1, 0.8)-influential community whose influence is $\omega(v_8) = 6$. Note that $\{v_7, v_9, v_{10}\} \subseteq \{v_7, v_8, v_9, v_{10}\}$ is equally a (1, 0.8)-influential community whose influence is $\omega(v_{10}) = 8$. That is, the influential community $\{v_7, v_9, v_{10}\}$ with influence 8 is contained in $\{v_7, v_8, v_9, v_{10}\}$ with a lower influence of 6.

Based on Observation 1, given k and η, to calculate all the corresponding (k, η)-influential communities of $\mathcal{G}$, we can first obtain all the connected components of (k, η)-core and then decompose them by iteratively removing the vertex with the smallest weight.

Algorithm. The pseudo-code of the (k, η)-influential community search algorithm is depicted in Algorithm 2. In Algorithm 1, the (k, η)-core of $\mathcal{G}$ needs to be obtained initially. Then, all the connected components of (k, η)-core (i.e., $\hat{\mathcal{G}}$) are added to the result set Res (Lines 2-3). A key component of the (k, η)-influential community search algorithm is a while loop that computes the (k, η)-influential communities with a peeling strategy. Specifically, for each $\hat{\mathcal{G}} \subseteq \mathcal{G}$, the vertex with the smallest weight is removed and the k-probabilities of the remaining vertices are updated accordingly. Subsequently, the (k, η)-core of $\hat{\mathcal{G}}$ is computed and utilized to refresh $\hat{\mathcal{G}}$. All the previous steps are repeated until $\mathcal{G}$ is empty (Lines 4-8). Finally, all the communities in Res are returned (Line 9).

Algorithm 2. (k, η)-Influential Community Search

Input: An uncertain graph $\mathcal{G}$, an integer k, and a probability threshold η
Output: (k, η)-influential communities of $\mathcal{G}$

1 Initialize an influential community set $Res \leftarrow \varnothing$;
2 $\mathcal{G} \leftarrow$ the (k, η)-core of $\mathcal{G}$;
3 Add the connected components of $\mathcal{G}$ to Res;
4 **while** $\mathcal{G}$ *is not empty* **do**
5 **for** *connected component* $\hat{\mathcal{G}} \subseteq \mathcal{G}$ **do**
6 Remove the vertex with the smallest weight from $\hat{\mathcal{G}}$;
7 $\hat{\mathcal{G}} \leftarrow$ the (k, η)-core of $\hat{\mathcal{G}}$;
8 Add the newly generated connected components of $\hat{\mathcal{G}}$ to Res;
9 **return** Res

Example 3. Take Fig. 1 as an example. To gain all the $(1, 0.7)$-influential communities, the $(1, 0.7)$-cores of $\mathcal{G}$ are calculated as initial communities, that is, $COM_1 = \{v_2, v_3, v_4, v_5\}$ with influence 1, $COM_2 = \{v_7, v_8, v_9, v_{10}\}$ with influence 6. Then the $(1, 0.7)$-cores get decomposed. Note that COM_1 cannot be decomposed because if any vertex in COM_1 is removed, then the k-probabilities of remaining vertices will be less than 0.7. Consider the community COM_2 where the vertex v_8 has the smallest weight. By removing the vertex v_8 from COM_2 and updating the k-probabilities of its neighbors, we can gain a new community $COM_3 = \{v_7, v_9, v_{10}\}$ which cannot be further decomposed. Finally, we gain three $(1, 0.7)$-influential communities of $\mathcal{G}$ which are $\{v_2, v_3, v_4, v_5\}$, $\{v_7, v_8, v_9, v_{10}\}$, and $\{v_7, v_9, v_{10}\}$.

Theorem 2. *The time complexity of the (k, η)-influential community search algorithm is $O(k{\cdot}m)$.*

Proof. Algorithm 2 can be divided into two stages: calculating (k, η)-core and decomposing (k, η)-core. The while loop in Algorithm 2 decomposes the (k, η)-core by iteratively removing vertices with the smallest weight and updating the k-probabilities of its neighbors. Therefore, in the worst case, the time complexity of the decomposition is $O(\sum_{v \in V} k{\cdot}d_v)$. Since the time complexity of Algorithm 1 is $O(\sum_{v \in V} k{\cdot}d_v)$, the time complexity of Algorithm 2 is

$$O\left(\sum_{v \in V} k{\cdot}d_v\right) + O\left(\sum_{v \in V} k{\cdot}d_v\right) = O\left(\sum_{v \in V} k{\cdot}d_v\right) = O(k{\cdot}m).$$

□

5 An Index-Based Algorithm

Although the online approach proposed can effectively calculate the (k, η)-influential community, it is not suitable for processing large-scale graphs.

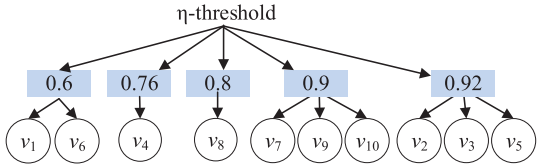


Fig. 2. η-threshold of all vertices in $\mathcal{G}$ $(k = 1)$.

In this section, we attempt to propose an index to improve query efficiency. That said, building an index for all possible communities is challenging. This is because the parameter η is a real number between 0 and 1, which means there are infinite possible values of η.

To tackle this problem,an index-based approach called influential community over uncertain graphs index (ICU-Index) is proposed. This partly result from a key observation that the corresponding communities of η in a certain interval are consistent. Specifically, communities are divided and organized as groups under different probability intervals.

5.1 The Structure of ICU-Index

The ICU-Index is utilized to organize communities under different k and η. To build the ICU-Index, we first fix the parameter k and then calculate the influential communities of all η under k. The range of k is $[1, k_{max}]$, where k_{max} is the largest k that enables the k-core of $\mathcal{G}$ to be non-empty.

As mentioned previously, different from k, η is a real number between 0 and 1, and thus there can be infinite possible values of η. This renders it is impractical to enumerate all the communities under different η. Besides, the communities are distinct under different η, so Algorithm 2 cannot be adapted to calculate the influential communities without specifying η.

To alleviate this problem, we make a key observation based on η-threshold [32]. The η-threshold of vertex u is the maximum value of η that enables u to be contained in the (k, η)-core.

Observation 2. *For a given $\mathcal{G}$ and k, (k, η)-cores of $\mathcal{G}$ have a nested property, that is, (k, η_j)-core $\subseteq$ (k, η_i)-core if $\eta_j > \eta_i$.*

Example 4. Consider the uncertain graph $\mathcal{G}$ in Fig. 1. With $k = 1$, Fig. 2shows the η-threshold of all the vertices in $\mathcal{G}$. Then, we have $(1, 0.92)$-core $\subseteq$ $(1, 0.9)$-core $\subseteq$ $(1, 0.8)$-core $\subseteq$ $(1, 0.76)$-core $\subseteq$ $(1, 0.6)$-core.

Let N_τ be the η-threshold set of all vertices in $\mathcal{G}$, $\eta_i \in N_\tau$, and η_{max} is the maximum value of N_τ. Due to the nested property of the (k, η)-cores, in the ICU-Index, the probability interval $[0, \eta_{max}]$ is divided into multiple subintervals so that under each subinterval it has the same (k, η)-influential communities. In this way, all the (k, η)-influential communities can be organized into different groups.

The ICU-Index is a data structure with different parts, each of which contains keys representing probability subintervals and values representing communities corresponding to probability intervals. Specifically, for any η belonging to a certain subinterval, the (k, η)-influential community search always returns the same communities associated with the subinterval in the ICU-Index. In this way, for a (k, η)-influential community search, we can readily get the final results by searching the subinterval where η belongs to.

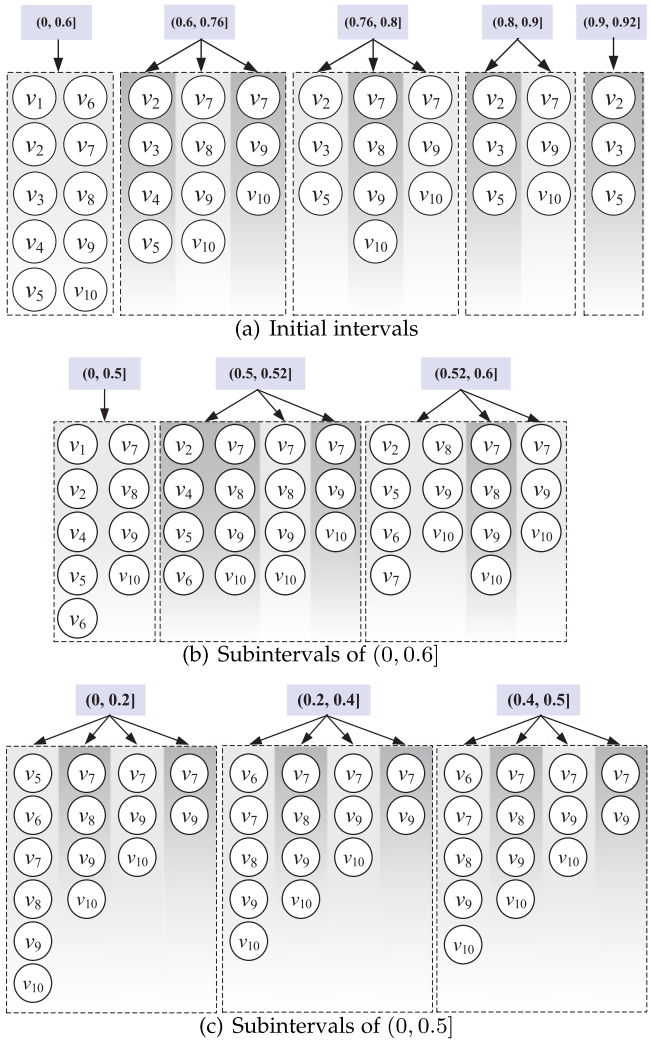


Fig. 3. ICU-Index of $\mathcal{G}$ $(k = 1)$.

Note that, for each interval, the connected component of (k, η)-core corresponding to the largest η is taken as its initial influential community. Then, we iteratively peel the vertex with the smallest weight in the initial community to get influential communities. Doing so can guarantee that most of the generated communities are located in the current interval. Therefore, each interval adopts the form of left open and right closed to get the largest η of the interval and cover all possible η. That is, $(0, \eta_1], (\eta_1, \eta_2], \ldots,$ and $(\eta_{i-1}, \eta_i]$. It is worth to notice that the first subinterval $(0, \eta_1]$ needs to be further subdivided. This subdivision operator is crucial to make sure for a certain interval, corresponding communities are always the same, which can ensure accurate results for users.

Example 5. Fig. 3 shows the ICU-Index of the graph in Fig. 1 with $k = 1$. The probability threshold $[0, 0.92]$ is divided into five subintervals $(0, 0.6], (0.6, 0.76], (0.76, 0.8], (0.8, 0.9]$, and $(0.9, 0.92]$. Each interval is associated with one or more communities. For instance, the probability interval $(0.9, 0.92]$ is associated with a community $\{v_2, v_3, v_5\}$. This indicates that for all the (k, η)-influential

community search with $\eta \in (0.9, 0.92]$, the community $\{v_2, v_3, v_5\}$ could be directly returned as final results directly.

5.2 Index Construction

To construct ICU-Index, we need to divide the interval $(0, \eta_{max}]$ into several subintervals and then calculate the communities corresponding to the left boundary value of each subinterval. Since each interval is left open, in order to calculate the communities corresponding to these intervals, we present (k, τ)-core as follows.

Definition 4 ((k, τ)-core). *Given an uncertain graph $\mathcal{G}$ and two parameters k and τ, $\mathcal{H}$ is a (k, τ)-core of $\mathcal{G}$ if the probability of each vertex in V_H owning at least k neighbors is greater than τ, i.e., $\forall v \in V_H, Pr[deg(v, \mathcal{H}) \geq k] > \tau$.*

The definition of (k, τ)-core is similar to that of (k, η)-core (Definition 2). Differently, it holds that the probability of each vertex in (k, τ)-core owning at least k neighbors exceeds the threshold τ.

In index construction, interval $(0, \eta_{max}]$ is first divided into i subintervals $(0, \eta_1], (\eta_1, \eta_2], \ldots, (\eta_{i-1}, \eta_i]$ for a specified k, where $\eta_i = \eta_{max}$. Any η in the same subinterval corresponds to the same influential communities. Note that, the initial community under subinterval $(\eta_j, \eta_{j+1}]$ $(0 \leq j < i)$ can be obtained by computing (k, τ)-core with the probability threshold η_j. After that, all the influential communities under the subinterval $(\eta_j, \eta_{j+1}]$ are computed by the peeling-pruning strategy to progressively decompose the initial influential communities.

Different from Algorithm 2, after removing the vertices with the smallest weight, we need to calculate the (k, τ)-core of the subgraph consisting of the remaining vertices. Note that the decomposition of the current communities under a subinterval may incur subdivision of the current interval. This subdivision is necessary to ensure that under each probability subinterval the corresponding communities remain the same. In particular, the subdivision rule could be described as follows.

Subdivision Rule. Consider an interval $(\eta_j, \eta_{j+1}]$ for $0 \leq j < i$. After removing the vertices with the smallest weight in the current communities under this interval, it first updates k-probabilities of vertices contained in the refined communities. The refreshed k-probabilities in the interval $(\eta_j, \eta_{j+1}]$ are selected as boundaries of new subintervals. For example, assume there are i vertices with k-probabilities $p_i \in (\eta_j, \eta_{j+1}]$. The interval $(\eta_j, \eta_{j+1}]$ can then be divided into $i + 1$ subintervals, i.e., $(\eta_j, p_0]$, $(p_0, p_1], \ldots, (p_i, \eta_{j+1}]$. This subdivision procedure is executed recursively until it cannot get any (k, η)-influential community under the new subinterval.

Algorithm. The ICU-Index construction algorithm is depicted in Algorithm 3. Let Ind_k be the ICU-Index of $\mathcal{G}$ under k; for each Ind_k, the η-thresholds of all vertices are calculated, and then (0, η_{max}] is divided into subintervals in non-descending order of these η-thresholds. That is, $(0, \eta_1]$, $(\eta_1, \eta_2], \ldots, (\eta_{i-1}, \eta_i]$, where $\eta_i = \eta_{max}$ (Lines 3-4). Each interval is further divided in accordance with the aforementioned subdivision rule. For the new intervals, the corresponding communities are computed (Lines 5-7). When decomposing the current communities, we first store each community under the current interval (Line 18), remove the vertex with the smallest weight and calculate the (k, τ)-core consisting of the remaining vertices (Lines 19-20). For the vertex whose k-*prob* is in the interval during the decomposition, its k-*prob* is used as a new boundary value to segment the current interval (Lines 21-24). The above steps will be repeated until all communities cannot be decomposed.

Algorithm 3. ICU-Index Construction

Input: An uncertain graph $\mathcal{G}(V, E, p, \omega)$.
Output: An index of all (k, η)-influential communities of $\mathcal{G}$.

```
1  for k = 1 to k_max do
2     Initialize Ind_k ← ∅ and Res ← ∅ // Ind_k is ICU-Index of G under k
3     η_t ← η-threshold of all vertices in ascending order ;
4     subInt← intervals from (0, η_1] to (η_{max-1}, η_max] // Divide (0, η_max] into subintervals
5     foreach Invl in subInt do
6        Res ←Segment(Invl, G, Res);
7        Append Res to Ind_k;
8     return Ind_k;
9  return all Ind_k from 1 to k_max
10
11 Procedure Segment(Invl, g, res)
12       η_i ← the left boundary of Invl;
13       η_j ← the right boundary of Invl;
14       τ ← η_i;
15       C ← (k, τ)-core of g;
16       foreach connected component C_n ⊆ C do
17          while C_n is not empty do
18             Append C_n to res.Invl;
19             u ← argmin_{v∈C_n} ω(v);
20             C_n ← (k, τ)-core of C_n \ {u};
21             foreach v ∈ C_n do
22                if k-prob(v) ∈ (η_i, η_j) then
23                   Segment((η_i, k-prob(v)], C_n, res);
24                   Segment((k-prob(v), η_j], C_n, res);
25       return res
```

Example 6. For $\mathcal{G}$ in Fig. 1 with $k = 1$, we have a η-threshold set $\{0.92, 0.9, 0.8, 0.76, 0.6\}$, and the initial interval is $(0, 0.92]$, which is divided into subintervals $(0.9, 0.92]$, $(0.8, 0.9]$, $(0.76, 0.8]$, $(0.6, 0.76]$, and $(0, 0.6]$. Among these intervals, only $(0, 0.6]$ can be further decomposed. According to the subdivision rule, the k-*probs* of all vertices in the subgraph corresponding to $(0, 0.6]$ is calculated, and we obtain 0.5 and 0.52 located in $(0, 0.6]$. After that, the interval $(0, 0.6]$ can be divided into three sub-intervals $(0, 0.5]$, $(0.5, 0.52]$, and $(0.52, 0.6]$. Likewise, the subinterval $(0, 0.5]$ can be further divided, and three new subintervals, $(0, 0.2]$, $(0.2, 0.4]$, and $(0.4, 0.5]$, are obtained.

Time Complexity. The construction of ICU-Index involves two parts, partition of intervals and calculation of the corresponding communities of the intervals. For a given k, the upper bound of the number of intervals is $O(2n)$, where n refers to the number of vertices in $\mathcal{G}$. The time complexity of calculating the corresponding communities of each interval is the same as that of Algorithm 2. In the worst case, it costs

$O(k{\cdot}m)$ time. Therefore, the time complexity of index construction is $O(k_{max}^2 \cdot n \cdot m)$, where k_{max} is the largest k that enables the k-core of $\mathcal{G}$ not to be empty.

Space Complexity. For a given k, suppose the probability threshold range $(0, \eta_{max}]$ is divided into t intervals. And the size of the communities in each interval is linear to the size of its corresponding (k, η)-core. In the worst case, for each node v in an interval, it can be stored at most d_v times, where d_v is the degree of v. Therefore, the size of communities in each interval is at most $O(\sum_{v \in V} d_v) = O(m)$, where m is the number of edges. Therefore, the space complexity of the ICU-Index is $O(k_{max} \cdot t \cdot m)$.

The size of t depends on the probability distribution of edges and the weight distribution of vertices. Generally, we have t and $k_{max} \ll n$. In the worst case, the number of intervals (i.e., t) is related to the accuracy of η. That is, if η takes δ decimal place, there can be at most 10^δ intervals. For example, given a graph $\mathcal{G}$, suppose that the value of η is accurate up to 3 decimal places, then there will be at most 10^3 intervals for $\mathcal{G}$. Note that even for small-scale graphs (e.g., Flickr in Section 8), the number of nodes is much greater than that of intervals (e.g., $10^5 \gg 10^3$).

5.3 The Query Algorithm

In this subsection, we present the ICU-Index query algorithm for the (k, η)-influential community search.

Algorithm 4. ICU-Index Query

Input: An integer k, a probability threshold η, and the ICU-Index of $\mathcal{G}$.
Output: (k, η)-influential communities of $\mathcal{G}$.
1 $Ind_k \leftarrow$ the ICU-Index of $\mathcal{G}$ under k;
2 Initialize an influential community set $Res \leftarrow \varnothing$;
3 **foreach** *interval Invl in* Ind_k **do**
4 **if** $\eta \in Invl$ **then**
5 Append the communities of $Invl$ to Res;
6 **return** Res

As depicted in Algorithm 4, it takes an uncertain graph $\mathcal{G}$, the ICU-Index of $\mathcal{G}$, two parameters k and η as inputs. It first initializes Ind_k as the ICU-Index of $\mathcal{G}$ under k (Line 1). Then, Algorithm 4 initializes an influential community set Res to store all the final results (Line 2). For each interval Invl of Ind_k, we verify whether it contains η (Lines 3-5). If η is located in the interval, all the corresponding communities will be added to Res as final results (Lines 4-5).

Theorem 3. *Given arbitrary k and η, Algorithm 4 correctly retrieves all the (k, η)-influential communities of $\mathcal{G}$ according to ICU-Index.*

Proof. To prove the correctness of Algorithm 4, we first prove that the ICU-Index generated by Algorithm 3 covers all the influential communities of $\mathcal{G}$. Note that Algorithm 3 generates the ICU-Index for all possible k, i.e., $k \in [1, k_{max}]$. For a given k, due to the nested property of (k, η)-core, all possible η can be divided into several intervals, and η in the same interval corresponds to the same communities. Therefore, decomposing the value range of η under each k into intervals can cover all possible η and obtain corresponding communities. This means the ICU-Index generated by Algorithm 3 covers all possible influential communities.

For the specified k and η, Algorithm 4 retrieves all intervals containing η under k to obtain all the corresponding (k, η)-influential communities. Therefore, the theorem holds. □

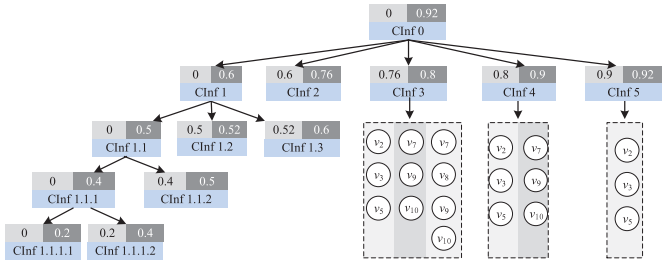


Fig. 4. Forest-based Index of $\mathcal{G}$ $(k = 1)$.

Time Complexity. As long as η is located at the right intervals, it takes $O(1)$ time to obtain all the corresponding communities. Thus, the time complexity of the ICU-based query processing is $O(t)$, where t is the number of all the intervals.

6 Optimization

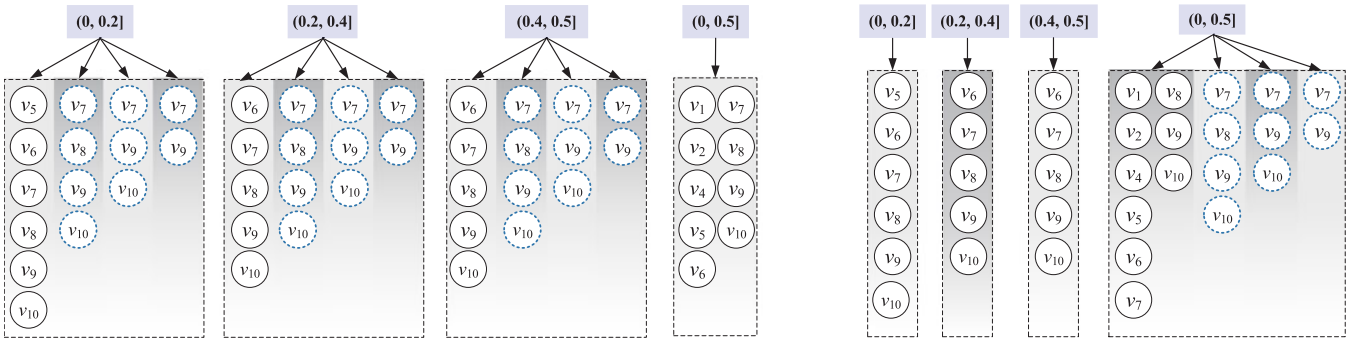
The proposed ICU-Index is efficient in improving the query performance of the (k, η)-influential community. This is because it avoids numerous repeated calculation of the k-probabilities for all the vertices in the given graph. On the contrary, the ICU-Index stores all the intervals and their corresponding communities, which takes up a large amount of space. Besides, due to the inclusion relationship between intervals, it is inefficient to traverse all the intervals to obtain the results. As a consequence, it is not appropriate to utilize the ICU-Index for processing large-scale graphs.

To alleviate this issue, two kinds of optimization strategies are proposed to further improve the ICU-Index. We utilize a forest-based index structure which helps boost query performance. In order to decrease the space cost of the ICU-Index, two strategies are proposed to avoid redundant storage of communities and vertices, respectively.

6.1 Forest-Based Index Structure

Based on the ICU-Index, for a (k, η)-influential community search, it needs to traverse all the keys in the index and return the intervals which the probability threshold η belongs to. To further reduce the search space, we develop a new forest-based index, namely forest-based influential community over uncertain graphs index (FICU-Index). All the intervals and their corresponding communities are organized as a tree structure and all the trees corresponding to k constitute a forest.

Fig. 4 shows the FICU-Index of $\mathcal{G}$ in Fig. 1 with $k = 1$. The root node of the index contains the initial interval $(0, \eta_{max}]$, and all the subintervals gained through decomposition are contained in its child nodes. Specifically, in Algorithm 3, the intervals obtained in the ith round are stored in the nodes of layer i of the tree, in which the decomposed intervals are in the parent nodes, and the decomposed subintervals are in the child nodes. The final intervals that cannot be decomposed are in the leaf nodes of the tree. Thus, for a



(a) communities before combination (b) communities after combination

Fig. 5. An example of community combination of ICU-Index of $\mathcal{G}$ $(k = 1)$.

(k, η)-influential community search, we can identify leaf nodes whose intervals contain the probability threshold η, and the communities under this interval can be returned as final results. Furthermore, all the ancestor nodes of these leaf nodes are computed in turn and the communities stored in these ancestor nodes are the final results of the (k, η)-influential community search.

The construction of FICU-Index is similar to Algorithm 3, except for the difference that the original interval is stored in the root node. In the segmentation procedure of Algorithm 3, the segmented interval is stored in the parent node, and the sub-intervals are stored in its child nodes.

It is noted that the main difference between ICU-Index and FICU-Index is the way of organizing intervals, which will not affect the accuracy of query results. Therefore, we can retrieve all communities given arbitrary k and η through FICU-Index according to Theorem 3. Based on FICU-Index, given a probability threshold η, a straightforward query approach is first used to search an interval containing η from the root node and then traverse the entire tree. Its time complexity is $O(t)$, where t is the number of intervals. To improve query efficiency, we adopt a bottom-up query technique. Particularly, for a (k, η)-influential community search, this approach stores all the leaf nodes of the tree in an array. The leaf node whose interval contains η is found by the binary search technique. The communities stored in this leaf node are added to the result set. Meanwhile, all the ancestors of the leaf node are selected and the communities stored in them are added to the final result.

Time Complexity. Regarding the FICU-Index query algorithm, the query time is linear to the depth of a given interval in the tree. In the worst case, i.e., FICU-Index is totally unbalanced, and the query based on FICU-Index degenerates to sequential search. In this scenario, the time complexity of FICU-Index query is O(t), where t is the total number of all the intervals. And if FICU-Index is balanced, it costs O $(\log(t))$ time to query influential communities.

Example 7. Given $k = 1$, Fig. 4 shows the forest-based index of $\mathcal{G}$. This index organizes all the intervals and their associated communities in Fig. 3 as a tree structure. That is, each node stores an interval and its associated corresponding communities. For instance, the node of interval $(0.8, 0.9]$ stores communities $COM_1 = \{v_2, v_3, v_5\}$ and $COM_2 = \{v_7, v_9, v_{10}\}$. Given a probability $\eta \in (0.8, 0.9]$, the two communities COM_1 and COM_2 can be returned as final results directly. Besides, given $k = 1$ and $\eta = 0.53$, we first find the interval of leaf nodes that containing 0.53 in the ICU-Index, that is, (0.52,0.6], and return its communities. We continue to search the parent node of (0.52,0.6] up to the root node and return the corresponding communities. Finally, $\{v_7, v_9, v_{10}\}$, $\{v_7, v_8, v_9, v_{10}\}$, $\{v_2, v_5, v_6, v_7, v_8, v_9, v_{10}\}$, and $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$ are returned as final results.

6.2 Space Occupancy Optimization

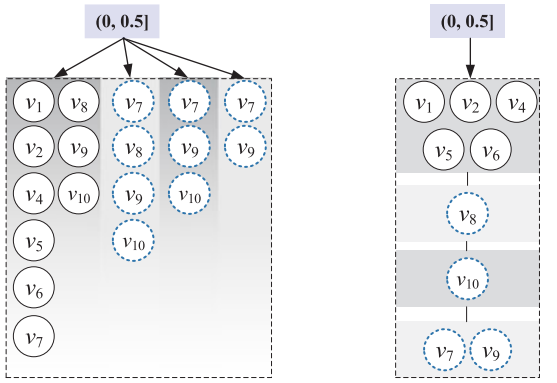
FICU-Index still bears much redundant space cost. In this subsection, we thus optimize the index by reducing redundant communities and vertices.

6.2.1 Community Combination

As shown in Fig. 5a, a community may be stored in different intervals of the index for many times. For instance, under the intervals $(0, 0.2]$, $(0.2, 0.4]$, and $(0.4, 0.5]$, there are three identical communities that are $\{v_7, v_8, v_9, v_{10}\}$, $\{v_7, v_9, v_{10}\}$, and $\{v_7, v_9\}$. To this end, we propose a community combination strategy to reduce the redundant space costs.

The main idea of the strategy is to merge the same communities under each interval and try to avoid storing these communities repeatedly. For an interval, we merge the communities that appear in all of its subintervals. Each subinterval is processed in sequence during the index construction. This means that we cannot identify duplicated communities of subintervals before the whole index is constructed. Therefore, we merge the duplicated communities in a bottom-up manner after the construction of FICU-Index. First, leaf nodes are taken into account. The communities contained in the leaf nodes with the same parent are moved to the parent node. For non-leaf nodes with the same parent, the same communities in these nodes are also moved to their parent node. Through this merging operator, it significantly reduces the size of the index, and decreases the space cost accordingly. It is worth noting that for an interval without a unique community, the vertex of this interval is kept and marked as empty to ensure the integrity of leaf nodes so that for any η there is a leaf node containing it.

Example 8. Fig. 5 shows the compression process of the ICU-Index in Fig. 2. Three communities $COM_1 = \{v_7, v_8, v_9,$



(a) communities before compression

(b) communities after compression

Fig. 6. An example of community compression of interval $(0, 0.5]$ of ICU-Index in Fig. 5.

$v_{10}\}$, $COM_2 = \{v_7, v_9, v_{10}\}$, and $COM_3 = \{v_7, v_9\}$ are stored in all the nodes corresponding to the intervals $(0, 0.2]$, $(0.2, 0.4]$, and $(0.4, 0.5]$. As shown in Fig. 5, these intervals are subintervals of the interval $(0, 0.5]$. Accordingly, it only needs to store these communities COM_1, COM_2, and COM_3 in the node of the interval $(0, 0.5]$. With this space compression strategy, the space overhead of the ICU-Index can be drastically reduced.

6.2.2 Community Compression

In FICU-index, each interval stores all the corresponding communities. As shown in Fig. 5b, the communities under interval (0,0.5] have inclusion relationships. That is, vertices of one community are proper subsets of another community, which means that many vertices are repeatedly stored in these subgraphs. This results in excessive space redundancy. Considering this, we propose a method to compress communities and alleviate the issue. Specifically, instead of storing all the communities corresponding to each interval of FICU-Index, we reorganize the communities with inclusion relationships as a tree. That is, for two communities C_1 and C_2, if C_2 contains C_1, C_1 is taken as the child of C_2, and C_2 retains the vertices that do not exist in C_1. In this way, the vertices of communities with inclusion relationships are stored only once, which can significantly reduce the space consumption of the index.

Example 9. Consider the ICU-Index in Fig. 5. Given $k = 1$, the communities of interval (0, 0.5] after community combination are $\{v_1, v_2, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$, $\{v_7, v_8, v_9, v_{10}\}$, $\{v_7, v_9, v_{10}\}$, and $\{v_8, v_9\}$, as shown in Fig. 6a. Note that there is a containment relationship between these communities, so these communities can be reorganized as a tree. As shown in Fig. 6b, the root node of the tree is $\{v_1, v_2, v_4, v_5, v_6\}$ and $\{v_7, v_9\}$ is a leaf node. In this way, we can reduce the space occupancy of the communities stored in each interval.

In Algorithm 5, for each interval, all the corresponding communities in the non-decreasing order of influence are organized as contained trees according to their inclusion relationships (Lines 3–4). This operation can be effectively implemented by using union-find, and each tree node only stores the community vertices that do not exist in its descendant nodes. In these contained trees, any two trees are merged if they have the same root (Line 5).

Algorithm 5. Community Compression

Input: The FICU-Index of the graph $\mathcal{G}$ after community combination and an interge k.
Output: The compressed FICU-Index.
1 $Ind_k \leftarrow$ the FICU-Index of $\mathcal{G}$ under k;
2 **foreach** $\mathrm{Ind}_k.Invl$ **do**
3 **foreach** *communities in non-descending order of influence* **do**
4 Construct contained trees `// each tree node retains its unique community vertices`
5 Merge two trees if they have the same root;

Therefore, for each contained tree, the communities stored in leaf nodes are non-contained influential communities [7]. To query communities with significant influence, we only need to traverse the tree from leaf nodes in a bottom-up manner until achieving desired results. The time complexity of querying all communities from the contained tree is linear to the result size [7]. In the worst case, the time complexity is $O(n)$.

Space Complexity. For a given k, there is at most t intervals. In the worst case, the size of the communities in each interval is $O(n)$. Hence, the space complexity of the FICU-Index is $O(k_{max} \cdot t \cdot n)$.

7 Index Maintenance

Maintaining the FICU-index is a very challenging problem. Due to the influence of edge probability, it is quite different from maintaining k-influential community. Specifically, if edges change in k-core, the existing information can be used to determine whether the core number of the affected vertices needs to be updated. However, the situation may be different in uncertain graphs. Because of the edge probability, the insertion or deletion of an edge affects the *k-prob* of the vertex and further affects the calculation of the influential community. This is because inserted or deleted edges may change the removal order of vertices during decomposition, thereby influencing the final results. As a result, edge updates tend to incur a prohibitively large number of vertices in FICU-Index needing to be updated in turn, which is time-consuming.

In this section, we develop a heuristic approach that avoids recalculating the entire index, which helps to enhance the performance of index maintenance. The mechanism of this method is to update the communities where the inserted or deleted edges are located instead of the entire graph.

Generally, there are two update operators, including the insertion and deletion of edges. Consider the edge insertion operator. To find the affected subgraph, we update the η-thresholds of the vertices on both sides of the insert edge and calculate the (k, η)-core containing the affected vertices with the largest η-threshold. Then we decompose it into (k, η)-influential communities. Specifically, after inserting an edge (u, v), the η-thresholds of u and v are refreshed as η_u and η_v, respectively, and we have $\eta_m = max\{\eta_u, \eta_v\}$. In the original index, those communities under the intervals whose low bounds are greater than η_m cannot be affected

by this update operator. Therefore, only a part of the original FICU-Index requires to be recalculated. Moreover, in these affected parts of the FICU-Index, it is unnecessary to update the communities that are disconnected with u and v. That is to say, the insertion of the edge (u, v) can only affect the connected subgraphs containing u or v.

From the above analysis, to update an edge (u, v), the heuristic approach primarily contains three steps, including calculating initial communities containing vertices u, v and their corresponding intervals according to the updated η-threshold of u and v, decomposing these initial communities, as well as finding and updating corresponding parts of the FICU-Index affected by the edge update. Similarly, the edge deletion operator can be processed in the same way.

8 Experiments

This section offers a detailed analysis of the extensive experiments we have conducted to evaluate our proposed solutions. Section 8.1 introduces the setup of the experiment, while Section 8.2 presents experimental results.

8.1 Setup

We evaluate the following five algorithms.

- *Online*: the online algorithm depicted in Section 4 without an index;
- *ICU-based Query*: the ICU-Index query algorithm depicted in Section 5;
- *FICU-based Query*: the FICU-Index query algorithm depicted in Section 6.
- *ICU-construct*: the ICU-Index construct algorithm depicted in Section 5;
- *FICU-construct*: the FICU-index construct algorithm with optimizations in Section 6.

Existing methods of influential community search are all for deterministic graphs, which are not suitable for calculating and updating k-*probs*. Therefore, Online is used as the baseline to compare with index-based algorithms. The experimental results show that the index-based methods are at least two orders of magnitude faster than the baseline. At the same time, we have also compared the construction time and space consumption of the two indexes. The construction time of the two indexes is similar, but the space consumption of the FICU-Index is less than half of that of the ICU-Index. Furthermore, we have compared the influential community model over deterministic graphs with the model proposed in this paper and analyzed the differences between the two.

Data Sets. Six data sets including Flickr, Human, DBLP, YouTube, WikiTalk, and LiveJournal are utilized to evaluate the above algorithms. Flickr[1] is a platform offered by Yahoo for providing photo storage, solution sharing and online community services. The probabilities of edges represent the jacquard coefficient of the groups of two users [38]. Human is a biological interactive network of BioMine organism set which is a database of the project BioMine[2]. The probability of an edge represents the confidence of the interaction between vertices [16], [39]. DBLP is a database system that provides open bibliographic information for major computer science journals and papers. In this dataset, vertices represent different authors, and edges between vertices represent the co-relationship between two authors. The probability of each edge is calculated by $1-e^{-x/2}$, where x is the number of collaborative two-authored papers [38], [40]. YouTube is a video website that users can download, watch and share videos or short films. Wiki-Talk is a network that includes all users and discussions from the time when Wikipedia was founded until January of 2008. LiveJournal is a comprehensive SNS dating website integrating functions of forum, blog, etc. These networks are chosen from the Stanford Network Analysis Platform[3]. For these datasets we have generated probability values uniformly distributed in (0, 1]. And the probabilities of all the edges take three significant digits.

TABLE 2
Real Data Sets

Graph	vertices	Edges	d_{max}	k_{max}
Flickr	24,125	300,836	546	225
Human	27,762	1,570,472	62,580	145
DBLP	636,751	2,366,461	446	118
YouTube	1,134,890	2,987,624	28,754	51
WikiTalk	2,394,385	4,659,565	10,029	131
LiveJournal	3,997,962	34,681,189	14,815	360

TABLE 3
Experimental Parameters

Parameter	Values	Default Value
k	5, 10, 15, 20, 25	15
η	0.1, 0.3, 0.5, 0.7, 0.9	0.5

Table 2 shows the details of the data sets. Some other statistical information is presented except for the total number of vertices and edges in data sets, where d_{max} is the maximum degree of all vertices in a graph, and k_{max} is the largest k that enables the k-core to be non-empty in a graph. Besides, the PageRank of each vertex is used as its weight, and the damping factor of PageRank is set to 0.85.

All the graphs are stored in memory before the algorithms are implemented in Python, version 3.8 (64 bit). The experiments are conducted on a computer with 2.8GHz Intel Core i5 CPU and 32 GB memory.

8.2 Evaluation

Experimental Parameters. The (k, η)-influential community query has two parameters k and η. The probability threshold η is varies from 0.1 to 0.9 with a step size of 0.2; k is an integer obtained from $\{5, 10, 15, 20, 25\}$. As shown in Table 3, the default of k and η are 15 and 0.5, respectively.

Exp-1: Evaluation of the Query Time With Different k. In this testing, we have evaluated the efficiency of the Online search, ICU-based query and FICU-based query algorithms with k varying. Fig. 7 shows the running time of the three

1. https://www.flickr.com/
2. https://www.cs.helsinki.fi/group/biomine/
3. http://snap.stanford.edu/

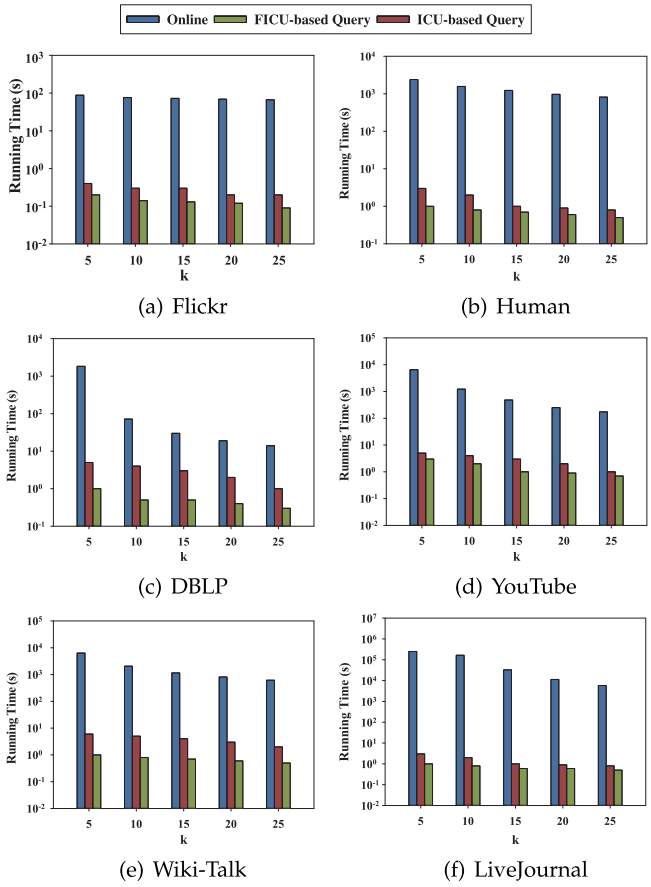


Fig. 7. Query time with varying k ($\eta = 0.5$).

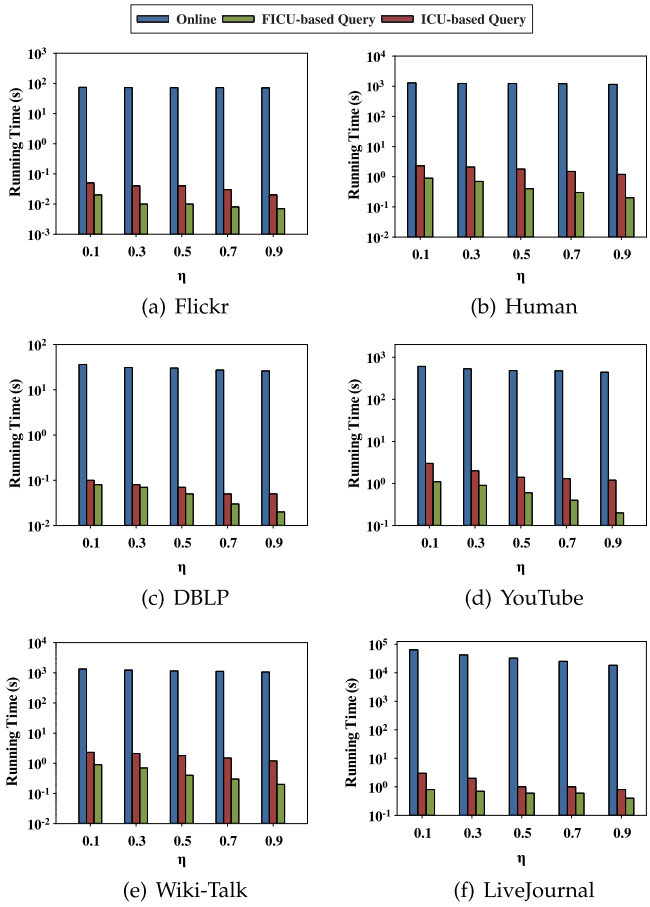


Fig. 8. Query time with varying η ($k = 15$).

algorithms on six datasets. Over all data sets and different k values, FICU-Index runs much faster than Online, while ICU-Index spends more time between them. It is noted that the gap between Online and index-based approaches increases with the decrease of k. This is because the smaller the k is, the more vertices satisfy the (k, η)-core constraint, which generates more time on the part of the Online method for calculating and updating *k-probs* of vertices.

Exp-2: Evaluation of the Query Time With Different η. In this testing, we have evaluated the efficiency of Online search, ICU-based query and FICU-based query algorithms with η varying. The running time of three algorithms on six datasets is shown in Fig. 8. With the increase of η, the running time of all algorithms decreases. The reason is that the increase of η makes the fewer nodes satisfy the constraint that *k-prob* is not less than η, and the calculation time amount of the Online search will be smaller. However, the trend of index-based methods is not so obvious, and the FICU-Index has the highest efficiency. Meanwhile, the query time of the FICU-Index is less than that of the ICU-Index, and the difference of query time between FICU-Index and ICU-Index with different η is larger than that with different k values. This indicates that the index-based method is more sensitive to η because, for FICU-Index, different values of η corresponding to vertices with different depths.

Exp-3: Evaluation of the Construction Time and Space Cost on Different Data Sets. In this testing, we have evaluated the index construction time and space cost of the two proposed indexes on different datasets. Here, ICU-construct denotes the algorithm for constructing ICU-Index and FICU-construct denotes the algorithm for constructing FICU-Index with the optimizations proposed in Section 6.

Fig. 9a shows the running time of two construction algorithms on six datasets. Compared to ICU-construct, FICU-construct spends more time constructing index because FICU-Index has extra forest structure. However, since the search tree is built during the decomposition of (k, η)-influential communities, the construction time difference between ICU-Index and FICU-Index is less than 10%.

Fig. 9b shows the index size on six datasets. The scale of the index increases with the growth of data scale, especially the number of edges in networks. Because the more edges there are, the more new *k-probs* will be newly generated in the process of community decomposition, which results in many new intervals and communities. Therefore, the space occupation of the index increases significantly. On the six datasets, the size of FICU-Index is smaller than that of ICU-

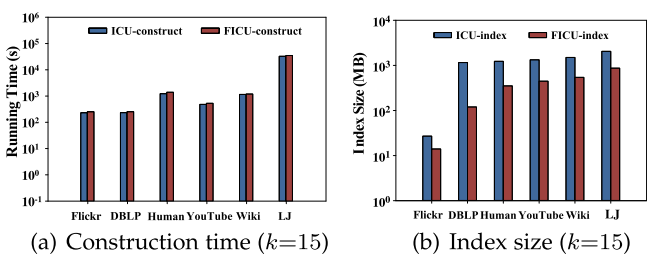
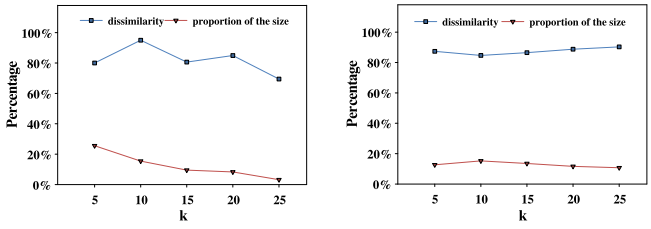
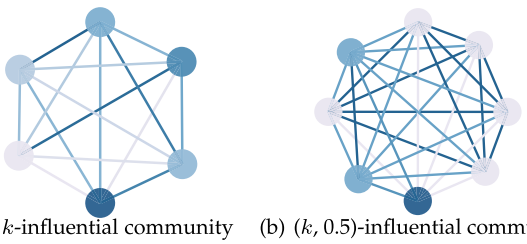


Fig. 9. Index construction time and space cost on different data sets.



(a) Flickr (b) DBLP

Fig. 10. The differences between deterministic and uncertain influential models under different k (η=0.5).



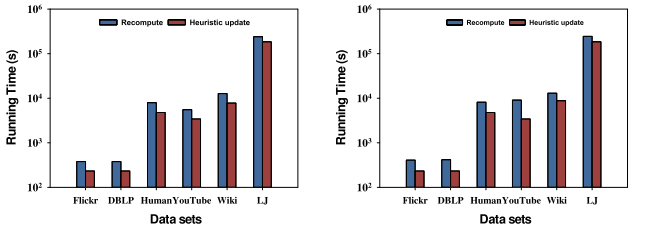
(a) k-influential community (b) $(k, 0.5)$-influential community

Fig. 11. The most influential community in Flickr under different influential models (k=5).

Index. For example, in the worst case (LiveJournal), ICU-Index takes up about 1.41GB memory, while FICU-Index uses roughly 830MB memory. The usage of memory is reduced by nearly 41%. This is because the proposed optimization strategies refine the communities and vertices that are repeatedly stored in ICU-Index. Thus, the construction time of the two indexes is similar, but FICU-Index significantly reduces the space consumption of the index.

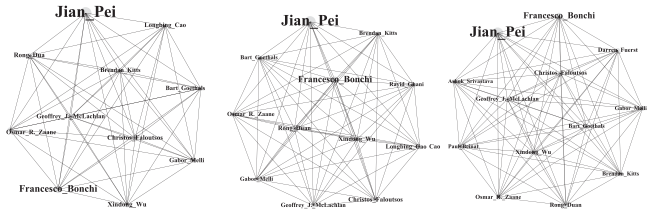
Exp-4: Influential Community Model Comparison. In this testing, we have evaluated the difference between the (k, η)-influential community model and the k-influential community model on two real datasets, Flickr and DBLP. We first evaluated the dissimilarity based on the Jaccard distance of the vertices of the top-10 communities obtained by the two models. The Jaccard distance between two communities X and Y is defined as $1 - \frac{X \cap Y}{X \cup Y}$. The larger the value, the greater the difference. We also compared the size of the top-r communities of two models through the size ratio of (k, η)-influential community and k-influential community under different k. The smaller the ratio, the larger the scale gap between them. Fig. 10 shows there is less overlap between them. Meanwhile, the communities obtained by the (k, η)-influential community model are much smaller than that of the k-influential community model. The reason lies in that uncertain graphs introduce the probability constraint of the relationship between vertices, and the members of (k, η)-influential communities are connected more closely.

Besides, we have compared the most influential communities in Flickr to further illustrate the characteristics of the two influential community models. Fig. 11 shows the most influential communities in Flickr under different influential community models with $k = 5$ and $\eta = 0.5$. Among them, the darker the color of the vertex (edge), the larger the weight (probability). It can be seen from the results that compared with the k-influential community, (k, η)-influential community model has a tighter connection, although the influence of the community is smaller. The main reason



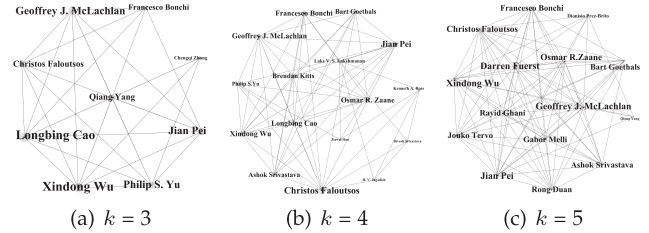
(a) Edge deletion (m=1000) (b) Edge insertion (m=1000)

Fig. 12. Dynamic update of FICU-Index (k=5).



(a) $\eta = 0.3$ (b) $\eta = 0.4$ (c) $\eta = 0.5$

Fig. 13. Top-1 (k, η)-influential community of DBLP (k=5).



(a) $k = 3$ (b) $k = 4$ (c) $k = 5$

Fig. 14. Top-1 (k, η)-influential community of DBLP (η=0.5).

is that (k, η)-influential community needs to add some vertices with smaller weights to maintain a higher k-*prob* and ultimately satisfy the constraint of η.

Exp-5: Index Maintenance. In this testing, we have evaluated the running time of our proposed index maintenance approaches. `Recompute` denotes the approach that uses the FICU-construct algorithm to recompute the entire index when graphs are updated. `Heuristic update` is the heuristic index update approach proposed in Section 7. Following the work in [7], to simulate the graph update, we randomly deleted 1000 edges from each dataset and then inserted them to restore the original dataset. In the process of deletion and insertion, we ran the index maintenance algorithms respectively. The running time of the algorithms is shown in Fig. 12. For all vertices affected by edge insertion/deletion, we need to re-decompose the (k, η)-cores containing them. On LiveJournal, `Heuristic update` saves less than 20% of the running time compared with `Recompute`, while on other data sets, it saves more than 30% of the running time.

Exp-6: Case Study. In this testing, we have conducted a case study on the DBLP data set where the weight of each vertex represents its PageRank. Two famous researchers in data management and graph computing, Jian Pei and Francesco Bonchi, are referenced here. Fig. 13 shows the (k, η)-influential communities containing Jian Pei with $k = 5$ and the probabilistic threshold η growing from 0.3 to 0.5. Moreover, Fig. 14 illustrates the (k, η)-influential communities of Francesco Bonchi with $\eta = 0.5$ and k varying from 3 to 5. As shown, the sizes of communities increase with the

growth of k, while the influence of communities represented by the font size of vertices decreases. Again, as η grows, the sizes of communities increase. This is in line with our expectation, since it requires adding more vertices for the communities to satisfy the probabilistic threshold constraint. From the above analysis, in practice, it needs an appropriate value of k and η for gaining expected communities. For example, if users want a dense community, then a large k and η are needed. Meanwhile, choosing a large η when k is small may result in no community that meets the (k, η)-influential community conditions.

9 Conclusion

In this paper, we have examined the influential community search problem over uncertain graphs for the first time. We have formulated the (k, η)-influential community model in uncertain graphs, and proposed an online algorithm without an index. We have designed an ICU-Index to search influential community in uncertain graphs, which achieves improved query efficiency. Moreover, two kinds of optimization strategies have been proposed for ICU-Index and FICU-Index, which can speed up query processing and significantly reduce space occupation. Through extensive experiments on real datasets , we have successfully demonstrated the efficiency and effectiveness of our approaches. It is also an interesting work of computing top-r influential communities. Although the proposed algorithms in this paper can resolve this problem to some extent, it has to be acknowledged that the performance needs to be improved. Hence, in our future work, we will seek to explore efficient pruning strategies and progressive algorithms to boost the query performance of the top-r influential community search over uncertain graphs.

Acknowledgments

The authors would like to thank the anonymous reviewers for their careful guidance of the paper.

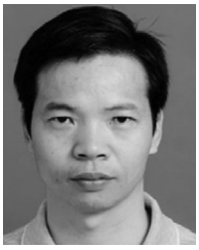
References

[1] S. B. Seidman, "Network structure and minimum degree," *Social Netw.*, vol. 5, no. 3, pp. 269–287, 1983.
[2] Y. Fang, R. Cheng, X. Li, S. Luo, and J. Hu, "Effective community search over large spatial graphs," *Proc. VLDB Endowment*, vol. 10, no. 6, pp. 709–720, 2017.
[3] Y. Fang, Y. Yang, W. Zhang, X. Lin, and X. Cao, "Effective and efficient community search over large heterogeneous information networks," *Proc. VLDB Endowment*, vol. 13, no. 6, pp. 854–857, 2020.
[4] J. Cohen, "Trusses: Cohesive subgraphs for social network analysis," *Nat. Secur. Agency, Tech. Rep.*, Citeseer, vol. 16, no. 3.1, 2008.
[5] Q. Liu, M. Zhao, X. Huang, J. Xu, and Y. Gao, "Truss-based community search over large directed graphs," in *Proc. SIGMOD Int. Conf. Manage. Data.*, 2020, pp. 2183–2197.
[6] L. Yuan, L. Qin, W. Zhang, L. Chang, and J. Yang, "Index-based densest clique percolation community search in networks," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 5, pp. 922–935, May 2018.
[7] R. H. Li, L. Qin, J. X. Yu, and R. Mao, "Influential community search in large networks," *Proc. VLDB Endowment*, vol. 8, no. 5, pp. 509–520, 2015.
[8] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 12, pp. 7821–7826, 2002.
[9] S. Brohee and J. Van Helden, "Evaluation of clustering algorithms for protein-protein interaction networks," *BMC Bioinf.*, vol. 7, no. 1, 2006, Art. no. 488.
[10] J. Li, X. Wang, K. Deng, X. Yang, T. Sellis, and J. X. Yu, "Most influential community search over large social networks," in *Proc. Int. Conf. Data Eng.*, 2017, pp. 871–882.
[11] F. Bi, L. Chang, X. Lin, and W. Zhang, "An optimal and progressive approach to online search of top-k influential communities," *Proc. VLDB Endowment*, vol. 11, no. 9, pp. 1056–1068, 2018.
[12] W. Luo, X. Zhou, J. Yang, P. Peng, G. Xiao, and Y. Gao, "Efficient approaches to top-r influential community search," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12 650–12 657, Aug. 2021.
[13] J. Ugander, L. Backstrom, C. Marlow, and J. Kleinberg, "Structural diversity in social contagion," *Proc. Nat. Acad. Sci. USA*, vol. 109, no. 16, pp. 5962–5966.
[14] R. Li, Q. Dai, G. Wang, Z. Ming, L. Qin, and J. X. Yu, "Improved algorithms for maximal clique search in uncertain networks," in *Proc. Int. Conf. Data Eng.*, 2019, pp. 1178–1189.
[15] X. Li, W. Min, C. K. Kwoh, and S. K. Ng, "Computational approaches for detecting protein complexes from protein interaction networks: a survey," *BMC Genomic.*, vol. 11, no. Suppl 1, pp. 1–19, 2010.
[16] Y. Gao, X. Miao, G. Chen, B. Zheng, D. Cai, and H. Cui, "On efficiently finding reverse k-nearest neighbors over uncertain graphs," *Int. J. Very Large Data Bases*, vol. 26, no. 4, pp. 467–492, 2017.
[17] Y. Qiu *et al.*, "Efficient structural clustering on probabilistic graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 10, pp. 1954–1968, Oct. 2019.
[18] T. Wolf, A. Schroter, D. Damian, L. Panjer, and T. H. D. Nguyen, "Mining task-based social networks to explore collaboration in software teams," *IEEE Softw.*, vol. 26, no. 1, pp. 58–66, Jan./Feb. 2009.
[19] L. Antiqueira, O. N. O. Jr, L. da Fontoura Costa, and M. das Graças Volpe Nunes, "A complex network approach to text summarization," *Inf. Sci.*, vol. 179, no. 5, pp. 584–599, 2009.
[20] X. Zhou, K. Li, Y. Zhou, and K. Li, "Adaptive processing for distributed skyline queries over uncertain data," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 2, pp. 371–384, Feb. 2016.
[21] N. Korovaiko and A. Thomo, "Trust prediction from user-item ratings," *Social Netw. Anal. Mining*, vol. 3, no. 3, pp. 749–759, 2013.
[22] M. T. Dittrich, G. W. Klau, A. Rosenwald, T. Dandekar, and T. Müller, "Identifying functional modules in protein–protein interaction networks: an integrated exact approach," *Bioinformatics*, vol. 24, no. 13, pp. i223–i231, 2008.
[23] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Learning influence probabilities in social networks," in *Proc. 3rd ACM Int. Conf. Web Search Data Mining*, 2010, pp. 241–250.
[24] F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich, "Core decomposition of uncertain graphs," in *Proc. 20th SIGKDD Int. Conf. Knowl. Discov. Data Mining.*, 2014, pp. 1316–1325.
[25] Y. Peng, Y. Zhang, W. Zhang, X. Lin, and L. Qin, "Efficient probabilistic K-core computation on uncertain graphs," in *Proc. IEEE 34th Int. Conf. Data Eng.*, 2018, pp. 1192–1203.
[26] N. Barbieri, F. Bonchi, E. Galimberti, and F. Gullo, "Efficient and effective community search," *Data Mining Knowl. Discov.*, vol. 29, no. 5, pp. 1406–1433, 2015.
[27] V. Batagelj and M. Zaversnik, "An O (m) algorithm for cores decomposition of networks," 2003, *arXiv:cs/0310049*.
[28] S. Chen, R. Wei, D. Popova, and A. Thomo, "Efficient computation of importance based communities in web-scale networks using a single machine," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, 2016, pp. 1553–1562.
[29] A. E. Sariyüce and A. Pinar, "Fast hierarchy construction for dense subgraphs," *Proc. VLDB Endowment*, vol. 10, no. 3, pp. 97–108, 2016.
[30] Y. Fang *et al.*, "A survey of community search over big graphs," *Proc. VLDB Endowment J.*, vol. 29, no. 1, pp. 353–392, 2020.
[31] F. Esfahani, V. Srinivasan, A. Thomo, and K. Wu, "Efficient computation of probabilistic core decomposition at web-scale," in *Proc. Int. Conf. Extending Database Technol.*, 2019, pp. 325–336.
[32] D. Wen, B. Yang, L. Qin, Y. Zhang, L. Chang, and R. Li, "Computing k-cores in large uncertain graphs: An index-based optimal approach," *IEEE Trans. Knowl. Data Eng.*, early access, Sep. 16, 2020, doi: 10.1109/TKDE.2020.3023925.
[33] M. Charikar, "Greedy approximation algorithms for finding dense components in a graph," in *Proc. Int. Workshop Approximation Algorithms Combinatorial Optim.*, 2000, pp. 84–95.
[34] A. V. Goldberg, *Finding a Maximum Density Subgraph.* Berkeley, CA, USA: Univ. California, 1984.
[35] C. Ma, Y. Fang, R. Cheng, L. V. Lakshmanan, W. Zhang, and X. Lin, "Efficient algorithms for densest subgraph discovery on large directed graphs," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2020, pp. 1051–1066.

[36] L. Chang, J. X. Yu, L. Qin, X. Lin, C. Liu, and W. Liang, "Efficiently computing k-edge connected components via graph decomposition," in *Proc. SIGMOD Int. Conf. Manage. Data.*, 2013, pp. 205–216.
[37] R. Zhou, C. Liu, J. X. Yu, W. Liang, B. Chen, and J. Li, "Finding maximal K-edge-connected subgraphs from a large graph," in *Proc. 15th Int. Conf. Extending Database Technol.*, 2012, pp. 480–491.
[38] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios, "k-nearest neighbors in uncertain graphs," *Proc. VLDB Endowment*, vol. 3, no. 1, pp. 997–1008, 2010.
[39] V. Podpecan, Z. Ramsak, K. Gruden, H. Toivonen, and N. Lavrac, "Interactive exploration of heterogeneous biological networks with biomine explorer," *Bioinformatics*, vol. 35, no. 24, pp. 5385–5388, 2019.
[40] M. Ceccarello, C. Fantozzi, A. Pietracaprina, G. Pucci, and F. Vandin, "Clustering uncertain graphs," *Proc. VLDB Endowment*, vol. 11, no. 4, pp. 472–484, 2017.

Wensheng Luo is currently working toward the PhD degree in computer science and technology with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. His research interests include graph analysis, data management, and parallel computing.

Xu Zhou received the PhD degree in computer science and technology from Hunan University, China, in 2016. She is currently an associate professor with the College of Computer Science and Electronic Engineering, Hunan University. She has authored or coauthored more than 30 papers in international journals and conferences, including, the *IEEE Transactions on Knowledge and Data Engineering*, *IEEETransaction Parallel Distributed Systems*, and *IEEE Transactions on Computers*. Her research interests include parallel and distributed processing and database systems.



Kenli Li (Member, IEEE) received the PhD degree in computer science from the Huazhong University of Science and Technology, China, in 2003. He has authored or coauthored more than 200 research papers in international conferences and journals, including the *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed. Systems*, and ICPP. His research interests include parallel computing, high-performance computing, and grid and cloud computing. He is on the editorial board of the *IEEE Transactions on Computers*.

Yunjun Gao (Member, IEEE) received the PhD degree in computer science from Zhejiang University, China, in 2008. He is currently a professor with the College of Computer Science, Zhejiang University, China. His research interests include spatial and spatio-temporal databases, metric and incomplete or uncertain data management, and spatio-textual data processing. He is a member of the ACM and a senior member of the CCF.

Keqin Li (Fellow, IEEE) is currently a SUNY distinguished professor of computer science with the State University, New York and the national distinguished professor with Hunan University, China. He has authored or coauthored nearly 800 journal articles, book chapters, and refereed conference papers. His research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber-physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, and intelligent and soft computing. He holds more than 60 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top ten most influential scientists in distributed computing based on a composite indicator of Scopus citation database. He is currently an associate editor for the *ACM Computing Surveys* and *CCF Transactions on High Performance Computing*. He was on the editorial boards of *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Services Computing*, and *IEEE Transactions on Sustainable Computing*. He was the recipient of several best paper awards.