# Approximation Algorithms for Probabilistic Graphs

#### Kai Han

School of Computer Science and Technology, University of Science and Technology of China, P.R.China hankai@ustc.edu.cn

#### **ABSTRACT**

We study the k-median and k-center problems in uncertain graphs. We analyze the hardness of these problems, and propose several algorithms with improved approximation ratios compared with the existing proposals.

#### **ACM Reference Format:**

Kai Han. 2018. Approximation Algorithms for Probabilistic Graphs. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnnnnnnnnnnn

#### 1 INTRODUCTION

Graph data are prevalent in a lot of application domains such as social, biological and mobile networks. Typically, the entities in realities are modeled by graph nodes, and the relationships between entities are modeled by graph edges. Uncertainty is evident in graph data due to a variety of reasons. Therefore, the methods for querying and mining uncertain graph data are of paramount importance.

Graph clustering is a fundamental problem in graph data mining, where the goal is to partition the graph nodes into some clusters, such that the nodes in each cluster is "close" to each other according to some distance measure. Among the numerous problem definitions on graph clustering, the k-median and k-center problems are perhaps the most celebrated ones which have been studied for decades [7]. In a traditional graph (without uncertainty), the goal of the k-median problem is to find k centering nodes in the network such that the average distance between each node to the centering nodes is maximized, while the goal of the k-center problem is to find a set of k nodes for which the largest distance of any point to its closest vertex in the k-set is minimum.

Surprisingly, although the k-median and k-center problems have been extensively studied in the literature, their counterpart problems in uncertain graphs have not been investigated until a recent study by Ceccarello et al. [1]. Following a large body of work on uncertain graphs, the work in [1] models an uncertain graph as a traditional graph augmented by existence probabilities associated to the edges. They use the connection probabilities as the distance measure between the nodes, and formulated the k-median and k-center problems as follows. In the k-median problem, they aim to partition the graph nodes into k subsets (clusterings) with a centering node in each of them, such that the average connection probability between each node and its corresponding centering node

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

is maximized. In the k-center problem, they aim to maximize the minimum connection connectivity between a node and its centering node. It can be seen that the definitions of their k-median and k-center problems are in spirit similar to those for the traditional graphs, so they can be considered as the reinterpretations of the k-median and k-center problems in traditional graphs.

In contrast to the traditional k-median and k-center problems, there are two unique challenges for clustering uncertain graphs. First, it is a #P hard problem to compute the connection probability between any two nodes in an uncertain graph. Second, the distance measure described above does obey the triangle inequality, which is required by almost all of the traditional k-center and k-median algorithms. Therefore, even if we have an oracle for computing the connection probabilities, the traditional k-median and k-center algorithms cannot be applied to our case.

Based on the above observations, the work in [1] provide new algorithms for graph clustering problem in uncertain graphs. However, the approximation ratio of their algorithms are far from satisfactory.

**Contributions.** Motivated by the deficiency of existing techniques, we propose new approximation algorithms for the k-median and k-center problems in uncertain graphs. Our contributions are summarized as follows.

(1) For the k-median problem:

We prove that the k-median problem is NP-hard, and propose an approximate algorithm with a 1-1/e approximation ratio. We also propose efficient sampling algorithms that achieves a  $1-1/e-\epsilon$  approximation ratio when there does not exist an oracle for computing the connection probabilities.

(2) For the k-center problem:

We prove that the k-center problem is NP-hard to approximate within any bounded ratio. We first propose a simple algorithm with the approximation ratio of  $\mathrm{OPT}_k^c$ , and then provide a bi-criteria approximation algorithm that achieve  $1-\epsilon$  approximation ratio using at most  $O(k\log\frac{n}{\epsilon})$  centering nodes. We also propose algorithms for the k-center problem without the connection oracle.

### 2 PRELIMINARIES

#### 2.1 Problem Definitions

An uncertain graph is represented by G = (V, E) where V is the set of nodes and E is the set edges, with |V| = n and |E| = m. We assume that each node in V has a unique node ID in [1, n]. Each edge  $e \in E$  is associated with a number  $p(e) \in (0, 1]$  denoting the probability that e exists. For any two nodes u and v in V, we use  $\Pr[u \sim v]$  to denote the probability that u and v is connected in G. For simplicity, we follow the work in [1] to assume that G is an undirected graph, but our approach can be readily extended to the case of directed graphs, which will be explained later.

A k-clustering of G can be represented by a tuple  $C = \langle C, Q_1, Q_2, \dots, Q_k \rangle$  Algorithm 1: Greedy $(G, k, q(\cdot))$ where  $C = \{c_1, \dots, c_k\}$  is the set of centering nodes and  $\{Q_1, Q_2, \dots, Q_k\}$ is a partition of the nodes in V satisfying  $c_i \in Q_i$  for all  $i \in [k]$ . For any  $i \in [k]$  and any  $v \in Q_i$ , we call the node pair  $(c_i, v)$  as a cluster link of C. We call the set of all clustering links in C as the *signature* of C, and we use  $\mathcal{S}_k^G$  to denote the set of signatures of all possible k-clusterings of G. Note that any two different kclusterings must have different signatures, and we can construct a k-clustering from any  $A \in \mathcal{S}_k^G$ . Therefore, we will also call any  $A \in \mathcal{S}_k^G$  as a k -clustering. Given any  $A \in \mathcal{S}_k^G$  , we define

$$KM(A) = \sum_{(u,v) \in A} \Pr[u \sim v]/n \tag{1}$$

$$KC(A) = \min_{(u,v) \in A} \Pr[u \sim v]$$
 (2)

With the above definitions, the k-median and k-center problems can be formally defined as follows:

Definition 2.1. The k-median (KMD) problem aims to identify an optimal solution  $A^o$  to the following optimization problem:

Maximize 
$$KM(A)$$
 [KMD]  
s.t.  $A \in S_L^G$ 

The k-center (KCT) problem aims to identify an optimal solution  $B^{o}$  to the following optimization problem:

Maximize 
$$KC(B)$$
 [KCT]  
s.t.  $B \in \mathcal{S}_{k}^{G}$ 

For convenience, we use  $\mathsf{OPT}^m_k$  to denote  $\mathit{KM}(A^o),$  and use  $\mathsf{OPT}^c_k$ to denote  $KC(B^o)$ .

#### SOLVING THE k-MEDIAN PROBLEM 3

### Hardness of the k-Median Problem

The prior work [1] has conjectured that the k-median problem is NP-hard. We prove this conjecture in the following theorem, by a reduction from the NP-hard Dominating Set problem:

THEOREM 3.1. The k-median problem is NP-hard, even if there exists an oracle for computing  $\forall u, v \in V : \Pr[u \sim v]$ .

PROOF. We prove the theorem by a reduction from the NP-hard dominating set problem [7]. Given any undirected graph G = (V, E)with |V| = n and any integer k, the decision version of the dominating set problem asks whether there exists  $S \subseteq V$  with |S| = ksuch that each node in  $V \setminus S$  is adjacent to certain node in S. Given such an instance G of the dominating set problem, we can construct an uncertain graph by setting  $p(e) = q = \frac{1}{n(n-k+2)}$  for each  $e \in E$ . Suppose that there exists a polynomial-time algorithm  $\mathcal{A}_{opt}$ to optimally solve the k-median problem. So we can run it on the uncertain graph G described above, and get an optimal k-clustering with its signature denoted by  $\tilde{A}$ . In the sequel, we will prove that: the graph G has a dominating set S satisfying |S| = k if and only if  $KM(\tilde{A}) \ge k + (n-k)q$ 

If the graph G has a dominating set S satisfying |S| = k, then we can use S as the set of center nodes, and hence we must have  $KM(A) \ge k + (n-k)q$ . Conversely, if G does not have a dominating set S satisfying |S| = k, then there must exist a cluster link (u, v)

1 
$$C \leftarrow \emptyset$$
  
2 while  $|C| < k$  do  
3 | Find  $u \in V \setminus C$  such that  $g(C \cup \{u\}) - g(C)$  is maximized;  
4 |  $C \leftarrow C \cup \{u\}$   
5 return  $C$ 

in  $\tilde{A}$  such that u and v are not adjacent, and hence we get

$$\Pr[u \sim v] \le nq^2 + n^2q^3 + n^3q^4 + \dots \le \frac{nq^2}{1 - nq},\tag{3}$$

where  $n^i q^{i+1}$  is an upper bound for the probability that u is connected to v through i+1 hops. Moreover, for any  $(u', v') \in \tilde{A} \setminus \{(u, v)\}$ satisfying  $u' \neq v'$ , we must have

$$\Pr[u' \sim v'] \le q + nq^2/(1 - nq)$$
 (4)

Therefore, we have

$$KM(\tilde{A}) \leq k + (n-k-1)\left(q + \frac{nq^2}{1-nq}\right) + \frac{nq^2}{1-nq}$$

$$< k + (n-k)q$$
(5)

The above reasoning implies that, if  $\mathcal{A}_{opt}$  exists, then the dominating set problem can also be optimally solved in polynomial time. Hence, the theorem follows.

### 3.2 k-Median Algorithms with an Oracle

In this section, we assume that there exists a connectivity oracle, i.e.,  $Pr[u \sim v]$  can be computed in polynomial time for any  $u \in V$ and  $v \in V$ .

It is highly non-trivial to find an approximation solution to the k-median problem, as it has a large searching space  $\mathcal{S}_k^G$  with the cardinality of  $\binom{n}{k}k^{n-k}$ . However, we find that the k-median problem can be transformed into a submodular maximization problem with a much-reduced searching space, as described below.

For any  $C \subseteq V$  and any  $v \in V$ , we define

$$f_v(C) = \max\{\Pr[u \sim v] \mid u \in C\}; \ F(C) = \sum_{v \in V} f_v(C)$$
 (6)

It is noted that, for any  $A \in \mathcal{S}_k^G$ , there must exist certain  $C \subseteq V$ such that |C| = k and  $F(C) \ge K\widetilde{M}(A)$ . Moreover, given any  $C \subseteq V$ , we can easily construct a k-clustering A such that C is the set of centering nodes in A and F(C) = KM(A). Therefore, the k-Median problem can be transformed into the following equivalent optimization problem:

Moreover, we find that the [KMD1] problem is actually a submodular maximization problem, as shown by the following theorem:

Theorem 3.2. For any  $v \in V$ , the function  $f_v(\cdot)$  is a monotone and submodular function defined on  $2^{V}$ .

It is a well-known fact that monotone submodular maximization problems can be addressed by a greedy algorithm with a 1 - 1/eapproximation ratio. Therefore, we can use a greedy algorithm

#### **Algorithm 2:** SearchKM( $G, k, \mathcal{R}$ )

- $1 \ C^* \leftarrow \mathsf{Greedy}(G, k, \widehat{F}(\mathcal{R}, \cdot)); \ A^* \leftarrow D(\mathcal{R}, C^*)$
- 2 return A\*

(shown in Algorithm 1) to find a 1-1/e approximation to [KMD1] and hence to the k-median problem. i.e.:

Theorem 3.3. Using the  $Greedy(G, k, F(\cdot))$  algorithm, we can find a solution with a 1-1/e approximation ratio ti the k-median problem

Besides, we can also provide a 1/k approximation to the k-median problem. Consider the following problem:

We have the following theorem:

Theorem 3.4. Suppose that  $C^{\dagger}$  is an optimal solution to the [KMD2] problem. We have  $F(C^{\dagger}) \geq (1/k)KM(A^{o})$ .

Proof. Suppose that  $C_{km}^o$  is the set of centering nodes in  $A^o$ . We have

$$\begin{split} F(C^{\dagger}) &= \sum_{v \in V} \max\{\Pr[u \sim v] \mid u \in C^{\dagger}\} \\ &\geq \sum_{v \in V} (1/k) \sum_{u \in C^{\dagger}} \Pr[u \sim v] \\ &\geq (1/k) \sum_{v \in V} \sum_{u \in C^{o}_{km}} \Pr[u \sim v] \\ &\geq (1/k) KM(A^{o}) \end{split}$$

Hence, the theorem follows.

#### 3.3 k-Median Algorithms without Oracle

In this section, we consider a more practical setting where the connection oracle is absent. We will first provide a basic sampling algorithm to address the k-median problem, and then provide some more efficient algorithms.

3.3.1 A Basic Sampling Algorithm for k-Median. A random sample R of G is a graph generated by removing each edge e in G with the probability of 1-p(e). For any  $u,v\in V$  and any random sample R of G, let  $X_R(u\sim v)=1$  when u and v is connected in R, and  $X_R(u\sim v)=0$  when u and v is not connected in R. For any set R of random samples of G, define

$$\begin{split} \widehat{\Pr}[\mathcal{R}, u \sim v] &= \sum_{R \in \mathcal{R}} X_R(u \sim v) / |\mathcal{R}| \\ \widehat{KM}(\mathcal{R}, A) &= \sum_{(u, v) \in A} \widehat{\Pr}[\mathcal{R}, u \sim v] / n \end{split}$$

It can be seen that  $\widehat{\Pr}[\mathcal{R}, u \sim v]$  and  $\widehat{KM}(\mathcal{R}, A)$  are unbiased estimations of  $\Pr[u \sim v]$  and KM(A), respectively. Similarly, for any  $A \in \mathcal{S}_k$ , any  $C \subseteq V$  and any  $v \in V$ , we define

$$\begin{split} \widehat{f}_{v}(\mathcal{R},C) &= \max\{\widehat{\Pr}[\mathcal{R},u \sim v] \mid u \in C\} \\ \widehat{F}(\mathcal{R},C) &= \sum_{v \in V} \widehat{f}_{v}(\mathcal{R},C) \end{split}$$

As computing the connection probabilities is NP-hard, it is hard to find the cluster links even if we know the set of centering nodes in an optimal solution. To bypass this difficulty, we create a mapping between  $V_k = \{C|C \subseteq V \land |C| = k\}$  and  $\mathcal{S}_k^G$  to reduce the number of generated samples for identifying the cluster links. More specifically, given any set  $\mathcal{R}$  of random samples, each node set  $C \in V_k$  is mapped to a unique k-clustering  $D(\mathcal{R},C)$  in  $\mathcal{S}_k^G$ , such that any cluster link  $(c,v) \in D(\mathcal{R},C)$  (with  $c \in C$ ) that satisfies: 1)  $\widehat{\Pr}[\mathcal{R},c \sim v] = \widehat{f_v}(\mathcal{R},C)$ ; 2) The node ID of c is minimized under condition 1).

With the above definitions, we design the SearchKM algorithm to find an approximate solution. Given a set  $\mathcal{R}$  of random samples, the SearchKM algorithm first calls the Greedy algorithm to find a set  $C^*$  of center nodes, and then returns  $A^* = D(\mathcal{R}, C^*)$  as an approximate solution. To ensure that  $A^*$  has a good approximation ratio, we give the following theorem to determine a upper-bound for the number of generated samples:

Theorem 3.5. If  $|\mathcal{R}| \geq T_{max}$  where

$$T_{max} = \left\lceil \frac{2(2e-1)(e\epsilon+2e-1)}{3e^2\epsilon^2 \mathrm{OPT}_k^m} \ln \frac{\binom{n}{k}+1}{\delta} \right\rceil = O\left(\frac{n}{\epsilon^2} \log \frac{n}{\delta}\right),$$

then the SearchKM(G, k, R) algorithm returns a  $(1-1/e-\epsilon)$ -approximate solution  $A^*$  to the k-median problem with probability of at least  $1-\delta$ .

3.3.2 Accelerations. The SearchKM algorithm can be further accelerated by leveraging the submodularity of the function  $\widehat{F}(\mathcal{R},\cdot)$ , as shown by the SearchKM+ algorithm. The SearchKM+ algorithm maintains a value UB(v) for each  $v\in V$ , which denotes an upper bound for the marginal gain of v with respect to the currently selected node set  $C^*$ . Initially, SearchKM+ calls GetFirstNode( $\mathcal{R}$ ) to calculate  $UB(v) = \widehat{F}(\mathcal{R}, \{v\})$  for all  $v\in V$ , and then add  $u^* = \arg\max_{u\in V} UB(v)$  into  $C^*$ . After that, it sorts V into the node list V according to the non-increasing order of  $\forall v\in V: UB(v)$ , and re-compute UB(v) only when necessary.

It can be seen that the idea of SearchKM+ is similar in spirit to the "lazy greedy" algorithm proposed in [5]. However, the lazy greedy algorithm has not considered the "cold start" problem, i.e., how to efficiently compute the upper bound of the marginal gain of any node in V (i.e., UB(v)) in the initialization phase. In our case, a naive approach for computing the initial value of UB(v) requires O(n) time for any  $v \in V$ , as we need to calculate  $\widehat{f}_u(\mathcal{R}, \{v\})$  for each  $u \in V$ . However, using the GetFirstNode procedure, we only need O(1) time to compute UB(v).

The SearchKM+ algorithm can be further accelerated by borrowing some ideas from the OPIM sampling framework proposed in [6]. The resulted algorithm is shown in Algorithm ??. We can prove:

Theorem 3.6. With probability of at least  $1-\delta$ , the Sampling KM algorithm returns a k-clustering with a  $1-1/e-\epsilon$  approximation ratio. The expected number of random samples generated in Sampling KM is at most  $O(\frac{1}{\epsilon^2 \mathrm{OPT}_{\nu}^m} \ln \frac{1}{\delta})$ .

#### 4 SOLVING THE K-CENTER PROBLEM

In this section, we address the k-center problem both with a connection oracle assumption and without it.

#### **Algorithm 3:** SearchKM+(G, k, R)

- $1(u, W) \leftarrow \text{GetFirstNode}(\mathcal{R}); C^* \leftarrow \{u\}$
- 2 **while**  $|C^*| < k$  **do**
- $(u, W) \leftarrow \mathsf{GetNextNode}(W, \widehat{F}(\mathcal{R}, \cdot), C^*)$
- $C^* \leftarrow C^* \cup \{u\}$
- 5 return C\*

#### **Algorithm 4:** GetFirstNode( $\mathcal{R}$ )

- 1 foreach  $v \in V$  do
- $UB(v) \leftarrow$  the summation of the sizes of the connected components in  $\mathcal{R}$  that contain v;
- $_3$  Sort V into the node list W according to the non-increasing order of  $UB(v) : v \in W$ ;
- 4 Remove the first node  $w_1$  from W;
- 5 return  $(w_1, W)$

### **Algorithm 5:** GetNextNode(W, $g(\cdot)$ , C)

- 1 for  $i \leftarrow 1$  to |W| do
- $UB(w_i) \leftarrow g(C \cup \{w_i\}) g(C)$
- if  $UB(w_i) \ge UB(w_{i+1})$  then break;
- 4 Re-sort the nodes in W according to the non-increasing order of  $UB(v): v \in W$
- 5 Remove the first node  $w_1$  from W;
- 6 return  $(w_1, W)$

### **Algorithm 6:** SamplingKM( $G, k, \epsilon, \delta$ )

- 1  $T_{max} \leftarrow \frac{2(2e-1)(e\epsilon+2e-1)n}{3e^2\epsilon^2k} \ln \frac{\binom{n}{k}+1}{\delta}$ ;  $T \leftarrow T \cdot \epsilon^2k/n$ 2 Generate two sets  $\mathcal{R}_1$  and  $\mathcal{R}_2$  of random samples of G, such
- that  $|\mathcal{R}_1| = |\mathcal{R}_2| = T$ ;
- $i_{max} \leftarrow \lceil \log_2(T_{max}/T) \rceil;$
- 4 for  $i \leftarrow 1$  to  $i_{max}$  do

4 for 
$$i \leftarrow 1$$
 to  $i_{max}$  do

5  $A^* \leftarrow \text{SearchKM}(G, k, \mathcal{R}_1)$ 

6  $a \leftarrow \ln(3i_{max}/\delta); \ \theta \leftarrow |\mathcal{R}_1|$ 

7  $\ln(A^*) \leftarrow \left(\sqrt{\widehat{KM}(\mathcal{R}_2, A^*)} - \sqrt{\frac{a}{6\theta}}\right)^2 - \frac{a}{6\theta}$ 

8  $\ln(A^o) \leftarrow \left(\sqrt{\frac{\widehat{KM}(\mathcal{R}_1, A^*)}{1 - 1/e}} + \frac{2a}{3\theta} + \sqrt{\frac{a}{6\theta}}\right)^2 - \frac{a}{6\theta}$ 

9  $\ln(A^o) \leftarrow \left(\sqrt{\frac{\widehat{KM}(\mathcal{R}_1, A^*)}{1 - 1/e}} + \frac{2a}{3\theta} + \sqrt{\frac{a}{6\theta}}\right)^2 - \frac{a}{6\theta}$ 

10  $\ln(A^*)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

11  $\ln(A^*)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

12  $\ln(A^*)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

13  $\ln(A^*)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

14  $\ln(A^*)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

15  $\ln(A^*)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

16  $\ln(A^*)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

17  $\ln(A^*)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

18  $\ln(A^*)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

19  $\ln(A^*)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

19  $\ln(A^*)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

10  $\ln(A^*)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

10  $\ln(A^*)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

11  $\ln(A^*)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

12  $\ln(A^*)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

13  $\ln(A^o)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

14  $\ln(A^o)/\ln(A^o)/\ln(A^o) \ge 1 - 1/e - \epsilon \text{ or } i = i_{max} \text{ then}$ 

15  $\ln(A^o)/\ln(A^o)/\ln(A^o)/\ln(A^o)/\ln(A^o)/\ln(A^o)/\ln(A^o)$ 

16  $\ln(A^o)/\ln(A^o)$ 

### 4.1 k-Center Algorithms with an Oracle

4.1.1 A Simple Algorithm. The work in [1] has proved that the connection probabilities of any three nodes  $u, v, w \in V$  must satisfy

$$\Pr[u \sim w] \ge \Pr[u \sim v] \cdot \Pr[v \sim w] \tag{7}$$

Let  $d(u, v) = -\ln \Pr[u \sim v]$  and  $d_v(C) = \min_{u \in C} d(u, v)$ , so we

$$d(u, w) \le d(u, v) + d(v, w) \tag{8}$$

which implies that  $d(\cdot)$  is a metric. Consider the following problem:

It can be seen that the set of centering nodes in  $B^{o}$  is also an optimal solution to the [KCT0] problem. Note that [KCT0] is a metric k-center problem, so it can be addressed by a simple greedy algorithm with a 2 approximation ratio [8]. More specifically, the greedy algorithm initializes by selecting an arbitrary node, and then iteratively selects a node which is furthest to the currently selected nodes until k nodes are selected. With this greedy algorithm, we can find  $B^* \in \mathcal{S}_k^G$  such that  $-\ln KC(B^*) \le -2\ln KC(B^o)$ , which implies that

$$KC(B^*) \ge (OPT_k^c)^2$$
 (9)

4.1.2 A Bi-Criteria Approximation Algorithm. Note that the approximation ratio proposed by (9) can be arbitrarily bad, as  $OPT_{L}^{c}$ can be arbitrarily small. Therefore, we ask whether there exists an algorithm with a bounded approximation ratio for the k-center problem. Unfortunately, we find that:

THEOREM 4.1. Unless P=NP, no polynomial-time algorithm can find a solution to the k-center problem within any approximation ratio  $\alpha > 0$ , even if there exists a connectivity oracle.

As the *k*-center problem is NP-hard to approximate, we further ask the question whether there exists a bi-criteria approximation algorithm for it, i.e., we permit such an algorithm to use more than k center nodes, such that it can approach  $OPT_{k}^{c}$ . However, the following theorem reveals that, we cannot achieve a large connectivity probability unless we allow the usage of a "sufficiently large" number of centering nodes:

THEOREM 4.2. Unless P=NP, no algorithm can find a l-clustering B in polynomial time, such that  $KC(B) \ge OPT_{L}^{c}$  and  $l < k \ln n$ .

Based on Theorem 4.2, we propose a bi-criteria approximation algorithm with nearly tight approximation ratios. First, we re-formulate the [KCT] problem into the following [KCT1] problem:

$$\begin{aligned} & \mathbf{Maximize} & & & \min_{v \in V} f_v(C) \\ & & \mathbf{s.t.} & & |C| = k; \ C \subseteq V \end{aligned}$$

It can be seen that, for any k-clustering  $B \in \mathcal{S}_k^G$ , we must have  $KC(B) \leq \min_{v \in V} f_v(C_B)$ , where  $C_B$  denotes the set of centering nodes in B. Therefore, the [KCT1] problem is equivalent to the [KCT] problem.

Recall that the  $f_v(\cdot)$  is monotone and submodular for any  $v \in V$ . Therefore, the [KCT1] problem is similar to the "robust submodular maximization" problem studied in [4]. However, the algorithms and performance bounds proposed in [4] are only suitable for the case where the considered submodular function is integer-valued, while the function  $f_v(\cdot)$  in our case is generally non-integral. Therefore, we adapt the algorithms proposed in [4] to our case and prove new performance bounds, as described in the following.

### **Algorithm 7:** SearchKC( $\epsilon_1, \epsilon_2, f_v(\cdot)$ ) $[q_1,q_2] \leftarrow [0,1]; C^* \leftarrow \emptyset$ 2 repeat $\begin{array}{l} C \leftarrow \emptyset; \ q \leftarrow \frac{q_1 + q_2}{2} \\ L(q, \cdot) \leftarrow \sum_{v \in V} \min\{q, f_v(\cdot)\} \end{array}$ 3 4 $v^* \leftarrow \arg\max_{u \in V \setminus C} [L(q, C \cup \{u\}) - L(q, C)];$ $C \leftarrow C \cup \{v^*\}$ if $|C| > \lceil \ln \frac{n}{\epsilon_1} \rceil k$ then break; **until** $L(q, C) \ge nq - \epsilon_1 q$ ; if $|C| \leq \lceil \ln \frac{n}{\epsilon_1} \rceil k$ then 10 $C^* \leftarrow C; q_1 \leftarrow q$ 11 12 13 $q_2 \leftarrow q$ 14 **until** $q_1 \ge (1 - \epsilon_2)q_2$ ; 15 return C\*

Our algorithm is based on a "potential function" L defined as follows:

$$\forall q \in (0,1], \forall C \subseteq V : L(q,C) = \sum_{v \in V} \min\{q, f_v(C)\}$$
 (10)

As  $f_{\upsilon}()$  is a submodular function, it can be verified that  $L(q,\cdot)$  is also a submodular function for any  $q\in(0,1]$ . Moreover, the function L has a remarkable property that it can be used to find an upper bound of  $\mathrm{OPT}_k^c$ , as clarified by the following lemma:

Lemma 4.3. Let  $C = \text{Greedy}(\lceil \ln \frac{n}{\epsilon_1} \rceil k, L(q, \cdot))$ . If  $L(q, C) < nq - \epsilon_1 q$ , then q must be an upper bound of  $\text{OPT}_k^c$ .

Note that  $L(q,C) \geq nq - \epsilon_1 q$  implies that  $\min_{v \in V} f_v(C) \geq (1 - \epsilon_1)q$ . So Lemma 4.3 actually tells that, if  $q < \mathrm{OPT}_k^c$ , then we can use function L and the Greedy procedure to find a clustering B with at most  $\lceil \ln \frac{n}{\epsilon_1} \rceil k$  centering nodes such that  $\widehat{KC}(B) \geq (1 - \epsilon_1)q$ . Conversely, if such a clustering cannot be obtained, then we must have  $\mathrm{OPT}_k^c \leq q$ .

With Lemma 4.3, we can use a binary searching process to find an approximate solution to [KCT1], as shown by the SearchKC algorithm. In the SearchKC algorithm, we maintain a searching interval  $[q_1,q_2]$  (initialized to [0,1]), and use Lemma 4.3 to judge whether  $q=\frac{q_1+q_2}{2}$  is an upper bound of  $\mathrm{OPT}_k^c$ . If we find that  $\mathrm{OPT}_k^c \leq q$ , then we halves  $[q_1,q_2]$  by setting  $q_2=q$ . Otherwise, we also halves  $[q_1,q_2]$  by setting  $q_1=q$ . As such, we always have

$$\mathrm{OPT}_{k}^{c} \leq q_{2}; \ |C^{*}| \leq \lceil \ln \frac{n}{\epsilon_{1}} \rceil k; \min_{v \in V} f_{v}(C) \geq (1 - \epsilon_{1}) q_{1}$$
 (11)

throughout the binary searching process, where the last inequality is due to  $L(q_1, C) \ge nq - \epsilon_1 q$ . Note that the binary searching process stops when  $q_1 \ge (1 - \epsilon_2)q_2$ . So we immediately get the following theorem:

Theorem 4.4. For any  $\epsilon, \epsilon_1, \epsilon_2 \in (0,1)$  satisfying  $1 - \epsilon = (1 - \epsilon_1)(1 - \epsilon_2)$ , the SearchKC algorithm can find a solution that achieves  $(1 - \epsilon)\text{OPT}_k^c$ , using at most  $\left[\ln \frac{n}{\epsilon_1}\right]k$  centers. This algorithm has no more than  $\left[\log_2 \frac{1}{\epsilon_2 \cdot \text{OPT}_s^c}\right]$  iterations.

### 4.2 k-Center Algorithms without an Oracle

4.2.1 Approximation Algorithm using Sampling. In this section, we study whether the algorithm suggested in Sec. 4.1.1 can be implemented without a connection oracle. Define

$$\widehat{d}(\mathcal{R},u,v) = -\ln \widehat{\Pr}[\mathcal{R},u \sim v]; \ \widehat{d}(\mathcal{R},v,C) = \min_{u \in C} \widehat{d}(\mathcal{R},u,v)$$

With this definition, we propose an approximation algorithm as follows:

### **Algorithm 8:** SearchKC\_1(G, k, $\mathcal{R}$ )

- 1 Select an arbitrary node  $v \in V$  and add it into  $C^*$
- 2 **while**  $|C^*| < k$  **do**
- Find  $v^* \in V \setminus C^*$  such that  $\widehat{d}(\mathcal{R}, v^*, C^*)$  is maximized;
- ${}_{4} \quad C \leftarrow C \cup \{v^*\}$
- 5  $B^* \leftarrow D(\mathcal{R}, C^*)$
- 6 return  $C^*$ ,  $B^*$

Next, we study the problem of how to determine the cardinality of  $\mathcal R$  such that the k-clustering  $B^*$  returned by the SearchKC\_1 algorithm can achieve a good approximation ratio. We give the following theorem:

Theorem 4.5. Given any  $\epsilon$ ,  $\epsilon_1$ ,  $\epsilon_2$ ,  $\delta \in (0, 1)$  satisfying  $\epsilon = \epsilon_1 + \epsilon_2$ , and given any set  $\mathcal{R}$  of random samples of G satisfying

$$|\mathcal{R}| \geq \max \left\{ \frac{2(1+\epsilon_1)}{3\epsilon_1^2(\mathsf{OPT}_L^c)^2} \ln \frac{n(n-1)}{\delta}, \frac{2(1-\epsilon_1)}{3\epsilon_2^2(\mathsf{OPT}_L^c)^2} \ln \frac{n(n-1)}{\delta} \right\},\,$$

the SearchKC\_1(G, k,  $\mathcal{R}$ ) can return a k-clustering  $B^*$  satisfying  $KC(B^*) \ge (1 - \epsilon)(\mathrm{OPT}_k^c)^2$  with probability of at least  $1 - \delta$ .

As  $\mathsf{OPT}_k^c$  is unknown, we present an algorithm that iteratively "guesses"  $\mathsf{OPT}_k^c$  until a good solution is found, as shown by Algorithm 9.

4.2.2 Sampling for Bi-Criteria Approximation. A straightforward idea is that, we first generate a set  $\mathcal R$  of random samples, and then call the SearchKC algorithm by replacing the function  $f_{\mathcal V}(\cdot)$  by  $\widehat f_{\mathcal V}(\mathcal R,\cdot)$ . After the SearchKC algorithm returns a set  $C^*$  of centering nodes, we use  $B^*=D(\mathcal R,C^*)$  as an approximate solution to the k-center problem. Clearly, if  $|\mathcal R|$  is sufficiently large, then  $B^*$  should achieve an approximation ratio close to that we can get with an connectivity oracle. The key problem in this approach, however, is how to determine the cardinality of  $\mathcal R$ . In the following theorem, we propose an upper bound for the number of random samples needed to be generated:

Theorem 4.6. Let  $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon$  and  $\delta$  be any numbers in (0, 1) that satisfy  $(1 - \epsilon)(1 + \epsilon_3) = (1 - \epsilon_1)(1 - \epsilon_2)(1 - \epsilon_3)$ . Let  $\mathcal R$  be any set of random samples of G such that

$$|\mathcal{R}| \ge \frac{2(1+\epsilon_3)}{3\epsilon_3^2(1-\epsilon)\text{OPT}_k^c} \ln \frac{n^2+n-2k}{2\delta}.$$

Then, we can use the SearchKC algorithm to find a k-clustering  $B^*$  with no more than  $\lceil \ln \frac{n}{\epsilon_1} \rceil k$  centering nodes such that  $KC(B^*) \ge (1 - \epsilon) \mathrm{OPT}^c_k$  with probability of at least  $1 - \delta$ .

14 **return**  $(C^*, B^*)$ 

## **Algorithm 9:** SamplingKC\_1( $\epsilon_1$ , $\epsilon_2$ , $\epsilon$ , $\delta$ ) **Input:** $\epsilon = \epsilon_1 + \epsilon_2$ $_{1} \mathcal{R} \leftarrow \emptyset; \mathcal{R}' \leftarrow \emptyset; i \leftarrow 0$ 2 repeat $i \leftarrow i + 1; \ q \leftarrow 2^{-i}; \ \delta \leftarrow \frac{6\delta}{\pi^2 i^2}; \ \mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}'$ 3 $\ell \leftarrow \max\left\{\frac{2(1+\epsilon_1)}{3\epsilon_1^2q^2}\ln\frac{2n(n-1)}{\delta}, \frac{2(1-\epsilon_1)}{3\epsilon_2^2q^2}\ln\frac{2n(n-1)}{\delta}\right\};$ if $|\mathcal{R}| < \ell$ then 5 Add more random samples into $\mathcal{R}$ until $|\mathcal{R}| \ge \ell$ $(C^*, B^*) \leftarrow \text{SearchKC}_1(G, k, \mathcal{R})$ Generate another set $\mathcal{R}'$ of random samples such that **foreach** $(u, v) \in B^* \land u \neq v$ **do** $\begin{vmatrix} a \leftarrow \ln \frac{2(n-k)}{\delta}; \ \theta \leftarrow |\mathcal{R}'| \\ z(u,v) \leftarrow \left(\sqrt{\widehat{\Pr}[\mathcal{R}', u \sim v]} - \sqrt{\frac{a}{6\theta}}\right)^2 - \frac{a}{6\theta} \end{vmatrix}$ 10 11 $(u^*, v^*) \leftarrow \arg\min_{(u, v) \in B^* \land u \neq v} z(u, v)$ 12 13 until $z(u,v) \geq q$ ;

```
Algorithm 10: SearchKC+(\epsilon_1, \epsilon_2, \epsilon_3, \epsilon, \delta)
      Input: (1 - \epsilon)(1 + \epsilon_3) = (1 - \epsilon_1)(1 - \epsilon_2)(1 - \epsilon_3)
 [q_1,q_2] \leftarrow [0,1]; C^* \leftarrow \emptyset; \mathcal{R} \leftarrow \emptyset; i \leftarrow 0
 2 repeat
              i \leftarrow i + 1; \ q \leftarrow \frac{q_1 + q_2}{2}; \ \delta \leftarrow \frac{6\delta}{\pi^2 i^2}\ell \leftarrow \lceil \frac{2(1 + \epsilon_3)}{3\epsilon_3^2(1 - \epsilon)q} \ln \frac{n^2 + n - 2k}{2\delta} \rceil;
 5
                 Add more random samples into \mathcal{R} until |\mathcal{R}| = \ell
                q' \leftarrow (1 - \epsilon_3)q; (u, W) \leftarrow \text{GetFirstNode}\_1(q', \mathcal{R})
                C \leftarrow \{u\}; \ \eta \leftarrow UB(u);
                while \widehat{L}_{\mathcal{R}}(q',C) < nq' - \epsilon_1 q' \wedge |C| < \lceil \ln \frac{n}{\epsilon_1} \rceil k do
 9
                     (u, W) \leftarrow \text{GetNextNode}(W, \widehat{L}_{\mathcal{R}}(q', \cdot), C);
10
                   C \leftarrow C \cup \{u\}
11
               \begin{array}{l} \textbf{if} \ |C| \leq \lceil \ln \frac{n}{\epsilon_1} \rceil k \wedge \widehat{L}_{\mathcal{R}}(q',C) \geq nq' - \epsilon_1 q' \ \ \textbf{then} \\ \Big| \ \ C^* \leftarrow C; \ q_1 \leftarrow q; \ B^* \leftarrow D(\mathcal{R},C^*) \end{array}
12
13
14
                  q_2 \leftarrow q
16 until q_1 \ge (1 - \epsilon_2)q_2;
17 return (C^*, B^*)
```

As  $\mathrm{OPT}_k^c$  is unknown, we need to find a lower bound of  $\mathrm{OPT}_k^c$  to determine the cardinality of  $\mathcal{R}$ . Recall that we have used a trivial lower bound k/n for  $\mathrm{OPT}_k^m$  in the k-median problem. However, it is hard to find an ideal lower bound for  $\mathrm{OPT}_k^c$ . A trivial lower bound for  $\mathrm{OPT}_k^c$  is the production of the existence probabilities of all the edges in E, but this lower bound could be too small and hence results in a large number of generated random samples. In the sequel we will provides more efficient algorithm for k-Center.

#### **Algorithm 11:** GetFirstNode\_1(q, $\mathcal{R}$ )

- 1 Compute the node list W and the values of  $\forall v \in V : UB(v)$  using Lines 1-3 of GetFirstNode( $\mathcal{R}$ )
  2  $(u, W) \leftarrow \text{GetNextNode}(W, \widehat{L}_{\mathcal{R}}(q, \cdot), \emptyset)$
- 3 return (u, W)

We first study whether we can apply the OPIM framework [6] to accelerate our algorithm. For any  $B \in \mathcal{S}_k^G$ , any  $C \subseteq V$  and any  $v \in V$ , we define

$$\widehat{KC}(\mathcal{R}, B) = \min_{(u, v) \in B} \widehat{\Pr}[\mathcal{R}, u \sim v]$$

Let  $(u^o, v^o)$  denote a cluster link in  $B^o$  such that  $\widehat{\Pr}[\mathcal{R}, u^o \sim v^o] = \widehat{KC}(\mathcal{R}, B^o)$ . The OPIM framework requires that we can find an upper bound of  $\widehat{\Pr}[\mathcal{R}, u^o \sim v^o]$  using  $B^*$ , under the purpose that we can get an upper bound of  $\widehat{\operatorname{OPT}}_k^c$ . This idea, however, cannot be applied to the k-center problem. To explain, note that  $\widehat{\Pr}[\mathcal{R}, u^o \sim v^o]$  could be larger than  $\widehat{KC}(\mathcal{R}, B^o)$ , while we can only guarantee that  $\widehat{KC}(\mathcal{R}, B^o)$  is no more than  $\widehat{\Pr}[\mathcal{R}, u^o \sim v^o]$  for all  $(u, v) \in B^*$ . Therefore, it is possible that  $\widehat{\Pr}[\mathcal{R}, u^o \sim v^o]$  is larger than the estimated probability of any cluster link in  $B^*$ . Therefore, the OPIM framework cannot be applied to the k-center problem.

Based on the above observation, we propose a method to judge whether  $q \geq \mathsf{OPT}_k^c$  using a relatively small number of random samples. For any  $q \in (0,1]$  and any  $C \subseteq V$ , we define

$$\widehat{L}_{\mathcal{R}}(q,C) = \sum_{v \in V} \min\{q, \widehat{f}_v(\mathcal{R}, C)\}$$
(12)

and we prove the following lemma:

LEMMA 4.7. Let  $q, \delta$  be any numbers in (0,1) and  $\mathcal{R}$  be any set of weakly dependant random samples of G. If  $\mathrm{OPT}_k^c > q$  and  $|\mathcal{R}| \geq \frac{2(1+\epsilon_3)}{3\epsilon_4^2q} \ln \frac{n-k}{\delta}$ , then we must have

$$\Pr[\widehat{L}_{\mathcal{R}}(q,C) \geq (n - \epsilon_1)(1 - \epsilon_3)q] \geq 1 - \delta,$$
where  $C = \text{Greedy}(\lceil \ln \frac{n}{\epsilon_1} \rceil k, \widehat{L}_{\mathcal{R}}(q,\cdot)).$ 

Note that we must have  $\widehat{KC}(\mathcal{R},B^o) \geq (1-\varepsilon_3)\mathrm{OPT}_k^c$  with probability of at least  $1-\delta$  when  $|\mathcal{R}| \geq \frac{2(1+\varepsilon_3)}{3\varepsilon_3^2q} \ln \frac{n-k}{\delta}$ . So the proof of Lemma 4.7 is similar to that of Lemma 4.3.

With Lemma 4.7, we propose a binary searching process similar to that in Algorithm 11 to find a bi-criteria approximation solution to the k-center problem, as shown by the SearchKC+ algorithm.

Similar to the SearchKC algorithm, the SearchKC+ algorithm also maintains a searching interval  $[q_1,q_2]$  and halves this interval in each iteration. The main difference between SearchKC and SearchKC+ is that we have replaced the function  $L(q,\cdot)$  by  $\widehat{L}_{\mathcal{R}}(q,\cdot)$  and used Lemma 4.7 to guide the direction of the binary searching process. More specifically, in each iteration i, we set  $q=\frac{q_1+q_2}{2}$ , and generate a set  $\mathcal{R}$  of  $\frac{2(1+\epsilon_3)}{3\epsilon_3^2q}\ln\frac{n-k}{\delta}$  random samples. Then we greedily select at most 1 nodes into C. If  $\widehat{L}_{\mathcal{R}}(q,C)\geq (n-\epsilon_1)(1-\epsilon_3)q$ , then we can judge that  $\mathrm{OPT}_k^c>q$ , and the probability that such a judgement is wrong is no more than  $\frac{6\delta}{\pi^2 i^2}$  due to Lemma A.1. By the union bound, the probability that we have searched the wrong direction is no more than  $\sum_{i=1}^\infty \frac{6\delta}{\pi^2 i^2} = \delta$ . If we never search

the wrong direction, then we use similar reasoning with that in Sec. 4.1.2 to know that we have got a good approximation solution. More specifically, we can prove:

Theorem 4.8. For any  $\epsilon, \epsilon_1, \epsilon_2, \epsilon_3 \in (0,1)$  satisfying  $1-\epsilon = (1-\epsilon_1)(1-\epsilon_2)(1-\epsilon_3)$ , the SearchKC+ algorithm can find a solution that achieves  $(1-\epsilon)\mathsf{OPT}_k^c$  with probability of at least  $1-\delta$ , using at most  $\left[\ln\frac{n}{\epsilon_1}\right]k$  centers. This algorithm has no more than  $\left[\log_2\frac{1}{\epsilon_2\cdot\mathsf{OPT}_k^c}\right]$  iterations.

Similar to the SearchKM+ algorithm, SearchKC+ also leverages the CELF framework to reduce the number of evaluating  $\widehat{L}_{\mathcal{R}}(q,\cdot)$ . However, it uses a different procedure (i.e., the GetFirstNode\_1 algorithm) to address the "cold start" problem.

Finally, we ask the question whether the SearchKC+ could generate "too many" random samples, compared with the upper bound proposed in Theorem 4.6. To answer this question, we prove the following theorem:

Theorem 4.9. The expected number of random samples generated in the SearchKC+ algorithm is at most  $O\left(\frac{1}{\epsilon^2 \text{OPT}_k^c} \left(\ln \frac{n}{\delta} + \ln \ln \frac{1}{\epsilon \text{OPT}_k^c}\right)\right)$ 

Note that the bound shown in Theorem 4.9 is very close to that shown in Theorem 4.6, and it has only introduced an additional  $\ln \ln \frac{1}{\epsilon \text{OPT}_k^c}$  factor. This demonstrates that the SearchKC+ algorithm would not generate a lot of unnecessary random samples.

#### 5 CONCLUSION

We have studied the k-median and k-center problems in uncertain graphs. We have analyzed the complexity of these problems and proposed efficient algorithms with improved approximation ratios compared with the prior art.

### A MISSING LEMMAS

LEMMA A.1. Given any set  $A \subseteq V \times V$ , any set  $\mathcal{R}$  of weakly-dependent random samples of G and any positive number  $\varepsilon$ , we have

$$\Pr\left[\widehat{\Upsilon}(A) - \Upsilon(A) \ge \varepsilon\right] \le \exp\left\{-\frac{3\varepsilon^2 |\mathcal{R}|}{2(\varepsilon + \Upsilon(A))}\right\}$$

where  $\Upsilon(A) = \sum_{(u,v) \in A} \Pr[u \sim v]/|A|$  and  $\widehat{\Upsilon}(A) = \sum_{(u,v) \in A} \widehat{\Pr}[\mathcal{R}, u \sim v]/|A|$ 

### **B** MISSING PROOFS

### **B.1** Proof of Theorem 3.2

PROOF. For any  $X \subseteq Y \subseteq V$  and any  $x \in V \setminus Y$ , we have

$$\begin{split} f_{\upsilon}(X) &= \max\{\Pr[u \sim \upsilon] \mid u \in X\} \\ &\leq &\max\{\Pr[u \sim \upsilon] \mid u \in Y\} = f_{\upsilon}(Y) \end{split} \tag{13}$$

So  $f_v(\cdot)$  is monotone. Note that

$$f_{v}(X \cup \{x\}) = \max\{\Pr[x \sim v], f_{v}(X)\}$$

So we have:

1) if  $\Pr[x \sim v] \ge f_v(Y) = \max\{\Pr[u \sim v] \mid u \in Y\}$ , then we also have  $\Pr[x \sim v] \ge f_v(X)$ , so

$$f_{v}(X \cup \{x\}) - f_{v}(X)$$

$$= \Pr[x \sim v] - f_{v}(X)$$

$$\geq \Pr[x \sim v] - f_{v}(Y)$$

$$= f_{v}(Y \cup \{x\}) - f_{v}(Y)$$
(14)

2) if  $\Pr[x \sim v] < f_v(Y) = \max\{\Pr[u \sim v] \mid u \in Y\}$ , then we have  $f_v(Y \cup \{x\}) = f_v(Y)$ , so we also have

$$f_v(X \cup \{x\}) - f_v(X) \ge 0 = f_v(Y \cup \{x\}) - f_v(Y)$$
 (15)

### **B.2** Proof of Theorem 3.5

PROOF. Let  $\mathcal{S}_{\mathcal{R}}=\{D(\mathcal{R},C)||C|=k\land C\subseteq V\}$ . Then we must have  $|\mathcal{S}_{\mathcal{R}}|\leq \binom{n}{k}$ . Let  $\widehat{A}^o$  denote the signature in  $\mathcal{S}_{\mathcal{R}}$  such that  $\widehat{KM}(\mathcal{R},\widehat{A}^o)$  is maximized. Let  $A^o$  denote the signature of an optimal clustering of the k-median problem. Note that  $A^o$  is not necessarily in  $\mathcal{S}_{\mathcal{R}}$ , but there must exist certain  $A'\in\mathcal{S}_{\mathcal{R}}$  such that  $\widehat{KM}(\mathcal{R},A')\geq\widehat{KM}(\mathcal{R},A^o)$ . Therefore, we must have:

$$\widehat{KM}(\mathcal{R}, A^*) \ge (1 - 1/e)\widehat{KM}(\mathcal{R}, \widehat{A}^o) \ge (1 - 1/e)\widehat{KM}(\mathcal{R}, A^o)$$

With the above equation, we can ensure that  $A^*$  has a  $1-1/e-\epsilon$  approximation ratio by adopting the following sampling method: we generate sufficient random samples in  $\mathcal R$  such that  $\widehat{KM}(\mathcal R,A)$  is an estimation of KM(A) within an absolute error of  $\beta=\frac{e}{2e-1}\cdot \epsilon \mathrm{OPT}_{km}$  for any  $A\in\mathcal S_{\mathcal R}\cup\{A^o\}$ . More specifically, when  $|\mathcal R|\geq \frac{2(2e-1)(e\epsilon+2e-1)}{3e^2\epsilon^2\mathrm{OPT}_{km}}\ln\frac{\binom{n}{k}+1}{\delta}$ , we must have

$$\Pr\left[\exists A \in \mathcal{S}_{\mathcal{R}} : KM(A) < \widehat{KM}(\mathcal{R}, A) - \beta\right]$$

$$\leq \sum_{A \in \mathcal{A}_{\mathcal{R}}} \Pr[KM(A) < \widehat{KM}(\mathcal{R}, A) - \beta]$$

$$\leq \frac{\binom{n}{k} \delta}{\binom{n}{k} + 1} \tag{16}$$

and

$$\Pr\left[\widehat{KM}(\mathcal{R}, A^o) < KM(A^o) - \beta\right] \le \frac{\delta}{\binom{n}{k} + 1} \tag{17}$$

Moreover, as  $A^* \in \mathcal{A}_{\mathcal{R}}$ , we can use the union bound to get that, with probability of at least  $1 - \delta$ , we have:

$$\begin{split} KM(A^*) & \geq \widehat{KM}(\mathcal{R}, A^*) - \beta \\ & \geq (1 - 1/e)\widehat{KM}(\mathcal{R}, A^o) - \beta \\ & \geq (1 - 1/e)\left(KM(A^o) - \beta\right) - \beta \\ & = (1 - 1/e - \epsilon) \mathrm{OPT}_k^m \end{split}$$

Hence, the lemma follows.

### **B.3** Proof of Theorem 4.1

PROOF. Again, we prove the theorem by a reduction from the NP-hard dominating set problem. Given an instance G=(V,E) of the dominating set problem where |V|=n, we can construct an uncertain graph by setting  $p(e)=q=\frac{\alpha}{2(1+\alpha)n}$  for all  $e\in E$ . Suppose that there exists a polynomial-time algorithm that can find  $B\in \mathcal{S}_L^G$ 

such that  $KC(B) \ge \alpha KC(\tilde{B})$ , where  $\tilde{B}$  is an optimal solution to the k-center problem in the uncertain G constructed above. We will prove that: there exists a dominating set S in G with |S| = k if and only if  $KC(B) \ge \alpha q$ . Indeed, if such a dominating set S exists, then we must have  $KC(\tilde{B}) \ge q$  and hence  $KC(B) \ge \alpha q$ . Conversely, if there does not exist such a dominating set S, then there must exist a cluster link (u,v) in B such that  $u \ne v$  and u is not adjacent to v. Thus, we can use similar reasoning as that in Theorem 3.1 to prove that

$$KC(B) \le \Pr[u \sim v] \le \frac{nq^2}{1 - nq} < \alpha q$$
 (18)

The above reasoning implies that, if there exists a polynomial-time algorithm to the k-center problem with any approximation ratio  $\alpha$ , then the dominating set can also be optimally solved in polynomial time. Hence, the theorem follows.

#### **B.4** Proof of Theorem 4.2

PROOF. Given any graph G = (V, E) with |V| = n, we can construct an uncertain graph by setting  $p(e) = q = \frac{1}{3n}$ . Under this setting, we can use similar reasoning with that in Theorem 4.1 to prove that: for any clustering B of G, the set of center nodes in B is a dominating set of G if and only if  $KC(B) \ge q$ .

Suppose that the cardinality of the minimum dominating set in G is  $k^*$  ( $k^*$  is unknown), and suppose by contradiction that there exists a bi-criteria approximation algorithm  $\mathcal A$  that achieves the properties described by the theorem. Then we can find a dominating set of G as follows. We run  $\mathcal A$  for all  $k \in [n]$ . Let  $C_k$  denote the set of center nodes returned by  $\mathcal A$  for any  $k \in [n]$ . We then return the set  $C \in \{C_1, \cdots, C_n\}$  such that C is a dominating set of G and |C| is minimized.

Note that  $\mathrm{OPT}_{k^*}^c \geq q$  in such a case. So  $C_{k^*}$  must be a dominating set of G according to the above reasoning. Moreover, we have  $|C_{k^*}| < k^* \ln n$  and hence  $|C| < k^* \ln n$ . This implies that we have built a polynomial-time dominating set algorithm with an approximation ratio less than  $\ln n$ . However, it is proved in [2] that such a dominating set algorithm should not exist unless P=NP. Therefore, we got a contradiction, which proves the theorem.

#### **B.5** Proof of Lemma 4.3

PROOF. Consider the following optimization problem:

Minimize 
$$|C|$$
 [KCT\_COVER]  
s.t.  $L(q,C) \ge nq; C \subseteq V$  (19)

Note that  $L(q, \cdot)$  is a monotone and submodular function and L(q, V) = nq. So this problem is actually a "submodular set cover" problem. Suppose that an optimal solution to [KCT\_COVER] is  $C^o_{cover}$ . We can use a greedy algorithm [3] to find C' such that

$$|C'| \le \left[\ln \frac{n}{\epsilon_1}\right] |C_{cover}^o|; L(q, C') \ge nq - \epsilon_1 q$$
 (20)

Now suppose by contradiction that  $OPT_{k}^{c} > q$ , then we must have

$$\min_{v \in V} f_v(C_{kc}^o) = \mathrm{OPT}_k^c > q, \tag{21}$$

where  $C_{kc}^{o}$  denote the set of centering nodes in  $A^{o}$ . So we have

$$L(q, C_{kc}^{o}) = \sum_{v \in V} \min\{q, f_{v}(C_{kc}^{o})\} = qn$$
 (22)

This implies that  $C^o_{kc}$  is a feasible solution to [KCT\_COVER]. Therefore, we must have

$$|C'| \leq \left\lceil \ln \frac{n}{\epsilon_1} \right\rceil |C^o_{cover}| \leq \left\lceil \ln \frac{n}{\epsilon_1} \right\rceil |C^o_{kc}| \leq \left\lceil \ln \frac{n}{\epsilon_1} \right\rceil k$$

and hence  $C' \subseteq C$ . This implies  $L(q,C) \ge nq - \epsilon_1 q$ , a contradiction. Hence, the lemma follows.  $\Box$ 

#### **B.6** Proof of Theorem 4.5

PROOF. Let  $Z=\{(u,v)|u,v\in V\land u\neq v\}$ . Define the sets  $Q_{\mathcal{R}},Q_{\mathcal{R}}'$  and the events  $\mathcal{E}_1,\mathcal{E}_2$  as

$$\begin{split} &Q_{\mathcal{R}} = \{(u,v) | (u,v) \in Z \land \Pr[u \sim v] \geq (\mathrm{OPT}_k^c)^2 \} \\ &\mathcal{E}_1 = \{ \forall (u,v) \in Q_{\mathcal{R}} : \widehat{\Pr}[\mathcal{R}, u \sim v] \geq (1 - \epsilon_1) (\mathrm{OPT}_k^c)^2 \} \\ &Q_{\mathcal{R}}' = \{(u,v) | (u,v) \in Z \land \Pr[u \sim v] < (1 - \epsilon) (\mathrm{OPT}_k^c)^2 \} \\ &\mathcal{E}_2 = \{ \forall (u,v) \in Q_{\mathcal{R}}' : \widehat{\Pr}[\mathcal{R}, u \sim v] < (1 - \epsilon_1) (\mathrm{OPT}_k^c)^2 \} \end{split}$$

In the sequel, we will prove that: when  $\mathcal{E}_1$  and  $\mathcal{E}_2$  both happen, then we must have  $KC(B^*) \geq (1 - \epsilon)(\mathsf{OPT}^c_L)^2$ .

Suppose that the nodes sequentially selected by SearchKC\_1 are  $v_1, v_2, \cdots, v_k$ . Let  $C_i^* = \{v_1, \cdots, v_i\}$ . Let  $v_{k+1}$  be a node in  $V \setminus C_k^*$  such that  $\widehat{d}(\mathcal{R}, v_{k+1}, C_k^*)$  is maximized. Let  $l_i = \widehat{d}(\mathcal{R}, v_{i+1}, C_i^*)$ . We first prove that  $l_k \leq -2 \ln KC(B^o) - \ln(1-\epsilon)$ . It can be seen that  $l_1, l_2, \cdots, l_k$  are non-increasing. Suppose by contradiction that  $l_k > -2 \ln KC(B^o) - \ln(1-\epsilon)$ , then there must exist two nodes  $v_i$  and  $v_j$  in one cluster of  $B^o$  such that  $\widehat{d}(\mathcal{R}, v_i, v_j) \geq -2 \ln KC(B^o) - \ln(1-\epsilon)$ , which implies that

$$\widehat{\Pr}[\mathcal{R}, v_i \sim v_j] < (1 - \epsilon)(\text{OPT}_k^c)^2$$
(23)

However, as  $v_i$  and  $v_j$  is in the same cluster of  $B^o$ , we must have  $\Pr[v_i \sim v_j] \geq (\mathsf{OPT}_k^c)^2$ , which contradicts the assumption that the event  $\mathcal{E}_1$  happens.

As  $l_k \le -2 \ln KC(B^o) - \ln(1 - \epsilon)$ , we must have

$$\forall (u, v) \in B^* : \widehat{\Pr}[\mathcal{R}, u \sim v] \ge (1 - \epsilon_1)(\operatorname{OPT}_k^c)^2$$
 (24)

As  $\mathcal{E}_2$  happens, we must have

$$KC(B^*) \ge (1 - \epsilon)(OPT_k^c)^2$$
 (25)

Now the problem left is to prove that

$$\Pr[\neg \mathcal{E}_1 \lor \neg \mathcal{E}_2] \le \delta \tag{26}$$

This can be proved by using the union bound and Lemma A.1. Hence the theorem follows.  $\Box$ 

#### **B.7** Proof of Theorem 4.6

PROOF. Let  $(u^*, v^*)$  be the cluster-link in  $B^*$  such that  $\Pr[u \sim v]$  is minimized. Let  $(\widehat{u}, \widehat{v})$  be the cluster-link in  $B^*$  such that  $\widehat{\Pr}[\mathcal{R}, \widehat{u} \sim \widehat{v}]$  is minimized. Let  $B^o$  be the signature of an optimal k-clustering in G such that  $KC(B^o)$  is maximized. Let  $(\widehat{u'}, \widehat{v'})$  be the cluster-link in  $B^o$  such that  $\widehat{\Pr}[\mathcal{R}, \widehat{u'} \sim \widehat{v'}]$  is minimized. Let  $(u^o, v^o)$  be the cluster-link in  $B^o$  such that  $\Pr[u^o \sim v^o]$  is minimized. Let  $\mathcal{E}_1$  denote the following event:

$$\mathcal{E}_1 = \{\widehat{\Pr}[\mathcal{R}, \widehat{u}' \sim \widehat{v}'] \ge (1 - \epsilon_3) \Pr[\widehat{u}' \sim \widehat{v}']\}$$
 (27)

If  $\mathcal{E}_1$  hold, then we must have

$$\widehat{\Pr}[\mathcal{R}, u^* \sim v^*] \ge \widehat{\Pr}[\mathcal{R}, \widehat{u} \sim \widehat{v}]$$

$$\ge (1 - \epsilon_1)(1 - \epsilon_2)\widehat{KC}(\mathcal{R}, \widehat{B}^o)$$

$$\ge (1 - \epsilon_1)(1 - \epsilon_2)\widehat{KC}(\mathcal{R}, B^o)$$

$$= (1 - \epsilon_1)(1 - \epsilon_2)\widehat{\Pr}[\mathcal{R}, \widehat{u}' \sim \widehat{v}']$$

$$\ge (1 - \epsilon_1)(1 - \epsilon_2)(1 - \epsilon_3)\Pr[\widehat{u}' \sim \widehat{v}']$$

$$\ge (1 - \epsilon_1)(1 - \epsilon_2)(1 - \epsilon_3)\Pr[u^o \sim v^o]$$

$$= (1 - \epsilon)(1 + \epsilon_3)\operatorname{OPT}_b^c$$

(28)

where (28) is due to the performance guarantee of SearchKC. Note that there are at most n-k possible choices for  $(\widehat{u}',\widehat{v}')$ .

Therefore, when  $|\mathcal{R}| \geq \frac{2(1+\epsilon_3)}{3\epsilon_3^2 \text{OPT}_c^c} \ln \frac{n-k}{\delta_1}$ , we can get

$$\Pr[\neg \mathcal{E}_1] \le (n - k) \exp\left(-\frac{3\epsilon_3^2 \Pr[\widehat{u}' \sim \widehat{v}']}{2(1 + \epsilon_3)} |\mathcal{R}|\right)$$

$$\le \frac{2(n - k)}{n^2 + n - 2k} \delta, \tag{29}$$

where we have used the the union bound and Lemma A.1.

Let  $\mathcal{K} = \{(u,v)|u,v \in V \land \Pr[u \sim v] < (1-\epsilon) \mathrm{OPT}_k^c\}$ . So we have  $|\mathcal{K}| \leq \binom{n}{2}$ . When  $|\mathcal{R}| \geq \frac{2(1+\epsilon_3)}{3\epsilon_3^2(1-\epsilon)\mathrm{OPT}_k^c} \ln \frac{\binom{n}{2}}{\delta_2}$ , we can use the the union bound and Lemma A.1 to get

$$\begin{aligned} & \Pr[(u^*, v^*) \in \mathcal{K} \wedge \mathcal{E}_1] \\ & \leq & \Pr[\exists (u, v) \in \mathcal{K} : \widehat{\Pr}[\mathcal{R}, u \sim v] \geq (1 - \epsilon)(1 + \epsilon_3) \mathrm{OPT}_k^c] \\ & \leq & \sum_{(u, v) \in \mathcal{K}} \Pr[\widehat{\Pr}[\mathcal{R}, u \sim v] \geq \Pr[u \sim v] + \epsilon_3 (1 - \epsilon) \mathrm{OPT}_k^c] \\ & \leq & \binom{n}{2} \exp\left(-\frac{3\epsilon_3^2 (1 - \epsilon) \mathrm{OPT}_k^c}{2(1 + \epsilon_3)} |\mathcal{R}|\right) \leq \frac{n(n - 1)}{n^2 + n - 2k} \delta \end{aligned}$$

Combining the above results, we get

$$\Pr[\Pr[u^* \sim v^*] < (1 - \epsilon) \text{OPT}_k^c] \le \Pr[(u^*, v^*) \in \mathcal{K}]$$

$$\le \Pr[(u^*, v^*) \in \mathcal{K} \land \mathcal{E}_1] + \Pr[\neg \mathcal{E}_1]$$

$$\le \delta_1 + \delta_2 \le \delta$$
(30)

Hence, the theorem follows.

### B.8 Proof of Lemma 4.7

PROOF. Let  $\widehat{B}^o$  denote the k-clustering in G such that  $\widehat{KC}(\mathcal{R},\widehat{B}^o)$  is maximized. If  $\widehat{KC}(\mathcal{R},\widehat{B}^o) > (1-\epsilon_3)q$ , then we can get  $\widehat{L}_{\mathcal{R}}(q,C) \geq (n-\epsilon_1)(1-\epsilon_3)q$  by similar reasoning with that in Lemma 4.3. Therefore, we get

$$\begin{split} &\Pr[\widehat{L}_{\mathcal{R}}(q,C) < (n-\epsilon_1)(1-\epsilon_3)q] \\ \leq &\Pr[\widehat{KC}(\mathcal{R},\widehat{B}^o) \leq (1-\epsilon_3)q] \\ \leq &\Pr[\widehat{KC}(\mathcal{R},B^o) \leq (1-\epsilon_3)q] \\ \leq &\Pr[\exists (u,v) \in B^o : \widehat{\Pr}[\mathcal{R},u \sim v] \leq (1-\epsilon_3)q] \\ \leq &\sum_{(u,v) \in B^o \land u \neq v} \Pr[\widehat{\Pr}[\mathcal{R},u \sim v] \leq (1-\epsilon_3)q] \\ \leq &(n-k) \exp\left(-\frac{3\epsilon_3^2 \mathrm{OPT}_k^c}{2(1+\epsilon_3)}|\mathcal{R}|\right) \leq \delta \end{split}$$

Hence, the lemma follows.

#### **B.9** Proof of Theorem 4.9

PROOF. Suppose that  $q_{min}$  is the smallest q that is tested by SearchKC+. Then we must have  $q_{min} \in \{2^{-j} | j \ge 1\}$ . Suppose that  $q_{min} = 2^{-i_{min}}$ . Then the SearchKC+ algorithm takes  $i_{min}$  iterations to reduce q from  $\frac{1}{2}$  to  $q_{min}$ , and then takes at most another  $i_{min} + \lceil \log_2 \frac{1}{\epsilon_2} \rceil$  iterations to end. Note that  $i_{min}$  is a random number. When  $i_{min} = i$ , the total number of generated random samples is no more than

$$\ell(i) = \left\lceil \frac{2^{i+1}(1+\epsilon_3)}{3\epsilon_3^2(1-\epsilon)} \ln \frac{\pi^2(n^2+n-2k)(2i+\lceil \log_2 \frac{1}{\epsilon_2} \rceil)^2}{12\delta} \right\rceil$$

It can be verified that  $\forall i \geq 1: \ell_{i+1} \leq 3\ell_i$ . Suppose that  $i_0 \geq 1$  is the smallest number such that  $2^{-i_0} \leq \mathsf{OPT}^c_k$ . So we must have  $2^{-i_0+1} \geq \mathsf{OPT}^c_k$ . For any  $i > i_0$ , we have

$$\Pr[i_{min} = i] \le \Pr[|C_{i-1}^*| > \lceil \ln(n/\epsilon_1) \rceil k]$$

$$\le \Pr[\widehat{KC}(\mathcal{R}_{i-1}, B^o) < (1 - \epsilon_3) \text{OPT}_k^c]$$

$$\le \exp\left(-\frac{3\epsilon_3^2 \text{OPT}_k^c}{2(1 + \epsilon_3)} |\mathcal{R}_{i-1}|\right) \le \delta^{\text{OPT}_k^c \cdot 2^{i-1}}$$
(31)

where  $\mathcal{R}_{i-1}$  and  $C_{i-1}^*$  denote the set of generated random samples and the set of centering nodes found by SearchKC+ when  $q=2^{-i+1}$ , respectively. Let  $a=\mathrm{OPT}_k^c\cdot 2^{i_0}$ . When  $\delta\leq 1/3$ , the total expected number of generated random samples is no more than

$$\begin{split} &\ell(i_0) + \sum_{i > i_0} \ell(i) \Pr[i_{min} = i] \\ &\leq & \ell(i_0) + \sum_{j=0}^{\infty} 3^{j+1} \ell(i_0) \delta^{2^j a} \leq 2\ell(i_0) + 3\ell(i_0) \sum_{j=1}^{\infty} 3^{j-2^j} \\ &\leq & 2\ell(i_0) + 3\ell(i_0) \sum_{j=1}^{\infty} 3^{-j} \leq 7\ell(i_0)/2 \end{split}$$

Note that  $i_0 = \lceil \log_2 \frac{1}{OPT_L^c} \rceil$ , so we have

$$\ell(i_0) = O\left(\frac{1}{\epsilon^2 \text{OPT}_k^c} \left( \ln \frac{n}{\delta} + \ln \ln \frac{1}{\epsilon \text{OPT}_k^c} \right) \right)$$
(32)

П

Hence the theorem follows

#### REFERENCES

- Matteo Ceccarello, Carlo Fantozzi, Andrea Pietracaprina, Geppino Pucci, and Fabio Vandin. 2017. Clustering Uncertain Graphs. PVLDB 11, 4 (2017), 472–484.
- [2] Uriel Feige. 1998. A threshold of ln n for approximating set cover. J. ACM 45, 4 (1998), 634–652.
- [3] Amit Goyal, Francesco Bonchi, Laks V. S. Lakshmanan, and Suresh Venkatasubramanian. 2013. On minimizing budget and time in influence propagation over social networks. Social Network Analysis and Mining 3, 2 (2013), 179–192.
- [4] Andreas Krause, H Brendan McMahan, Carlos Guestrin, and Anupam Gupta. 2008. Robust submodular observation selection. Journal of Machine Learning Research 9, Dec (2008), 2761–2801.
- [5] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne Van-Briesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In KDD. 420–429.
- [6] Jing Tang, Xueyan Tang, Xiaokui Xiao, and Junsong Yuan. 2018. Online Processing Algorithms for Influence Maximization. In SIGMOD. 991–1005.
- [7] Vijay V. Vazirani. 2001. Approximation Algorithms. Springer-Verlag, Berlin.
- [8] David P Williamson and David B Shmoys. 2011. The design of approximation algorithms. Cambridge university press.