Efficient Probabilistic K-Core Computation on Uncertain Graphs

You Peng†, Ying Zhang§, Wenjie Zhang†, Xuemin Lin†, Lu Qin§

†*University Of New South Wales,* §*University of Technology Sydney*

unswpy@gmail.com,{zhangw,lxue}@cse.unsw.edu.au,{ying.zhang,lu.qin}@uts.edu.au

***Abstract*—As uncertainty is inherent in a wide spectrum of graph applications such as social network and brain network, it is highly demanded to re-visit classical graph problems in the context of uncertain graphs. Driven by real-applications, in this paper, we study the problem of k-core computation on uncertain graphs and propose a new model, namely (k,θ)-core, which consists of nodes with probability at least θ to be k-core member in the uncertain graph. We show the computation of (k,θ)-core is NP-hard, and hence resort to sampling based methods. Effective and efficient pruning techniques are proposed to significantly reduce the candidate size. To further reduce the cost of k-core computation on multiple sampled graphs, we design a k-core membership check algorithm following a novel expansion-based search paradigm. Extensive experiments on real-life graphs demonstrate the effectiveness and efficiency of our proposed techniques.**

I. Introduction

A k-core of a graph is the maximal induced subgraph in which every node has at least k neighbors, which has been used in a wide range of applications such as social contagion [1], influence spread [2] and graph clustering [3]. Existing works on k-core computation assume that each edge of the graph always exists, i.e., graph is *deterministic*. But in some real-life applications, we need to consider the occurrence probability of the edges, where a graph is modeled as an *uncertain graph*. For instance, when an edge is used to capture the *influence* between two people in social networks, it can be naturally represented by a probability value obtained by some heuristics or machine learning methods (e.g., [4]). As the significance of the *influences* between two users may vary from person to person, it is a crude approximation of reality by treating all influences (edges) as the same in existing work on k-core computation. This motivates us to develop a probabilistic k-core model as well as efficient algorithms to properly handle probability values (i.e., uncertainty) of the edges. Similarly, in Protein-Protein Interaction (PPI) networks [5], an edge represents the interaction between two proteins associated with an occurrence probability provided by statistical predictions.

The *possible world semantics* has been widely applied to model a variety of problems on uncertain graph. Generally speaking, a possible world in the context of uncertain graph is an instance graph with a certain probability to occur. In this paper, we say a model is global if it tackles the problem on each individual instance graph and then aggregates the results. If the computation of each node is based on its current degree distribution (i.e., the local distribution of the instance graphs), we say a model is local. Note that the degree distribution of a node may be updated during the computation. We remark that two types of models capture the uncertainty of graph from different perspectives, and both are useful in a variety of applications.

To our best knowledge, there is only one existing work which addresses the problem of k-core computation on uncertain graphs. In [6], given the k value and a probabilistic threshold $\eta \in [0, 1]$, the k-core on uncertain graph, namely (k,η)-core, is computed in a similar way with k-core computing algorithm on deterministic graph. Particularly, a node is removed if, with probability less than η, the degree on its current degree distribution is no less than k. Garas *et al.* study the problem of k-core computation in the context of weighted graph [7]. Inspired by their study, we can have an *expected* k-core model on uncertain graph by treating the edge probability as weight, and details will be presented in Section II-D. Following exactly the same computing paradigm with [6], a node is removed if its current *expected* degree is smaller than k. Both (k, η)-core and expected k-core are local models as their computation is based on degree distributions.

In existing k-core applications, k-core members may be used together or individually. For instance, the k-core members can be used together to describe the Internet topology [8] or form a team [6]. While in some applications, it has been shown that the k-core membership of a node (user) can capture the engagement [9] and influence [1], [2] of the individual node, or serve as upper bound of other models (e.g., clique, k-truss and k-edge connected component [10]). In these applications, a node belonging to k-core with large k value usually *indicates* higher influence or engagement. In the context of uncertain graph, a natural extension of the k-core membership is to evaluate the likelihood of a node being k-core member in possible worlds (i.e., instance graphs). This motivates us to propose a *global* model, namely (k, θ)-core, to capture the k-core membership of the node such that we can find important nodes (hubs) among the possible k-core communities. Specifically, given an uncertain graph $\mathcal{G}$ with m edges, there are 2^m possible worlds and each possible world is a possible instance graph of the uncertain graph. Given k and probability threshold θ, we aim to find the nodes with probability at least θ to be included in the k-core in possible worlds [1].

We illustrate the inherent difference of *local* and *global* k-core models with two toy examples. Given an uncertain graph $\mathcal{G}$ as shown in Fig. 1(a), which is a ring with many

[1]Each instance graph in the possible world is a deterministic graph, and we have the deterministic k-core accordingly.

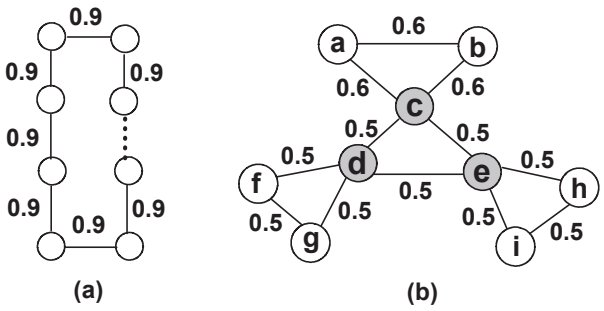


Fig. 1. Motivating examples for k-core models

nodes. Suppose the existence probability of each edge is 0.9 and the occurrence of an edge is independent to others, we can see that none of the nodes will be removed in (k,η)-core model for $k = 2$ and $\eta = 0.8$ because each individual node has the probability $0.9 * 0.9 = 0.81$ to have two neighbors; that is, every node has sufficient support regarding degree distributions. While in the global model, although the occurrence probability of each edge is high, there is a very small chance to come up with a 2-core in a possible world because the absence of *any* edge will trigger the collapse of the graph in 2-core computation, and all nodes will be eventually removed. With similar rationale, in Fig. 1(b) the (k, η)-core consists of nodes $S_1 = \{a, b, c\}$ for $k = 2$ and $\eta = 0.36$. While (k, θ)-core members are $S_2 = \{c, d, e\}$ for $k = 2$ and $\theta = 0.23$. Here we can see the different criteria to choose core-members for two models. Particularly, (k,η)-core pays more attention to the local degree distribution and direct support from k-core members. On the other hand, (k,θ)-core focuses on the k-core probability of each individual node, i.e., find hub nodes which frequently contribute to possible k-core communities. For instance, node d may contribute to k-core members regarding sets $\{d, c, e\}$ or $\{d, f, g\}$ and hence the accumulated probability is relatively high. It is worth noting that, as (k,θ)-core aims to find important (hub) nodes for the possible k-core communities, its core members retrieved may be loosely connected or even isolated to each other compared with local models. Thus, (k,θ)-core is more appropriate for applications where k-core members are used individually.

In Section II-D and II-E, we further discuss the difference of the models, and show that both models are useful in real-life applications. In particular, we remark that the proposed (k, θ)-core model is useful in the k-core membership related applications, especially the scenarios in which the possible world semantics is employed in a global way such as influence spread on independent cascade (IC) model [11] and uncertain clique computation [12].

Challenges and Contributions. In the paper, we show that the k-core computation on probabilistic k-core model is NP hard. A straightforward solution is to sample the possible worlds and compute the k-core in each possible world (i.e., sampled graph). Then the k-core probability of each node can be obtained according to the number of times that it is included by the k-cores in the sampled possible worlds (graphs). Although the computation of k-core in each possible world is efficient, we usually need a considerable number of samples for a decent theoretical accuracy guarantee. As such, the overall computational cost may be expensive. To tame the computational hardness of probabilistic k-core, we develop effective upper and lower bounds based on pruning techniques to significantly reduce the computational cost. To speed up the k-core computation on each sampled graph, we propose a novel expansion-search based algorithm to check the k-core membership of some nodes, which need verification after applying pruning techniques.

Notation	Definition
$\mathcal{G} = (V, E, p)$	uncertain graph with nodes V, edges E, edges probabilistic functions p
G, G_i	deterministic graph, instance graph, sample graph
s	number of sample graphs in sampling algorithms
G_0	deterministic graph of $\mathcal{G}$
$p(u)$, $p_k(u, \mathcal{G})$	k-core probability of u on uncertain graph $\mathcal{G}$
θ	probability threshold
$C_{k,\theta}$	(k,θ)-core for given k and θ
$\hat{p}(u)$	estimator of $p(u)$ based on sample graphs of $\mathcal{G}$
$p^+(u)$ ($\hat{p}^+(u)$)	upper bound of $p(u)$ ($\hat{p}(u)$)
$\hat{p}^-(u)$	lower bound of the estimator $\hat{p}(u)$
$u.stat$	the k-core status of a node u in *sample graph* { $\surd$, $\otimes$, ? }
$u.low(u.up)$	lower and upper bound of number of supports of u in k-core computation in *sample graph*
R	(k,θ)-core candidate nodes

TABLE I
THE SUMMARY OF NOTATIONS

Our principal contributions are summarized as follows.

- We proposed a new probabilistic k-core model for the uncertain graph, namely (k,θ)-core. We also show the difference between our proposed model and two local k-core models in terms of model analysis and empirical evaluation.
- We show the problem studied is NP-hard. Then we propose efficient sampling algorithms to compute the probabilistic k-core. Novel pruning techniques are designed to significantly reduce the candidate size. A novel k-core membership check algorithm is proposed to further speed up the computation.
- We conduct comprehensive experiments to evaluate the effectiveness and efficiency of our proposed techniques.

II. PRELIMINARIES

Table I summarizes the mathematical notations used throughout this paper.

A. Problem Definition

We first briefly recall the problem of k-core computation on deterministic/certain graphs (i.e., every edge has occurrence probability 1.0). Let $G = (V, E)$ be an undirected and unweighted deterministic graph, where V is the set of n nodes and $E \subseteq V \times V$ is the set of m edges. We use $deg(u, S)$ to denote the degree of a node u regarding a subgraph S. Below is the definition of k-core on deterministic graph.

*Definition 1 (**k-core**):* Given a graph G, a subgraph J is the k-core of G, denoted by $C_k(G)$, if (i) $deg(u, J) \geq k$ for every $u \in J$; and (ii) J is maximal.

Note that we say a subgraph of G is a **k-core subgraph** if it satisfies the degree constraint, but not necessarily to be maximal in G. It is immediate that k-core subgraph is a subgraph of k-core due to the maximal property of k-core. the k-core of a deterministic graph G can be obtained by recursively removing the vertices whose degrees are less than k, with a time complexity of $\mathcal{O}(m)$. The *core number* of a

vertex $u \in G$ is the highest core where u appears, denoted by $\mathrm{cn}(u)$.

Below we introduce the model of uncertain graphs.

*Definition 2 (**Uncertain graph**):* An uncertain graph $\mathcal{G} = (V, E, p)$ is a graph where V represents a set of n nodes, E represents a set of m edges and p is a function that maps every edge e in this graph to a real number between [0,1], denoted by $p(e)$.

We say an uncertain graph $\mathcal{S}$ is an induced subgraph of $\mathcal{G}$ if $V(\mathcal{S}) \subseteq V(\mathcal{G})$, $E(\mathcal{S}) \subseteq E(\mathcal{G})$, and the occurrence probabilities of the edges are the same. By $G_0 = (V, E)$, we denote the deterministic graph of $\mathcal{G}$ with $V(G_0) = V(\mathcal{G})$ and $E(G_0) = E(\mathcal{G})$.

In this paper, we use the well-known possible-world semantics to define our probabilistic k-core model. In the context of uncertain graph, a possible world is an instance of the uncertain graph, which is a deterministic graph denoted by $G = (V, E' \subseteq E)$ where E' are edges of E appearing in this possible world. In this paper, we say G is an instance graph of $\mathcal{G}$, denoted by $G \sim \mathcal{G}$. Same as most of the existing works on uncertain graph (e.g., [13]), we assume the existence probability of each edge is independent to each other. Thus, there are 2^m possible worlds (instance graphs) in total. The possibility of observing an instance graph G, denoted by $p(G)$, is:

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

Following the possible world semantics, we define the k-core probability of a node u in the uncertain graph $\mathcal{G}$ according to the k-core computation in all its instance graphs.

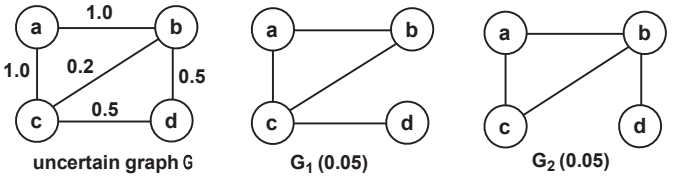


Fig. 2. Example of instance graphs regarding the uncertain graph

*Definition 3 (**k-core probability**):* Given an uncertain graph $\mathcal{G} = (V, E, p)$ and the degree constraint k, the k-core probability of a node $u \in V$ regarding $\mathcal{G}$, denoted by $p_k(u, \mathcal{G})$, is the probability that u appears in the k-core under possible world semantics; that is

$$p_k(u, \mathcal{G}) = \sum_{all\ G \sim \mathcal{G}} p(G) I_G(k, u) \tag{2}$$

where is an indicator function with $I_G(k, u) = 1$ if node u is in k-core of the instance graph G, otherwise $I_G(k, u) = 0$.

For presentation simplicity, we use $p(u)$ to denote $p_k(u, \mathcal{G})$ when there is no ambiguity.

Example 1: In Fig. 2, we show the uncertain graph $\mathcal{G}$ with edge occurrence probabilities. There are 8 possible instance graphs of $\mathcal{G}$ where G_1 and G_2 are two of them with probability 0.05 and 0.05, respectively. According to the definition, we have $p_k(a, \mathcal{G}) = 0.4$ and $p_k(d, \mathcal{G}) = 0.25$.

Now we have the definition of probabilistic k-core on uncertain graph.

*Definition 4 ((k,θ)-**core**):* Given an uncertain graph $\mathcal{G} = (V, E, p)$, the degree constraint k, and the probability threshold θ, the (k,θ)-core of $\mathcal{G}$, denoted by $C_{k,\theta}(\mathcal{G})$, consists of all nodes with k-core probability not less than θ, i.e., $p(u) \geq \theta$ for every node $u \in C_{k,\theta}(\mathcal{G})$.

Example 2: Given uncertain graph $\mathcal{G}$ in Figure 2, we have $C_{2,0.3}(\mathcal{G}) = \{a, b, c\}$ with $k = 2$ and $\theta = 0.3$.

Problem Statement. Given an uncertain graph $\mathcal{G}$, degree constraint k and probability threshold θ, we aim to develop efficient algorithm to compute the probabilistic k-core, i.e., (k,θ)-core, on uncertain graphs.

B. Problem Complexity

Given a deterministic graph, the k-core can be computed in linear time. However, with the presence of uncertainty, the computation of probabilistic (k,θ)-core is NP-hard.

Theorem 1: The problem of probabilistic k-core computation is NP-hard.

Proof: The reduction is from the k-clique problem [14], which checks if there is a clique with size k in a given deterministic graph. Given a deterministic graph G =(V, E), we construct the uncertain graph $\mathcal{G}$=(V, E, p) as follows. Two graphs have the same set of nodes and edges, and we set $p(e) = \frac{1}{2^{2m}+1}$ for every edge e in uncertain graph $\mathcal{G}$ where m is the number of edges. For probabilistic threshold $\theta = (\frac{1}{(2^{2m}+1)})^{\frac{k*(k+1)}{2}}$, we show that

(k,θ)-core of $\mathcal{G}$ is not empty $\iff$ $(k+1)$-clique exists in G.

"$\Longleftarrow$": If there is a $(k+1)$-clique S in the graph G, S will be k-core subgraph[2] of $\mathcal{G}$ with probability at least θ because S has $\frac{k*(k+1)}{2}$ edges. Note that S is a k-core if all of its edges appear in the instance graph (possible world). The probability mass of these possible worlds is $(\frac{1}{2^{2m}+1})^{\frac{k*(k+1)}{2}} = \theta$.

"$\Longrightarrow$": Below, we show that (k,θ)-core of $\mathcal{G}$ is empty if there is no $(k+1)$-clique in G. Let $p(S)$ denote the occurrence probability of a k-core subgraph where $p(S) = p(e)^{e(S)}$ where $e(S)$ is the number of edges. The key motivation of the proof is the fact that a k-core subgraph S with minimal number of edges is a $(k+1)$ clique (i.e., $p(S) = p(e)^{\frac{k*(k+1)}{2}}$). If S is not a $(k+1)$-clique, it has at least $k+2$ nodes (i.e., with smaller occurrence probability where $p(S) \leq p(e)^{\frac{k*(k+2)}{2}}$). Suppose $\mathcal{S} = \{S_1, S_2, \ldots, S_h\}$ be a set of k-core subgraphs in G, and we have at most $h = 2^m$ distinct k-core subgraphs. Since there is no $(k+1)$-clique in G, we have $p(S_i) \leq p(e)^{\frac{k*(k+2)}{2}}$. For any node $u \in V$, its k-core probability is bounded by α with $\alpha = h \times (\frac{1}{(2^{2m}+1)})^{\frac{k*(k+2)}{2}}$. Recall that $\theta = \frac{1}{(2^{2m}+1)^{\frac{k*(k+1)}{2}}}$, we have $\alpha \leq 2^m \times (\frac{1}{2^{2m}+1})^{\frac{k*(k+1)}{2}} \times (\frac{1}{2^{2m}+1})^{\frac{k}{2}} \leq 2^m \times (\frac{1}{2^{2m}+1})^{\frac{k*(k+1)}{2}} \times (\frac{1}{2^{2m}+1})^{\frac{1}{2}} \leq 2^m \times \theta \times (\frac{1}{(2^{2m}+1)})^{\frac{1}{2}} \leq \theta$. So the ($k$,$\theta$)-core of $\mathcal{G}$ is empty. ■

C. Some Properties

We show that some of the properties of deterministic k-core can be naturally extended to (k, θ)-core.

[2]We say S is a k-core subgraph if every nodes in S has at least k neighbors. It would be the k-core or a subgraph of k-core.

Nest property. The derterminstic k-core has the nest property, i.e., k_1-core $\subseteq$ k_2-core if $k_1 \geq k_2$. The (k,θ)-core also has the nest property in the sense that we have (k_1, θ_1)-core $\subseteq$ (k_2, θ_2)-core for $k_1 \geq k_2$ and $\theta_1 \geq \theta_2$. This is because we have k_1-core $\subseteq$ k_2-core in *every* possible world.

Upperbound. In [12], the uncertain clique, namely α clique defined based on possible worlds semantics is a global model where the probability of an α clique S is a subgraph where S is clique in the possible instance graphs with probability at least α. In the deterministic case, we may safely claim that if a node u is *not* a k-core member, it cannot contribute to any $(k+1)$-clique because $(k+1)$-clique is a k-core. With similar rationale, we have that if a node u is not a (k, θ)-core member, it cannot contribute to an α $(k+1)$-clique S with $\alpha \geq \theta$. For instance, if we have $p(u) = 0.2$ (i.e., probability of u being k-core member) for $k = 10$, we may immediately exclude u from the computation of α 11-clique if $\alpha > 0.2$.

D. Discussion of k-core Models

We first formally introduce two *local* k-core models discussed in Section I.

Probabilistic degree based k-core model is proposed in [6] for k-core computation on uncertain graphs. The key is the concept of η-degree for a node u and an uncertain graph $\mathcal{S}$, denoted by η-deg$(u, \mathcal{S})$, which is the largest possible d such that u has at least probability η to have at least d neighbors, i.e., $Pr(deg(u, \mathcal{S}) \geq d) \geq \eta$ where $deg(u, \mathcal{S})$ is a random variable for the degree distribution of u on uncertain graph $\mathcal{S}$. The computation of the k-core, namely (k,η)-core, is immediate by replacing G with $\mathcal{G}$ and the condition $deg(u, G) < k$ with η-deg$(u, \mathcal{G}) < k$.

Expected degree based k-core model is a variant of weighted k-core model [7]. We consider the expected degree of node in the expected k-core model. The only difference with (k,η)-core algorithm is that we remove a node if $\mathbb{E}(deg(u, \mathcal{G})) < k$.

Discussion. According to the definitions of three models, the essential difference between our (k, θ)-core model and two local models is how to use possible world semantics. Particularly, our proposed model computes the deterministic k-core on each instance graph (possible world), and find the nodes which are likely to be included as k-core members in instance graphs, i.e., aggregation on the computation of each individual possible world. While two local models compute k-core based on the degree distributions. Not surprisingly, our empirical study on real-life data shows that the results of our probabilistic model are different with that of two local models.

An alternative global k-core model is to find subgraphs $\{S\}$ such that the probability of S being k-core in the possible worlds is no less than θ, i.e., $Pr(S$ is k-core$) \geq \theta$. However, the inherent limit of this model is that there are many possible sets of $\{S\}$ (i.e., candidate k-core set) and all of them have extremely small appearance probabilities. For instance, we randomly sample $1,000$ possible instance graphs from DBLP data (See data description in Section VI), and none of the corresponding 10-cores is the same to others. Thus it is difficult to find some "representative" probabilistic k-core sets in this model. We may alleviate this issue by ignoring the maximal property of k-core, and consider k-core subgraphs (i.e., subgraphs where every node has k neighbors) in each possible world. Nevertheless, the large number of possible k-core subgraphs in each possible world make the computation prohibitively expensive [3].

E. Application of (k,θ)-core

Intuitively, compared with two local models, (k,θ)-core model is more suitable to the scenarios where the possible world semantics is applied in a global way. As shown in Section II-C, (k,θ)-core can be used for the pruning of nodes in α clique computation. Below is a concrete example.

Example 3: Let $\mathcal{G}$ be an uncertain complete graph with n nodes and $\frac{n(n-1)}{2}$ edges with occurrence probability p for each edge, it is immediate that the existence probability of a n-clique of $\mathcal{G}$ is $p^{\frac{n(n-1)}{2}}$. Regarding (k,θ)-core with k=n-1, we have $p_{n-1}(u, \mathcal{G}) = p^{\frac{n(n-1)}{2}}$ for any node u which is exactly same as the n-clique probability. For (k,η)-core model, a node will be returned if $k = n-1$ and $\eta \leq p^{(n-1)}$, which is much larger than $p^{\frac{n(n-1)}{2}}$. Thus, for any α value with $p^{\frac{n(n-1)}{2}} < \alpha \leq p^{(n-1)}$, ($k$,$\theta$)-core can prune all nodes from the computation of α n-clique while (k,η)-core cannot prune any node.

Similarly, (k,θ)-core model shows a good performance on the influence evaluation of the nodes following the IC model, which is a well-known influence model using possible world semantics in a global way. Specifically, the influence of a node u on IC model is the average number of reachable nodes of u in all instance graphs. The following example demonstrates that the (k,θ)-core model is more likely to find influential nodes.

Example 4: As shown in the Introduction, to find three most influential nodes in Fig. 1(b), $\{a, b, c\}$ and $\{c, d, e\}$ will be returned by (k,η)-core and (k,θ)-core models, respectively. Intuitively, nodes d and e are better positioned (closer to the graph center and well connected to nearby nodes) in terms of influence compared with nodes a and b. It turns out that, under IC model, the influence of a, b, d, e is 3.58, 3.58, 4.21 and 4.21, respectively.

In the experiments, we justify this observation on Twitter data in the second case study. We also show that (k,θ)-core model can better evaluate the engagement in the first case study.

III. Basic Sampling Algorithm

In this Section, we introduce the basic sampling technique to compute (k,θ)-core with theoretical guarantee.

It is computational prohibitive to calculate k-core probabilities of the nodes directly based on Equation 2. As the number of possible words is 2^m, where m is the number of edges. Therefore, we resort to the Monte Carlo sampling method, which has been widely used to solve problems on uncertain data. Algorithm 1 outlines the basic sampling algorithm. Each instance graph $G \sim \mathcal{G}$ is sampled with probability $p(G)$. In

[3]According to our initial study, it is difficult to develop efficient k-core subgraph enumeration algorithm, because given a node u is chosen, we cannot trivially identify the candidate nodes for the following k-core subgraph computation. Note that for the problem of clique, we can immediately narrow down the search space to the neighbors of u.

practice, we generate each sampled graph by independently choosing every edge according to its occurrence probability. The k-core probability of each node is obtained based on the count of k-cores of the sampled possible worlds (sampled graphs). Let $\hat{p}(u)$ denote the estimator of the k-core probability of u (i.e., $p(u)$), which is estimated by

$$\hat{p}(u) = \frac{u_c}{s} \tag{3}$$

at Line 8 where u_c is the number of sampled graphs in which u is in k-core and s is the total number of samples. It is easy to see that the expectation of $\hat{p}(u)$ is $p(u)$. Based on the well-known Hoeffding's inequality [15] and union bound [16], we have that $\hat{p}(u)$ is ϵ-approximation of $p(u)$ (i.e., $|\hat{p}(u)-p(u)| \leq \epsilon$) for *every* node $u \in \mathcal{G}$ with confidence $1-\delta$, when the number of samples s is set to $\frac{1}{2\epsilon^2}\ln(\frac{2n}{\delta})$ where n is the number of nodes in $\mathcal{G}$, approximate error ϵ and confidence δ fall in $(0, 1]$. The time complexity of the algorithm is $O(sm)$ since the k-core can be computed in $O(m)$ time for each sampled graph.

Algorithm 1: **Basic Sampling($\mathcal{G}$, k, θ, ϵ, δ)**

Input : $\mathcal{G}$: an uncertain graph, k : degree constraint
θ: probabilistic threshold, ϵ: approximate error
δ: confidence

Output: $C_{k,\theta}(G)$

1 $u_c := 0$ for all nodes $u \in \mathcal{G}$;
2 $s \leftarrow$ number of sample graphs required;
3 $\mathcal{S} \leftarrow$ sample s instance graphs;
4 **for** each instance graph G in $\mathcal{S}$ **do**
5 $C_k(G) \leftarrow$ k-core of G;
6 **for** each node $u \in C_k(G)$ **do**
7 $u_c := u_c + 1$;
8 $\hat{p}(u) := \frac{u_c}{s}$ for each $u \in \mathcal{G}$;
9 **return** nodes $\{u\}$ with $\hat{p}(u) \geq \theta$

IV. Pruning Techniques

To achieve a decent estimation theoretical guarantee, we usually need a considerable large number of samples. In the paper, we use $p^+(u)$ to denote the upper bound of the k-core probability of a node u (i.e., $p(u)$). For the estimator $\hat{p}(u)$, we use $\hat{p}^-(u)$ and $\hat{p}^+(u)$ to denote its lower and upper bounds. Clearly, we can safely exclude a node u from (k,θ)-core if $p^+(u) < \theta$. In the sampling based algorithm, a node u can also be excluded if $\hat{p}^+(u) < \theta$. Similarly, we can confirm a node u belong to (k,θ)-core if $\hat{p}^-(u) \geq \theta$. In this Section, we design three pruning techniques to derive tight lower and upper bounds to reduce the computational cost of Algorithm 1.

A. Deterministic K-core based Pruning

Let G_0 denote the deterministic graph which treats every edge of the uncertain graph $\mathcal{G}$ as a deterministic edge. Clearly, for any instance graph $G \sim \mathcal{G}$, we have $G \subseteq G_0$. In the following, we show how to prune some nodes based on the k-core of G_0.

Theorem 2: Given an uncertain graph $\mathcal{G}$ and its deterministic graph G_0, if a node is not included in k-core of G_0, it can be excluded from further computation.

Proof: Given two deterministic graphs G_1 and G_2, with $G_1 \subseteq G_2$, we show that $C_k(G_1) \subseteq C_k(G_2)$, i.e., the k-core of G_1 is a subset of k-core of G_2. As $G_1 \subseteq G_2$, for each node $u \in G_1$, we have $NB(u, G_1) \subseteq NB(u, G_2)$ where $NB(u, G)$ denote the neighbors of u in graph G. Let S_1 and S_2 be the k-core of G_1 and G_2, it is immediate that nodes in S_1 is a k-core subgraph in G_2. Due to the maximal property of k-core, we have $S_1 \subseteq S_2$. This implies that if a node u is removed in the k-core computation of G_2, it must also be removed from G_1. For any instance graph $G \sim \mathcal{G}$, we have $G \subseteq G_0$, therefore, we can safely exclude the nodes not in $C_k(G_0)$ from the G in the k-core computation. Thus, for a non k-core node u of G_0, we have $p(u) = 0$ and it does not contribute to the k-core computation in any instance graph of $\mathcal{G}$. Consequently, the theorem holds. ■

In the following pruning techniques, we assume all nodes not in k-core of G_0 are already removed from $\mathcal{G}$ for presentation simplicity.

B. Probabilistic Upper Bound based Pruning

We investigate how to compute the k-core probability upper bound of a node u based on its neighborhood information in $\mathcal{G}$. Note that we rely on probabilistic theory and do not need to sample the uncertain graph (i.e., materialize some instance graphs). Thus, the probabilistic upper bound derived is $p^+(u)$ for the k-core probability of each node u.

For each instance graph $G \in \mathcal{G}$, we can immediately remove a node u from k-core of G if it has less than k neighbors. This implies that

$$p(u) \leq Pr(deg(u, \mathcal{G}) \geq k) \tag{4}$$

because here we already assume that neighbors of u are always in k-core in any instance graph. Thus, we can exclude u from (k,θ)-core if u has at least k neighbors with probability less than θ; that is, $Pr(deg(u, \mathcal{G}) \geq k) < \theta$. Thanks to the independent assumption among edges, we can quickly compute $Pr(deg(u, \mathcal{G}) \geq k)$ following the classic dynamic programming methods for uncertain data [6].

Based on the above degree distribution computation on neighbor edges, we can come up with an upper bound of k-core probability, $p^+(u)$, for every node $u \in \mathcal{G}$. The following theorem suggests that we can further refine $p^+(u)$ based on the upper bounds of its neighbors.

Theorem 3: Let v_1, v_2, ..., v_h be the neighbor nodes of the node u, we have $p(u) \leq (\sum_{i:=1}^{h} min(p(e(u, v_i)), p^+(v_i)))/k$.

Proof: We use E_v to denote the event that the edge (u, v) occurs *and* v is a k-core member. The occurrence probability of E_v is the probability mass of the instance graphs in $\mathcal{G}$ where edge (u, v) occurs and v is a member of k-core. We have $Pr(E_v) \leq min(p(e(u, v)), p^+(v))$ since two events may depend on each other.

Let X_i be an indicator random variable for the event E_{v_i}, where $X_i = 1$ if event E_{v_i} occurs, and $X_i = 0$ otherwise. Let $X = X_1 + X_2 + \ldots + X_h$. To be included in k-core of an instance graph, a node u needs at least k supports from k-core nodes. According to the definition of k-core probability in Equation 2, we have $p(u) = Pr(X \geq k)$. Then we have $Pr(X \geq k) \leq (\sum_{i:=1}^{h} min(p(e(u, v_i)), p^+(v_i)))/k$ according to Markov Inequality [17]. So the theorem holds. ■

Algorithm 2: Calculate Probabilistic Upper bound ($\mathcal{G}$)

Input : uncertain graph $\mathcal{G}$
Output : $p^+(u)$ for every node $u \in \mathcal{G}$
1 **for** each node $u \in \mathcal{G}$ **do**
2 $p^+(u) :=$Pr$(deg(u,\mathcal{G}) \geq k)$; /* Inequality 4 */;
3 $u_f := true$;
4 **while** there exists a node u with $u_f = true$ **do**
5 $u_f := false$;
6 Update $p^+(u)$ according to Theorem 3;
7 **if** $p^+(u)$ is decreased by at least Δ at Line 6 **then**
8 $v_f := true$ for every neighbor of u;

Computing Probabilistic upper bounds. Based on Inequality 4 and Theorem 3, Algorithm 2 describes the calculation of k-core probability upper bounds for nodes in $\mathcal{G}$. Lines 1-3 initialize the probabilistic upper bounds by Inequality 4 using dynamic program techniques [6]. The time cost is $O(ndk)$ where d is the average degree. We use a flag u_f to indicate if $p^+(u)$ needs to be updated. At each iteration (Lines 4-8), we update $p^+(u)$ following Theorem 3 if there is a tighter upper bound, and the change will not be propagated to neighbors if the decrease is not significant[4]. The time cost in each iteration is $O(d)$.

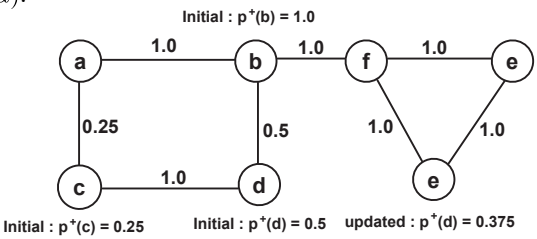


Fig. 3. Example of probabilistic upper bound pruning

Below is an example of probabilistic upper bound.

Example 5: Given the uncertain graph in Figure 3 with $k = 2$, we have $p^+(b) = 1.0$, $p^+(c) = 0.25$, and $p^+(d) = 0.5$ at the initial stage (Line 2 of Algorithm 2). By applying Theorem 3, we have $p(d) \leq \frac{\min(1,0.25)+\min(1,0.5)}{2} = 0.375$. Thus, $p^+(d)$ is updated from 0.5 to 0.375.

C. Sampling based Upper/Lower Bounds for Pruning

In addition to the probabilistic bound, we also investigate how to obtain lower and upper bounds for k-core probability estimator $\hat{p}(.)$ when we partially compute the sampled graphs. As shown in Section III, we can compute k-core in each individual sampled graph, and then come up with the k-core probability of a node u based on its estimator $\hat{p}(u)$. In Section V-B, we propose a new computing paradigm to compute $\hat{p}$ such that we can reduce the computing cost. The key is to take advantage of the upper/lower bounds obtained from *partially* computed data, instead of simply sampling all edges and then computing the complete k-core in each sampled graph.

In a sample graph G_i, we use $u.stat$ to denote three possible status of a node u:
(1) ✓ : confirmed in k-core of G_i.
(2) ⊗ : excluded from k-core of G_i.
(3) ? : needing further computation.

[4]In our implementation, we set Δ to 0.01.

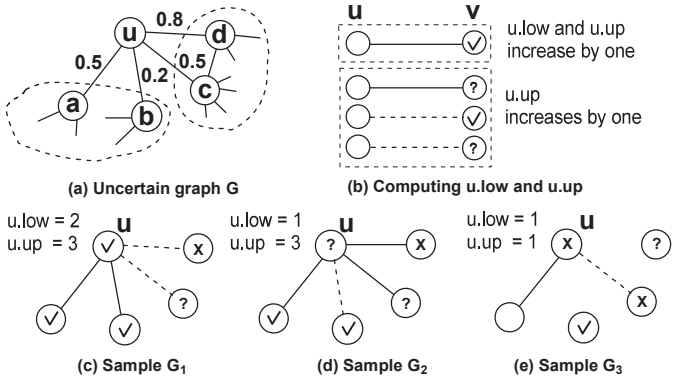


Fig. 4. Example of sampling based upper/lower bounds

Similarly, each edge $e(u,v)$ in G_i also has three status:
(1) *chosen*: chosen in G_i.
(2) *unchosen*: not chosen in G_i.
(3) *untouched*: not sampled yet.

Given a node $u \in \mathcal{G}$, we can derive the lower/upper bounds for $\hat{p}(u)$ based on the status of its neighbor nodes and edges in a set of sampled graphs. In sample graph G_i, we use $u.low$ and $u.up$ to denote the upper and lower bounds of k-core support from its neighbors. As shown in Figure 4(b), if there is a neighbor v with status ✓ (i.e., is k-core) and the edge $e(u,v)$ is chosen (i.e., solid line) in G_i, v is a k-core support of u regardless of the following computation because v will never change status in G_i. Thus, both $u.low$ and $u.up$ increases one for this. There are three scenarios in which v *may* support u where $u.up$ will increase one: (i) the edge is chosen but $v.stat$ is ?; (ii) the edge is not sampled yet (i.e., dashed line), and $v.stat$ is ✓; and (iii) the edge is not sampled yet and $v.stat$ is ?. In the remaining five scenarios, v *cannot* provide support to u, and hence does not contribute to $u.low$ and $u.up$. Clearly, we can set $u.stat$ to ✓ if $u.low \geq k$. Similarly, $u.stat$ is set to ⊗ if $u.up < k$. Note that, if $u.stat =$ ⊗, we do not need to sample any of its untouched neighbor edges. Fig 4(c)-(e) show situations where $u.stat$ changes to ✓,? and ⊗ with k = 2.

According to the status of a node u in the sample graphs $\{G_1, G_2, \ldots, G_s\}$, we can obtain the upper and lower bounds of $\hat{p}(u)$ by counting the number of ✓ and ?. Let n_1 and n_2 denote the number of ✓ and ?, respectively, we have

$$\hat{p}^-(u) = \frac{n_1}{s} \quad (5)$$

$$\hat{p}^+(u) = \frac{n_1 + n_2}{s} \quad (6)$$

The correctness of the above two Equations is immediate based on definition of $\hat{p}$ in Equation 3. Then we can safely (1) exclude the node u from (k,θ)-core if $\hat{p}^+(u) < \theta$; and (2) add the node u to (k,θ)-core if $\hat{p}^-(u) \geq \theta$.

V. Advanced Sampling Algorithms

A. Motivation

As discussed in Section II-D, the key challenge of (k,θ)-core computation is that, even we already find a candidate set R, we cannot simply exclude other nodes from following computation because they may contribute to the verification

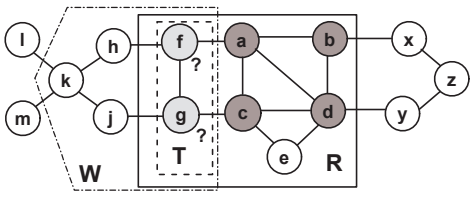


Fig. 5. Motivation of advanced sampling algorithm

of these candidates. To verify a candidate, the existing k-core algorithm on sample graph (i.e., deterministic graph) follows a *peeling-like* computing paradigm. In practice, users often use a considerably large k to get a cohesive subgraph, resulting in a k-core with much smaller size than the original graph. This implies that the majority of the edges and nodes will be accessed in each sample graph because all nodes with lower core values must be processed before the nodes in k-core are identified.

This motivates us to develop a new sampling algorithm which will carefully explore nodes and edges to avoid unnecessary computation. Figure 5 illustrates the key idea of our advanced sampling algorithm which consists of three steps:
(1) **Pruning**. By applying the deterministic k-core pruning (Section IV-A), probabilistic upper bound pruning (Section IV-B) and sampling-based pruning technique (Section IV-C), we can identify a set of result candidate R based on their k-core probability upper-bound such that $p^+(u) \geq \theta$ for any node $u \in R$; In Figure 5, we assume $R = \{a, b, c, d, e, f, g\}$.
(2) **Initial Computing**. Conduct the basic sampling technique (Algorithm 1 in Section III) on R such that most of the nodes $\{u\}$ in R are either *confirmed* (i.e., $\hat{p}^-(u) \geq \theta$) or *pruned* (i.e., $\hat{p}^+(u) < \theta$)[5]. In Figure 5, we assume the dark gray nodes (e.g., a, b, c, and d) are confirmed, and node e is pruned from result.
(3) **Verification**. Verify the remaining nodes, denoted by T, to complete the (k,θ)-core computation. Although the nodes outside of R may be involved in the verification phase, our algorithm aims to access as few as possible. Toward this end, we develop a new k-core computing paradigm for sample graphs, namely *k-core membership check* (Section V-C). In Fig. 5, we assume T has two the light gray node $\{f, g\}$ and nodes $\{h, k, j\}$ are visited in the verification.

B. Advanced Sampling technique

Algorithm 3 outlines the framework of our Advanced Sampling algorithm. In the *pruning phase* (Lines 1-2), we exclude the non-promising nodes from the result candidate and keep remaining nodes in R. In the *initial computing phase* (Lines 3-6), the basic sampling algorithm is conducted on $\mathcal{R}$, which is a deduced uncertain graph of $\mathcal{G}$ according to nodes in R. Note that, we need to count the nodes in $V(\mathcal{G}) \setminus V(\mathcal{R})$ for the computation of $\hat{p}^+(u)$. Based on each computed sample graph $\mathcal{R}_i$, we construct a *partially computed* sample graph $\mathcal{G}_i$ where $u.low$ and $u.up$ are set to the same values for node $u \in R$, and we do not need to re-sample the edges in R_i. For the remaining nodes $\{v\}$, we simply set $v.low = 0$ and $v.up = deg(v, G_0)$. Then, for each $u \in R$ we can compute $\hat{p}^+(u)$ and $\hat{p}^-(u)$ based on the sampling based lower/upper bound techniques (Section IV-C). We use a set T to keep the nodes in R which need verification. In *verification phase* (Lines 8-19), we apply proposed k-core membership check algorithm (Details will be introduced in Section V-C) on sample graphs $\{G_i\}$ with focus on verifying whether a node u in T is a k-core member in each sample graph based on $\hat{p}^-(u)$ and $\hat{p}^+(u)$. The algorithm terminates when T is empty as all nodes with $\hat{p}(u) \geq \theta$ are included in the result set C.

Correctness. According to the correctness of the pruning rules and k-core computation in each sample graph [6], the advanced sampling algorithm is the same as the basic algorithm in terms of k-core probability estimation. Every node u with $\hat{p}(u) \geq \theta$ will be included in the result. Thus, Algorithm 3 can also achieve ϵ-approximate with confidence $1 - \delta$ for every node in $\mathcal{G}$.

Time Complexity. The dominant costs of Algorithm 3 are initialization (Line 5) and k-core membership check (Line 8). The cost of initialization is by $O(sm)$ where s is the number of sample graphs. As the time cost of each k-core membership check computation is $O(m)$ as shown in Section V-C, the worst case time complexity of algorithm is the same as Algorithm 1. Nevertheless, our empirical study shows that the advanced sampling algorithm is much more competitive due to advanced techniques proposed.

Algorithm 3: **Advanced Sampling($\mathcal{G}$, k, θ, ϵ, δ)**

Input : $\mathcal{G}$: an uncertain graph, k : degree constraint
θ: probabilistic threshold, ϵ: approximate error
δ: confidence
Output : $C_{k,\theta}(\mathcal{G})$

1 $G_0 \leftarrow$ deterministic graph of $\mathcal{G}$;
2 $R \leftarrow$ deterministic k-core pruning on G_0 (Theorem 2);
3 $\mathcal{R} \leftarrow$ the induced uncertain subgraph of $\mathcal{G}$ based on nodes in R;
4 $R \leftarrow$ apply probabilistic upper bound pruning on $\mathcal{R}$ (Algorithm 2) ;
5 $\{G_i\} \leftarrow$ Apply Basic Sampling (Algorithm 1) on $\mathcal{R}$;
6 $T \leftarrow$ nodes need to be verified; $C \leftarrow$ nodes confirmed;
7 **for** each G_i **do**
8 **K-core membership check**(G_i, T);
9 **for** each node $u \in T$ **do**
10 **if** $u.stat = \surd$ **then**
11 $p^-(u) := p^-(u) + \frac{1}{s}$(Equation 5);
12 **else**
13 $p^+(u) := p^+(u) - \frac{1}{s}$ (Equation 6) ;
14 **if** $p^-(u) \geq \theta$ **then**
15 Move node u from T to C;
16 **else if** $p^+(u) < \theta$ **then**
17 exclude u from T;
18 **if** $T = \emptyset$ **then**
19 **return** C

C. K-core Membership Check

The key of the verification phase is the k-core membership check of the nodes in T on sample graphs $\{G_i\}$. We illustrate two possible search paradigms in Fig 6. The existing k-core computation on sample graph (i.e., deterministic graph) follows a peeling-based search order. Let V denote all the

[5]Note that we need consider neighbor nodes for $\hat{p}^+(u)$

[6]Correctness of target k-core computation will be presented in Section V-C

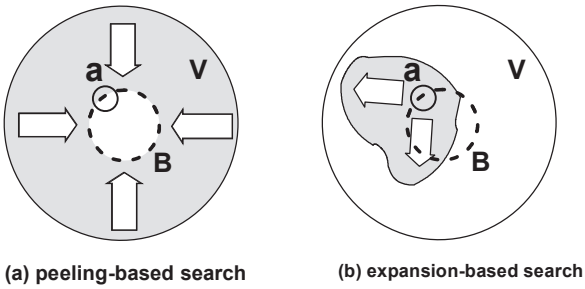


Fig. 6. Two k-core membership check search paradigms

nodes of G, which are included in the circle with solid line. We assume ones with higher core number are closer to the center. Given a degree constraint k, B denote the nodes in k-core, covered by the circle with dash line. As shown in Fig. 6(a), we need to access all nodes outside of B to identify if node a is a k-core member due to the nature of the peeling-based computing paradigm. As an alternative, we may identify the membership of a in an expansion by recursively checking if there are enough k-core supports. With a good chance, we may stop by only exploring a small set of nodes as shown in Fig. 6(b).

Intuitively, the goodness of two search paradigms depends on the size of T. Expansion-based search is expected to have a good performance if the size of T is small. On contrary, if the size of T is large, the peeling-based search is a better choice because it can identify all nodes in T within one round regardless the size of T. As shown in Fig. 6(b), there is an expansion region for each vertex $u \in T$. A large number of vertices in T will lead to many overlaps among these regions (i.e., visit vertices/edges multiple times).

Check k-core membership in sample graph

Algorithm 4 illustrates the pseudo-code of our k-core membership check method on a partially computed sample graph G_i and a set T of nodes to be verified. A priority queue, denoted by Q, is employed to access nodes in sequence, which is initialized by the nodes in T at Line 1. The key of a node is its status, status $\surd$ and $\otimes$ have higher priority than $?$. We use $u.sampled$ to denote if all edges of node u has been sampled. Line 38 terminates the algorithm whenever the status of nodes in T are clear ($\surd$ or $\otimes$). Note that we can claim all nodes with status $?$ is k-core member if $Q = \emptyset$ since all nodes have been processed (Lines 39-40).

For a node u popped, Lines 5-31 process its neighbor nodes according to their status of the node and the edge. A flow chart is also depicted in Fig. 7 for better understanding of the algorithm. In the flow chart, an edge is represented by a solid line with a tick if it is *recently* chosen by sampling at Line 6. Note that we do not need to sample an existing edge again if it is already chosen, which is represented by a solid line alone. An edge with dash line and a cross means that this edge is rejected in sampling at Line 6. Because the untouched edges always contribute to the upper bounds of two corresponding nodes, whose upper bounds will be both decreased by one whenever the edge is rejected and hence removed from G_i (e.g., Line 11, Line 16, and Line 21). We do not need to discuss the case of edge rejection in the following part.

For each neighbor node v of u, we carefully update their

Algorithm 4: **K-core membership check(G_i, T)**

Input : G_i : a partially computed sample graph of $\mathcal{G}$
T : a set of nodes need membership check
Output : check if each node in T is in k-core of G_i

1 $Q \leftarrow$ Push the nodes in T; $u.sampled = false$ for $u \in G_i$;
2 **while** $Q \neq \emptyset$ **do**
3 $u \leftarrow Q.pop()$; $u.sampled := true$;
4 **for** each neighbor v of u **do**
5 **if** $e(u,v)$ is *untouched* and $u.stat \neq \otimes$ **then**
6 Choose $e(u,v)$ with $p(e(u,v))$;
7 **switch** $u.stat$ **do**
8 **case** $\surd$ **do**
9 **if** $v.stat = ?$ **then**
10 **if** e is *unchosen* **then**
11 **Decrease**(v) ;
12 **else**
13 **Increase**(v) ;
14 **case** $\otimes$ **do**
15 **if** $v.stat = ?$ **then**
16 **Decrease**(v);
17 **case** $?$ **do**
18 **if** $e(u,v)$ is *unchosen* **then**
19 $u.up - -$; remove $e(u,v)$ from G_i;
20 **if** $v.stat = ?$ **then**
21 **Decrease**(v);
22 **else if** $e(u,v)$ is *recently chosen* **then**
23 **if** $v.stat = \surd$ **then**
24 $u.low + +$;
25 **else if** $v.stat = \otimes$ **then**
26 $u.up - -$;
27 remove $e(u,v)$ **If** $v.stat$ is $\surd$ or $\otimes$;
28 **if** $u.up < k$ **then**
29 $u.stat := \otimes$; Push u to Q;Goto Line 2;
30 **else if** $u.low \geq k$ **then**
31 $u.stat := \surd$; Push u to Q;Goto Line 2;
32 **if** $u.stat = ?$ **then**
33 **for** each neighbor v of u **do**
34 Push v to Q **if** $v.sampled = false$ and $v.stat = ?$;
35 **else**
36 Remove u from G_i;
37 **if** none of the nodes in T is $?$ **then**
38 **return**
39 **for** all nodes $u \in T$ with $u.stat = ?$ **do**
40 $u.stat = \surd$;

lower and upper bounds of k-core support (i.e., $u.low$, $v.low$, $u.up$ and $v.up$) as shown in Fig. 7. Then their status may be updated accordingly. As mentioned in Section IV-C, the k-core membership of node u is *confirmed* (i.e.,$u.stat = \surd$) if $u.low \geq k$. A node u is excluded from k-core (i.e., $u.stat = \otimes$) if $u.up < k$. Otherwise, the membership of u is unclear, with $u.stat = ?$.

Clearly, we only need to maintain the lower/upper bounds for nodes whose k-core memberships are unclear, i.e., with

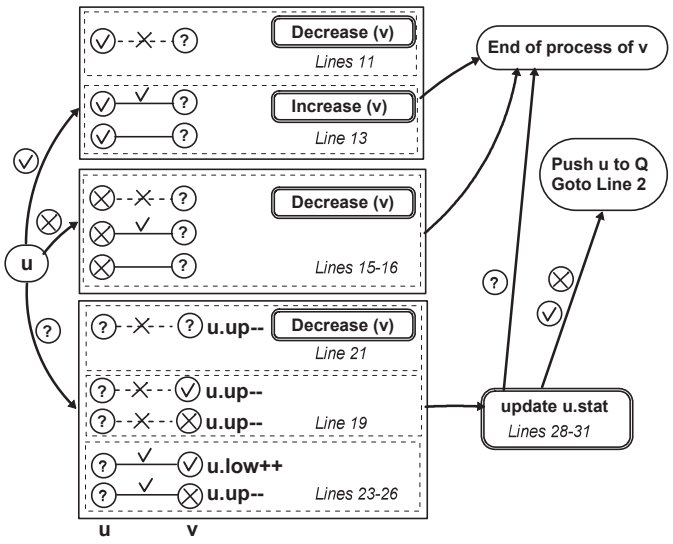


Fig. 7. Flows chart of k-core membership check

status ?. As shown in Fig. 7, we update $u.up$ and $u.low$ when $u.stat = ?$ (left side of the edges). The same rule goes to v when $v.stat = ?$ (right side of the edges). Generally, given a node x, its bounds will be updated in two ways: (1) *active*: when x is popped from Q (i.e., u in the algorithm); and (2) *passive*: x is updated by neighbors (i.e., v in the algorithm).

Active update ($u.stat = ?$, Lines 18-31). If the edge is chosen in this iteration (i.e., solid line with a tick), Line 24 (resp. Line 26) increases (resp. decreases) $u.low$ (resp. $u.up$) by one if $v.stat = \surd$ (resp. $\otimes$). Lines 28-31 may set the status of u to $\surd$ or $\otimes$ following the active update, and then put u to the Q because it will propagate this information to its neighbors. The remaining unvisited edges will not be processed until u is popped with status $\surd$ or $\otimes$. If $u.stat$ remains ? after processing all neighbors, Line 34 ensures the unsampled neighbors (i.e., nodes has not been pushed) with status ? are in Q for further processing. Note that $u.stat$ may be passively updated later due to the change of other nodes.

Passive update ($u.stat$ is $\surd$ or $\otimes$, Lines 9-16). The lower/upper bounds of a node v may be updated passively when its neighbor u is activated (i.e., popped from Q). Algorithm 5 and Algorithm 6 illustrate the details of the update (i.e., increase of $v.low$ and decrease of $v.up$). In addition to update lower/upper bounds, the node v will be pushed to Q for status information propagation purpose if its status is set to $\surd$ or $\otimes$. Below are details.

(1)*Pass* $\surd$ (Lines 9-13). The increase of $v.low$ is invoked by u when edge (u, v) is chosen.

(2)*Pass* $\otimes$ (Lines 15-16). $v.up$ will be decreased by one if $v.stat$ is ?.

Note that we ensure a sampled edge $e(u, v)$ will be removed from further computation if the status of u or v is clear at Lines 36 and 27.

Correctness. The status and lower/upper bounds of the nodes in G_i have been initialized at the beginning of membership check. In the following computation, the bounds and status may be updated in an active or passive way. As shown in the flow chart in Fig. 7, we ensure that the status and upper/lower bounds of each node are correctly maintained upon each possible update [7]. For any two nodes with edge $e(u, v)$, if e is *untouched*, both $u.up$ and $v.up$ count e. Once e is rejected, both will decrease by one and e is removed. Suppose the e is sampled by u, if one of the status of u and v is clear ($\surd$ or $\otimes$), their lower or upper bounds will be updated accordingly, and e will be removed to avoid duplicate computation. If the status of u remains ?, e will be left for possible passive updates. Note that if all nodes are sampled and queue is empty, it is immediate that all the nodes with status ? are k-core members in G_i. The computation is driven by the k-core membership check of nodes in T, and all following invoked updates are correctly handled. Thus, we can eventually correctly confirm memberships of the nodes in T.

Algorithm 5: **Increase(v)**

```
   Input : v : the node v in sample graph
1  v.low := v.low + 1;
2  if v.low ≥ k then
3      v.stat := √ ;
4      if v is not in Q then
5          Push v to Q ;
```

Algorithm 6: **Decrease(v)**

```
   Input : v : the node v in sample graph
1  v.up := v.up − 1;
2  if v.up < k then
3      v.stat := ⊗ ;
4      if v is not in Q then
5          Push v to Q ;
```

Time Complexity. As there are three types of key for priority queue Q, the maintenance cost is $O(1)$ per update. Each node can be popped from Q at most twice during the computation, one with status ? and another with status $\surd$ or $\otimes$. Thus, each edge will be processed at most twice, and the time complexity of the k-core membership check algorithm is $O(m)$.

VI. Experiment

In this section, we evaluate the effectiveness and efficiency of proposed techniques on comprehensive experiments.

A. Experimental Setting

Algorithms. We evaluate the efficiency of the following two techniques for (k,θ)-core computation in the experiments.

- **BSampling**. The basic sampling technique proposed in Algorithm 1 (Section III).
- **ASampling**. Our advanced sampling technique proposed in Algorithm 3 (Section V-B).

We also implement the (k,η)-core [6] and the expected k-core (Section II-D) computing algorithms for the comparison of k-core models.

Datasets. We deploy four graph datasets to evaluate effectiveness and efficiency of all techniques. Table II shows important statistics of these graphs.

[7]We do not need to maintain lower/upper bounds if the status of a node is $\surd$ or $\otimes$.

Dataset	#Nodes	#Edges	d_{avg}	d_{max}	k_{max}
Flickr	105,938	2,316,948	43.7	546	226
DBLP	1,566,919	6,461,300	8.3	611	115
Email Eron	36,692	183,831	10.0	1383	44
Yelp	552,339	1,781,908	6.5	3812	106

TABLE II
STATISTICS OF DATASETS

Flickr (https://www.flickr.com) is an online community. Its service is widely used by photo researchers and bloggers. As shown in [18], the edge probability is set to the Jaccard coefficient to the interest groups shared by the two users. We set Flickr as our default dataset.

DBLP (http://dblp.uni-trier.de) is a computer science bibliography website. Nodes represent authors and edges represent co-authorship. As shown in [18], the edge probability is derived based on an exponential function based on the number of collaborations.

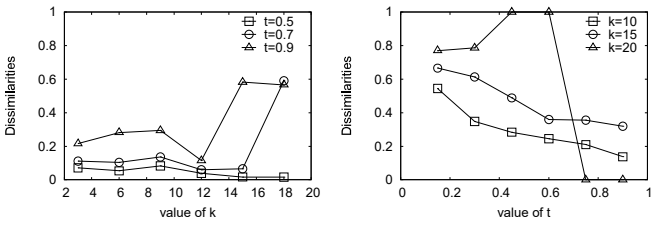
Email Eron (http://snap.stanford.edu) covers all the email communication within a dataset of around half million emails. This dataset is obtained by integrating data from the Stanford Network analysis Project.

Yelp (https://www.yelp.com.au/dataset/challenge) publishes crowd-sourced reviews about local businesses and the online reservation service Yelp Reservations. We use extracted data from [19]. In this extracted dataset, nodes represent users and edges represent friendship. For the Email Eron and yelp data, we set the edge probability by assigning a random value from the interval [0,1].

Settings. In experiments, the degree constraint k varies from 5 to 50 with default value 15. The probabilistic threshold θ ranges from 0.15 to 0.9, with default value 0.7. By default, we set $\epsilon = 0.1$ and $\delta = 0.1$. All programs were implemented in standard C++ and compiled with Visual Studio 2017. All experiments were performed on a machine with Intel Core i5-6500 3.20 GHz and 16GB DDR3-RAM in Windows 10 Pro.

B. Probabilistic K-core model comparison

We evaluate the *dissimilarity* of three probabilistic k-core models based on the *Jaccard distance* of the corresponding k-core nodes. Given two sets of nodes X and Y, the Jaccard distance between two nodes is defined as $1 - \frac{|X \cap Y|}{|X \cup Y|}$. Higher values indicate larger difference between two sets.



(a) Flickr (b) DBLP

Fig. 8. Differences between (k,θ)-core and (k,η)-core

Result comparison with (k,η)-core and expected k-core models. We compare the differences between (k,θ)-core and (k,η)-core in Fig. 8 by reporting the dissimilarity of the k-core nodes obtained from two models. Two graphs, Flickr and DBLP, are deployed for comparison. We increase the degree constraint k and probabilistic threshold t, where $\theta = t$ and $\eta = t$ in (k,θ)-core and (k,η)-core models, respectively. It is reported that, for the same k and probabilistic threshold value, the nodes included by two models do not have large overlaps.

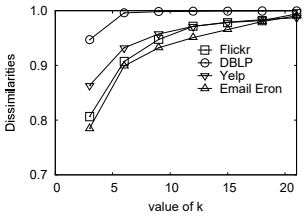


Fig. 9. Comp. expected model

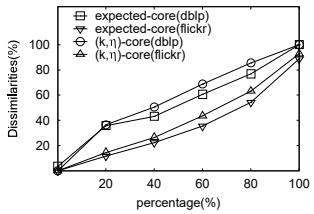


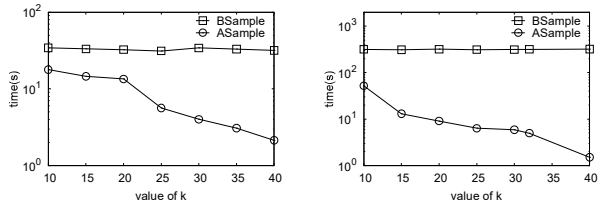
Fig. 10. Diff. parameter settings

To compare the k-core result of (k,θ)-core with *expected* k-core model, we apply the possible world semantics to the expected k-core model in a similar way with (k,θ)-core model in Equation 2. Let $cn(u, G)$ denote the core number of u in an instance graph G, then we have the expected core number of u, denoted by $\mathbb{E}(cn(u), \mathcal{G})$ is

$$\mathbf{E}(cn(u), \mathcal{G}) = \sum_{all\ G \sim \mathcal{G}} p(G) cn(u, G) \tag{7}$$

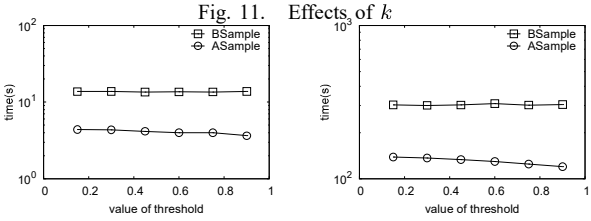
Then for a given k, the new expected k-core model includes nodes $\{u\}$ with $\mathbf{E}(cn(u), \mathcal{G}) \geq k$. Fig. 9 shows that the result comparison of two expected k-core models are very different on four datasets, especially when k grows.

We also evaluate core members of three models under different setting of k and probability threshold values. Particularly, let $\mathcal{A}$ (resp. $\mathcal{B}$) denote a set of (k,θ)-core (resp. (k,η)-core) results on DBLP and Flickr datasets by varying k from 10 to 20 and θ (η) from 0.05 to 0.95. We calculate the pairwise Jaccard distances, denoted by $\{d\}$, for every pair of sets from $\mathcal{A}$ and $\mathcal{B}$, and report the 0% (*min*), 20%, 40%, 60%, 80% and 100% (*max*) ranked Jaccard distances. Similarly, we also compare expected k-core model with (k, θ)-core model by varying k from 10 to 20. Fig. 10 shows that different models may end up with very overlapping result sets[8]. as well as very small overlapping results for different parameter settings. Nevertheless, the overall difference is significant between (k,θ)-core and other two local models.



(a) Flickr (b) DBLP

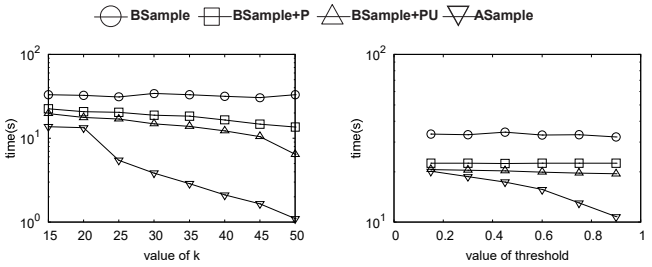
Fig. 11. Effects of k



(a) Email Eron (b) Yelp

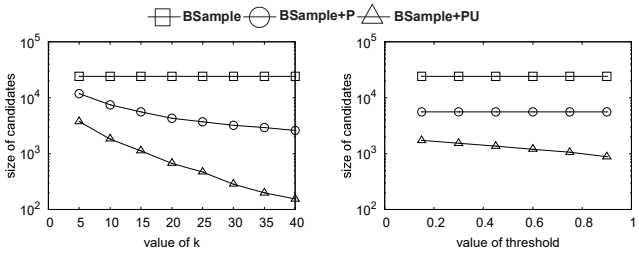
Fig. 12. Effects of θ

[8]For instance, the result of (k_1,θ)-core is the same as (k_2, η)-core on DBLP data when $k_1 = 14$, $\theta = 0.4$, $k_2 = 12$, and $\eta = 0.9$.



(a) Varying k (b) Varying θ

Fig. 13. Evaluation of proposed techniques



(a) Varying k (b) Varying θ

Fig. 14. Evaluation of pruning performance

C. Efficiency

In this subsection, we evaluate time efficiency of two algorithms (*BSample* and *ASample*) by reporting the CPU time for (k,θ)-core computation under different settings.

Fig. 11 reports running time of two algorithms on Flickr and DBLP with k varying from 10 to 40. It is shown that the performance of *BSample* has been significantly outperformed by that of *ASample* due to the lack of efficient pruning techniques and k-core computing algorithms in sample graphs. Under the same theoretical guarantee with *BSample*, *ASample* takes much less running time than that of *BSample*. This demonstrates the effectiveness and efficiency of our proposed pruning and k-core membership check techniques. *ASample* achieves better performance when k grows because the size of the candidate set R decreases upon the growth of k, and also leads to a few nodes to be verified for *ASample* Algorithm. Similar trend is also observed in Fig. 12, where Eron and Yelp are evaluated with probabilistic threshold θ ranging from 0.15 to 0.9. Note that a higher probabilistic threshold will lead to a smaller number of k-core candidates in set R.

We evaluate the techniques proposed in this paper by incrementally including them. By *BSample-P*, we denote the *BSample* algorithm with deterministic k-core pruning techniques which remove the nodes not in k-core of the deterministic graph G_0. By *BSample-PU*, we denote the *BSample-P* algorithm with upper bound techniques. Note that our k-core membership check algorithm (Algorithm 4) is not used in *BSample-P* and *BSample-PU*. Fig. 13 reports running time of four algorithms (*BSample*, *BSample-P*, *BSample-PU*, and *ASample*) on Flickr graph regarding different k and θ values. It is shown that all techniques make their contributions to improve the performance of *ASample* Algorithm. Particularly, the contribution of k-core membership check technique becomes more significant for large k and θ values due to the advantage of smaller number of nodes to be verified.

Fig. 14 evaluates the pruning power of our proposed pruning techniques by reporting the size of (k,θ)-core candidate set size. It is shown that both deterministic k-core pruning techniques and our upper bounds based pruning techniques can significantly reduce the candidate size.

Fig. 15 reports the running time of *ASample* against the number of samples on Flickr with $k = 10$ and θ. We also report the running time of (k,η)-core ($\eta = 0.8$) and expected k-core algorithms. As expected, the performance of *ASample* degrades linearly with the number of samples and two local models have better efficiency, especially the expected k-core model. It is worth noting that the result of *ASample* tend to be quite stable after sample size is larger than 200. For instance, the Jaccard distance of the results between sample sizes 200 and $4,000$ is only 0.02.

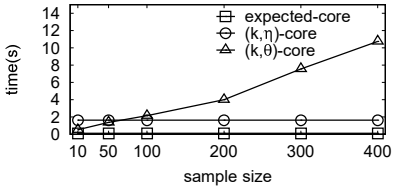


Fig. 15. Running time of three probabilistic k-core models.

D. Case Studies

To compare probabilistic k-core models in real-life applications, we use two case studies to evaluate the influence and engagement of the users with three k-core models.

Engagement. We use public available social network data *Brightkite* (http://snap.stanford.edu/data/loc-brightkite.html) to evaluate the engagement of the users. It consists of $58,228$ nodes, $214,078$ edges, and $4,491,143$ checkins during April 2008 and October 2010. We assume a user turns to inactive (i.e., disengaged) after his/her last checkin. Particularly, we generate probability of each edge based on the Jaccard similarity between the neighborhood of two edge nodes. We first retrieve 818 users for expected k-core model with k=6, then find (k,θ)-core (k=7 and θ=0.5) and (k,η)-core (k=5 and η=0.8) with similar sizes 722 and 733, respectively. And the common users of three models are removed for better comparison. Fig. 16 reports the percentage of users remaining engaged after August 2009 in every two months. It is shown that (k,θ)-core model can better capture user engagement compared to other two models.

Influence spread. We use a public available directed Twitter graph (http://snap.stanford.edu/data/egonets-Twitter.html) with $81,306$ nodes and $1,768,149$ edges) to valuate the influence spread for three uncertain k-core models. Following the weighted cascade model [11], we set probability of an edge $e(u,v)$ as $\frac{1}{d_{in}(v)}$ where d_{in} is the in-degree a node. We use the popular independent cascade (IC) model to evaluate the influence spread of an user. The influence of a node is the average number of reachable nodes in all possible worlds, which is a global model in this sense. To apply k-core models on Twitter graph, we only consider the out degree of the nodes, and boost the probabilities of edges by taking square root for a wider range of possible k values during the k-core computation. We choose the users within expected k-core with k varying from 6 to 9, and tune the k and probability

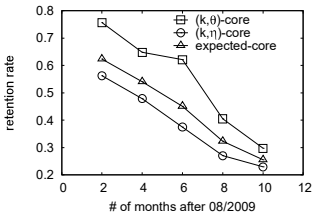


Fig. 16. Diff. Engagement

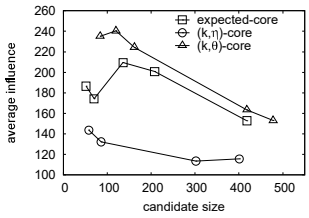


Fig. 17. Diff. Influence

thresholds values to achieve similar size. Fig. 17 reports the average influence of the core members for three models with respect to the size. It is shown that (k,θ)-core model can better capture the nodes with high influence, specially when the desired number of core members is small.

E. Related Work

In this section, we review previous work which is closely relevant to our problem.

Uncertain Graphs. A large number of classical graph problems have been studied in the context of uncertain graphs. For instance, Jin et al. [13] investigate the distance-constraint reachability problem in uncertain graph. Potamias et al. [18] introduce a framework which can address k nearest neighbors (kNN) queries on uncertain graphs. The problem of reverse kNN on uncertain graph is investigated in [20]. Papapetrou et al.[21] consider discovering and mining frequent patterns in uncertain graph. Recently, truss decomposition of probabilistic graphs are studied by [22].

K-Core Computation. Seidman [23] first introduced k-core computation which is a fundamental graph problem. An efficient linear-time in-memory algorithm is proposed by Batagelj and Zaversnilk [24] for core decomposition. For large graphs, Wen et al. [25] and Cheng et al. [26] propose I/O efficient algorithm for core number computation. Distributed algorithm is proposed by [27]. K-core problem is also studied in weighted graphs [7] and multi-dimensional graphs [28]. As to our best knowledge, [6] is the only existing work on uncertain k-core whose model is inherently different from ours.

VII. Conclusion

In this paper, we study the problem of probabilistic k-core computing on uncertain graphs. We propose a new k-core model, namely (k,θ)-core, based on the well-known possible world semantics. Particularly, we aim to find important nodes which contribute to the k-core communities with high probabilities on the uncertain graph. By semantic analysis and empirical study, we show that the new model provides a new perspective to model the k-core on uncertain graphs. We show the problem is NP-hard, and then efficient pruning techniques are proposed to significantly reduce the candidate size. We also develop an interesting k-core membership check algorithm to further speed up the computation. Extensive experiments on real-life graphs show that the our proposed techniques can significantly improve the efficiency of the baseline solutions.

Acknowledgments. Ying Zhang is supported by ARC DE140100679 and DP170103710. Wenjie Zhang is supported by ARC DP180103096. Xuemin Lin is supported by NSFC 61672235, DP170101628 and DP180103096. Lu Qin is supported by ARC DP160101513.

References

[1] J. Ugander, L. Backstrom, C. Marlow, and J. Kleinberg, "Structural diversity in social contagion," *PNAS*, vol. 109, no. 16, pp. 5962–5966, 2012.

[2] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley, and H. A. Makse, "Identification of influential spreaders in complex networks," *Nature physics*, vol. 6, no. 11, pp. 888–893, 2010.

[3] C. Giatsidis, F. Malliaros, D. M. Thilikos, and M. Vazirgiannis, "Corecluster: A degeneracy based graph clustering framework," in *IAAI*, 2014.

[4] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "Learning influence probabilities in social networks," in *WSDM*, 2010, pp. 241–250.

[5] S. Asthana, O. D. King, F. D. Gibbons, and F. P. Roth, "Predicting protein complex membership using probabilistic network reliability," *Genome research*, vol. 14, no. 6, pp. 1170–1175, 2004.

[6] F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich, "Core decomposition of uncertain graphs," in *SIGKDD*. ACM, 2014, pp. 1316–1325.

[7] A. Garas, F. Schweitzer, and S. Havlin, "A k -shell decomposition method for weighted networks," *New Journal of Physics*, vol. 14, no. 8, p. 083030, 2012.

[8] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, and E. Shir, "A model of internet topology using k-shell decomposition," *PNAS*, vol. 104, no. 27, pp. 11 150–11 154, 2007.

[9] S. Wu, A. D. Sarma, A. Fabrikant, S. Lattanzi, and A. Tomkins, "Arrival and departure dynamics in social networks," in *WSDM*, 2013.

[10] L. Chang, J. X. Yu, L. Qin, X. Lin, C. Liu, and W. Liang, "Efficiently computing k-edge connected components via graph decomposition," in *ACM SIGMOD*, 2013.

[11] D. Kempe, J. M. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *ACM SIGKDD*, 2003.

[12] A. P. Mukherjee, P. Xu, and S. Tirthapura, "Enumeration of maximal cliques from an uncertain graph," *IEEE TKDE*, vol. 29, no. 3, pp. 543–555, 2017.

[13] R. Jin, L. Liu, B. Ding, and H. Wang, "Distance-constraint reachability computation in uncertain graphs," *Proceedings of the VLDB Endowment*, vol. 4, no. 9, pp. 551–562, 2011.

[14] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[15] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American statistical association*, vol. 58, no. 301, pp. 13–30, 1963.

[16] J. Galambos and I. Simonelli, *Bonferroni-type inequalities with applications*. Springer Verlag, 1996.

[17] E. M. Stein and R. Shakarchi, *Real analysis: measure theory, integration, and Hilbert spaces*. Princeton University Press, 2009.

[18] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios, "K-nearest neighbors in uncertain graphs," *PVLDB*, vol. 3, no. 1-2, pp. 997–1008, 2010.

[19] F. Zhang, W. Zhang, Y. Zhang, L. Qin, and X. Lin, "Olak: an efficient algorithm to prevent unraveling in social networks," *Proceedings of the VLDB Endowment*, vol. 10, no. 6, pp. 649–660, 2017.

[20] Y. Gao, X. Miao, G. Chen, B. Zheng, D. Cai, and H. Cui, "On efficiently finding reverse k-nearest neighbors over uncertain graphs," *VLDB J.*, vol. 26, no. 4, pp. 467–492, 2017.

[21] O. Papapetrou, E. Ioannou, and D. Skoutas, "Efficient discovery of frequent subgraph patterns in uncertain graph databases," in *Proceedings of the 14th International Conference on Extending Database Technology*. ACM, 2011, pp. 355–366.

[22] X. Huang, W. Lu, and L. V. Lakshmanan, "Truss decomposition of probabilistic graphs: Semantics and algorithms," in *SIGMOD*, 2016, pp. 77–90.

[23] S. B. Seidman, "Network structure and minimum degree," *Social networks*, vol. 5, no. 3, pp. 269–287, 1983.

[24] V. Batagelj and M. Zaversnik, "An o (m) algorithm for cores decomposition of networks," *arXiv preprint cs/0310049*, 2003.

[25] D. Wen, L. Qin, Y. Zhang, X. Lin, and J. X. Yu, "I/o efficient core graph decomposition at web scale," in *ICDE*, 2016.

[26] J. Cheng, Y. Ke, S. Chu, and M. T. Özsu, "Efficient core decomposition in massive networks," in *ICDE*, 2011.

[27] A. Montresor, F. De Pellegrini, and D. Miorandi, "Distributed k-core decomposition," *IEEE Transactions on parallel and distributed systems*, vol. 24, no. 2, pp. 288–300, 2013.

[28] F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, "When engagement meets similarity: Efficient (k, r)-core computation on social networks," *PVLDB*, vol. 10, no. 10, pp. 998–1009, 2017.