Efficient and Effective Algorithms for Clustering Uncertain Graphs

Kai Han[†], Fei Gui[†], Xiaokui Xiao[‡], Jing Tang[*], Yuntian He[†], Zongmai Cao[†], He Huang[§]
[†]SCST/Suzhou Institute for Advanced Study, University of Science and Technology of China
[‡]School of Computing, National University of Singapore
[*]Department of Industrial Systems Engineering and Management, National University of Singapore
[§]School of Computer Science and Technology, Soochow University
hankai@ustc.edu.cn, guifei@mail.ustc.edu.cn, {xkxiao, isejtang}@nus.edu.sg,
ythe@mail.ustc.edu.cn, ustcczm@gmail.com, huangh@suda.edu.cn

ABSTRACT

We consider the edge uncertainty in an undirected graph and study the k-median (resp. k-center) problems, where the goal is to partition the graph nodes into k clusters such that the average (resp. minimum) connection probability between each node and its cluster's center is maximized. We analyze the hardness of these problems, and propose algorithms that provide considerably improved approximation guarantees than the existing studies do. Specifically, our algorithms offer $(1-1/e)$-approximations for the k-median problem and (OPT_k^c)-approximations for the k-center problem, where OPT_k^c is the optimal objective function value for k-center. In addition, our algorithms incorporate several non-trivial optimizations that significantly enhance their practical efficiency. Extensive experimental results demonstrate that our algorithms considerably outperform the existing methods on both computation efficiency and the quality of clustering results.

PVLDB Reference Format:
Kai Han, Fei Gui, Xiaokui Xiao, Jing Tang, Yuntian He, Zongmai Cao, and He Huang. Efficient and Effective Algorithms for Clustering Uncertain Graphs. *PVLDB*, 12(6): 667-680, 2019.
DOI: https://doi.org/10.14778/3311880.3311884

1. INTRODUCTION

Graph data are prevalent in numerous application domains, such as social, biological, and mobile networks. In these applications, entities are typically modeled as graph nodes, and the relationships among entities are modeled as graph edges. Uncertainties in graph edges are common in this context, due to a variety of reasons such as noisy measurements and inconsistent information sources [33]. Dealing with such uncertainies require efficient and effective methods for querying and mining uncertain graphs. Note that there are different types of uncertainties associated with edges, nodes or attributes. In this paper, we only consider edge uncertainty.

Clustering is a fundamental problem in graph mining. It aims to partition the graph nodes into a number of clusters, such that similar nodes are grouped together according to some similarity (or node-distance) measures. There exist a plethefora of formulations of graph clustering, among which the k-median and k-center problems are perhaps the most well studied [59]. Given a graph without uncertainy, the k-median problem aims to identify a set C of k *center nodes* in the graph, such that the average distance from each node to its closest center node is minimized. Meanwhile, the k-center problem aims to find k center nodes, such that the largest distance from any node to its closest center node is minimized [59, 60]. These two problems are generally studied in the *metric space*, i.e., the distances among different nodes satisfy the triangle inequality [12, 18, 59, 60].

Although k-median and k-center without uncertainty have been extensively studied in the literature, their counterparts for uncertain graphs have not been investigated until a recent study by Ceccarello et al. [11]. Following a large body of work on uncertain graphs [20, 30, 32, 33, 36, 40, 49], Ceccarello et al. [11] model an uncertain graph as a graph where each edge is associated with an existence probability, and they measure the distance between any two nodes as the probability that they are connected by a path. With this distance measure, they reformulate the k-median (resp. k-center) problem, such that the objective is to maximize the average (resp. largest) connection probability from any node to its closest center node.

Applications. Many application data can be modeled as uncertain graphs where the edges naturally have weights indicating connection probabilities and several applications require clustering the nodes of such graphs, which reduce to solving the k-median/k-center problems in uncertain graphs. Some applications include:

1) Sink/gateway placement in wireless sensor networks: Wireless sensor networks can be modeled as uncertain graphs where the edge between two sensor nodes is assigned a probability denoting the reliability of the wireless communication between them [21]. The multiple sink/gateway-placement problem in wireless sensor networks [16,35,52] can be formulated as an uncertain k-median/k-center problem, where the goal is to identify k sink nodes to maximize the average/minimum probability that the sensing data from all sensor nodes can be successfully transmitted to the sink nodes.

2) Detecting protein complexes in PPI networks: In Protein-Protein Interaction (PPI) networks, proteins and their interactions are represented by nodes and uncertain links, respectively. Protein complexes are assemblages of proteins that interact with each other, which play essential roles in many biological processes [54]. As indicated by [11], the uncertain k-median/k-center algorithms can be used to detect protein complexes in PPI networks.

3) Clustering social networks: A lot of related studies model the peer influence and interactions between the users in social networks

Proceedings of the VLDB Endowment, Vol. 12, No. 6
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3311880.3311884

Table 1: Comparing the approximation ratio and time complexity with the algorithms in [11]

		Approximation ratio (with $1-\delta$ probability)	Time complexity with unknown OPT_k^m and OPT_k^c
k-median	[11]	$\frac{(1-\epsilon)(\mathrm{OPT}_k^m)^2}{[(1+\gamma)H(n)]^3}$	$\mathcal{O}\left(\frac{(1+\frac{1}{\gamma})^3(\ln n)^3 kn^2}{\epsilon^2(\mathrm{OPT}_k^m)^3}\left(\ln\frac{n}{\delta}+\ln\log_{1+\gamma}\frac{n\ln n}{k}\right)\right)$
	this work	$1-1/e-\epsilon$	$\mathcal{O}\left(\frac{n^2}{\epsilon^2\mathrm{OPT}_k^m}\ln\frac{n}{\delta}+kn^2\ln n\right)$
k-center	[11]	$\frac{1-\epsilon}{1+\gamma}\mathrm{OPT}_k^c$	unknown
	this work	$(1-\epsilon)\mathrm{OPT}_k^c$	$\mathcal{O}\left(\frac{kn+m}{\epsilon^2(\mathrm{OPT}_k^c)^2}\left(\ln\frac{n}{\delta}+\ln\ln\frac{1}{\mathrm{OPT}_k^c}\right)\right)$

as uncertain links, so social networks are generally considered as uncertain graphs [6, 26, 29, 30, 46, 49, 62–66]. Our k-median/k-center algorithms could have many applications in clustering social networks. For example, a political party may want to group a social network into k clusters and find a representative people in each cluster, such that a popular political policy could be made by studying the characteristics of the k representative people.

4) Grouping the users in opportunistic networks: Opportunistic network (or delay tolerant network) [15, 28, 48] can be modeled as uncertain graphs where the link probabilities denote the encounter rates or data-delivery success rates between two mobile nodes [42, 43]. Our uncertain k-median/k-center algorithms can be used to group the users in an opportunistic network, which has many applications including information dissemination [44], wide animal tracking [31] and disease control [23]. For example, the medical staff could want to group the students in a campus opportunistic network and monitor k socially-active students to control the spread of flu.

Limitations of Prior Art. Compared with the conventional k-median and k-center problems, clustering uncertain graphs presents two unique challenges. First, the connection probabilities among the nodes in uncertain graphs do not satisfy the triangle inequality, whereas the conventional k-center and k median algorithms rely on the triangle inequality in deriving their approximation guarantees. Second, it is #P-hard in general to compute the connection probability between two different nodes [5]. In contrast, in conventional k-median and k-center, computing the distance between two given nodes usually incurs only a small cost.

To address the aforementioned challenges, Ceccarello et al. [11] propose new algorithms for k-center and k-median on uncertain graphs. The main idea of their algorithms is to (i) use Monte-Carlo simulations to estimate the connection probabilities among different nodes, and (ii) build upon the "clustering with outliers" algorithms proposed for metric k-center and k-median [12]. Their k-median algorithm achieves an approximation ratio of $\frac{(1-\epsilon)(\mathrm{OPT}_k^m)^2}{(1+\gamma)^3\ln^3 n}$ with high probability, where OPT_k^m is the optimal value of the k-median objective function, ϵ is a parameter that controls the number of Monte-Carlo samples, and γ is a parameter that is inversely proportional to the running time of the algorithm. Meanwhile, their k-center algorithm offers $\left(\frac{(1-\epsilon)\mathrm{OPT}_k^c}{1+\gamma}\right)$-approximation with high probability, with OPT_k^c being the optimal value of the k-center objective function.

The algorithms in [11], however, leave much room for improvements. First, to achieve the claimed aymptotic guarantees, the k-center algorithm in [11] requires knowing a lower bound of OPT_k^c, but there is no solution in [11] for deriving such a lower bound. Second, both the k-center and k-median algorithms in [11] incur significant computation overheads when rigorous approximation assurance is required, as we demonstrate in Section 6. Third, both algorithms' approximation guarantees are rather weak no matter the connection probability between any two nodes is known or not.

Contributions. Motivated by the deficiencies of the existing studies, we propose new approximation algorithms for k-median and k-center on uncertain graphs that provide significantly improved approximation and efficiency guarantees than existing solutions. (See Table 1 for a detailed comparison.) In addition, our algorithms incorporate several advanced optimizations that lead to superior practical efficiency. Furthermore, we provide a thorough analysis on the hardness and inapproximability of k-center and k-median on uncertain graphs.

More specifically, our major contributions include the following. First, we prove that uncertain k-median is NP-hard even when there exists an oracle that returns the connection probability for any two given nodes. We also show that the objective function of k-median is monotone and submodular, based on which we propose a greedy algorithm for k-median that leverages graph sampling to achieve $(1-1/e-\epsilon)$-approximations with high probability. Compared to the conventional greedy algorithms for submodular function maximization, our algorithm leverages several heuristic optimizations that drastically improve the empirical efficiency of the algorithm without affecting its approximation assurance.

Second, we present a new greedy algorithm for uncertain k-center that offers $((1-\epsilon)\mathrm{OPT}_k^c)$-approximations with high probability. Compared with the k-center algorithm proposed in [11], our approximation guarantee is better by a factor of $1/(1+\gamma)$. (Note that the k-center algorithm [11] has a running time that increases with $\frac{1}{\log(1+\gamma)}$, due to which γ has to be a positive number.) In addition, our algorithm does not require prior knowledge of OPT_c^k, whereas the algorithm in [11] requires that a tight lower bound of OPT_c^k is given. Furthermore, we prove that it is NP-hard to get any data-independent approximation guarantee for the uncertain k-center problem, and that any bi-criteria algorithm must use at least $\Omega(k\log n)$ centering nodes, unless P=NP.

Third, we evaluate the efficiency and effectiveness of our algorithms with extensive experiments. The experimental results demonstrate that our algorithms significantly outperform the algorithms in [11] on both the processing time and the quality of clustering results.

Due to the space constraint, the proofs of some theorems/lemmas presented in our paper can be found in the technical report [1].

2. PRELIMINARIES

2.1 Problem Definition

We model an uncertain graph G as a three-tuple $(V, E, p(\cdot))$, where V is the set of nodes and E is the set of edges, with $|V| = n$ and $|E| = m$, and $p : E \mapsto (0, 1]$ is a function such that $p(e)$ denotes the probability that e exists for any $e \in E$. Following [11], we assume that G is an *undirected graph*, and the presence of any

Table 2: Frequently used notations

Notation	Description
$G = (V, E)$	A uncertain graph with node set V and edge set E.
n, m	the numbers of nodes and edges in G, respectively
k	the number of clusters
$u \sim v$	the event that node u is connected to node v in G
$KM(A)$, $KC(A)$	the average and minimum connection probability of the cluster links in the k-clustering A, respectively
A^o, B^o	the optimal solutions to the k-median and k-center problems, respectively
C^o_{km}, C^o_{kc}	the set of center nodes in A^o and B^o, respectively
OPT^m_k	the value of $KM(A^o)$
OPT^c_k	the value of $KC(B^o)$
$f_v(C)$	the maximum connection probability between v and any node in C
$F(C)$	the value of $\sum_{v \in V} f_v(C)/n$
$X_R(u \sim v)$	If u is connected to v in the random sample R, then $X_R(u \sim v) = 1$, otherwise $X_R(u \sim v) = 0$
$\widehat{\Pr}[\mathcal{R}, u \sim v]$	the value of $\sum_{R \in \mathcal{R}} X_R(u \sim v)/\lvert\mathcal{R}\rvert$
$\widehat{f}_v(\mathcal{R}, C)$	the value of $\max\{\widehat{\Pr}[\mathcal{R}, u \sim v] \mid u \in C\}$
$\widehat{F}(\mathcal{R}, C)$	the value of $\sum_{v \in V} \widehat{f}_v(\mathcal{R}, C)$
$I(\mathcal{R})$	collection of all connected components in all $R \in \mathcal{R}$
$\Delta_{\mathcal{R}}(v\lvert C)$	the value of $\widehat{F}(\mathcal{R}, C \cup \{v\}) - \widehat{F}(\mathcal{R}, C)$

$e \in E$ is independent of all other edges. For any two nodes $u, v \in V$, we use $u \sim v$ to denote the event that u is connected to v via a path. Note that we only consider edge uncertainty in our model.

A *k-clustering* of G is represented by a tuple $\mathcal{C} = \langle C, Q_1, Q_2, \ldots, Q_k \rangle$, where $C = \{c_1, \ldots, c_k\}$ is the set of *center nodes* and $\{Q_1, Q_2, \ldots, Q_k\}$ is a partition of the nodes in V satisfying $c_i \in Q_i$ for all $i \in \{1, \ldots, k\}$. For any $i \in \{1, \ldots, k\}$ and any $v \in Q_i$, we refer to the node pair (c_i, v) as a *cluster link* of $\mathcal{C}$. We refer to the set of all cluster links in $\mathcal{C}$ as the *signature* of $\mathcal{C}$, and use $\mathcal{S}^G_k$ to denote the set of signatures of all possible k-clusterings of G. Note that any two different k-clusterings must have different signatures, and hence, we can construct a unique k-clustering from any $A \in \mathcal{S}^G_k$. For convenience, we abuse notation and refer to each $A \in \mathcal{S}^G_k$ as a k-clustering.

Given any $A \in \mathcal{S}^G_k$, we define

$$KM(A) = \frac{1}{n} \sum\nolimits_{(u,v) \in A} \Pr[u \sim v]; KC(A) = \min_{(u,v) \in A} \Pr[u \sim v].$$

With the above definitions, we formalize the k-median and k-center problems as follows:

DEFINITION 1. *The k-median problem aims to identify an optimal solution A^o to the following optimization problem:*

$$\textbf{Maximize } KM(A); \quad \textbf{s.t. } A \in \mathcal{S}^G_k$$

DEFINITION 2. *The k-center problem aims to identify an optimal solution B^o to the following optimization problem:*

$$\textbf{Maximize } KC(B); \quad \textbf{s.t. } B \in \mathcal{S}^G_k$$

Let $\mathrm{OPT}^m_k = KM(A^o)$ and $\mathrm{OPT}^c_k = KC(B^o)$, i.e., OPT^m_k and OPT^c_k are the optimal value of the objective function of k-median and k-center, respectively. In addition, we use C^o_{km} (resp. C^o_{kc}) to denote the sets of center nodes in A^o (resp. B^o). Table 2 lists the notations frequently used in the remainder of the paper.

2.2 Existing Solutions

To the best of our knowledge, only Ceccarello et al. [11] have studied the uncertain k-median/k-center problems. As computing the connection probability between any two nodes is #P-hard, they first propose algorithms with the assumption that there exists a *connectivity oracle* to compute $\Pr[u \sim v]$ for any $u, v \in V$, and then drop this assumption to propose more practical algorithms. For completeness, we quote the pseudo codes of their algorithms under the connectivity oracle in Algorithms 1-3, where ACP and MCP denote their k-median and k-center algorithms, respectively.

Algorithm 1: MIN-PARTIAL(G, k, q, α, z) /*from [11]*/

```
1  S ← ∅; V' ← V;
2  for i ← 1 to k do
3      select an arbitrary T ⊆ V' with |T| = min{α, |V'|};
4      for v ∈ T do M_v ← {u ∈ V' : Pr[u ~ v] ≥ z};
5      c_i ← arg max_{v∈T} |M_v|; S ← S ∪ {c_i};
6      V' ← V' − {u ∈ V' : Pr[u ~ c_i] ≥ q};
7  if |S| < k then
8      add k − |S| arbitrary nodes of V − S to S;
9  S ← {c_1, ··· , c_k};
10 for i ← 1 to k do C_i ← {u ∈ V' : c(u, S) = c_i};
11 return C = ({c_1, ··· , c_k}, C_1, ··· , C_k};
```

Algorithm 2: ACP(G, k, γ) /*from [11]*/

```
1  C ← MIN-PARTIAL(G, k, 1, n, 1);
2  φ_best ← (1/n) Σ_{u:u is contained in certain C_i∈C} Pr[c_i ~ u];
3  C_best ← any full k-clustering completing C;
4  q ← q/(1 + γ);
5  while q^3 ≥ φ_best do
6      C ← MIN-PARTIAL(G, k, q^3, n, q);
7      φ ← (1/n) Σ_{u:u is contained in certain C_i∈C} Pr[c_i ~ u];
8      if φ ≥ φ_best then
9          φ_best ← φ;
10         C_best ← any full k-clustering completing C;
11     else q ← q/(1 + γ); ;
12 return C_best;
```

Algorithm 3: MCP(G, k, γ) /*from [11]*/

```
1  q ← 1;
2  while true do
3      C ← MIN-PARTIAL(G, k, q, 1, q);
4      if C covers all nodes then return C ;
5      else q ← q/(1 + γ); ;
```

A key procedure in ACP/MCP is MIN-PARTIAL, shown in Algorithm 1. By inputting q, MIN-PARTIAL greedily finds a *partial k-clustering* $\mathcal{C}$ covering the maximum number of nodes in V (Lines 2-6), such that any uncovered node has a probability of less than q for being connected to the cluster centers in $\mathcal{C}$. Both ACP and MCP iteratively call MIN-PARTIAL using decreasing q (i.e., dividing q by $1 + \gamma$ at each time), as shown by Lines 5-11 in Algorithm 2 and Lines 2-5 in Algorithm 3. The iterations in MCP stop when all the nodes in V are covered, and Ceccarello et al. [11] prove that such a stopping rule must be satisfied (under the connectivity oracle) when $q \leq (\mathrm{OPT}^c_k)^2$. The stopping rule for ACP is more involved and the details can be found in [11].

Note that MIN-PARTIAL has an important input parameter α, which controls the size of its searching space (see Line 3 of Algorithm 1). A smaller α would result in smaller time complexity. Ceccarello et al. [11] have required $\alpha = n$ in ACP and $\alpha = 1$ in MCP (see Line 1 in Algorithm 2 and Line 3 in Algorithm 3), otherwise their claimed approximation ratios no longer hold.

Ceccarello et al. [11] have also extended Algorithms 2-3 to the case without a connectivity oracle, based on Monte-Carlo sam-

pling. The main idea of the extension is to generate a set of r random samples of G to estimate the connection probabilities before calling MIN-PARTIAL. More specifically, they set

$$r = \left\lceil \frac{12}{q^3\epsilon^2} \ln\left(2n^3\left(1+\left\lfloor \log_{1+\gamma}\frac{H(n)}{p_L^m}\right\rfloor\right)\right)\right\rceil \quad (1)$$

for the ACP algorithm, where $H(n)$ is the nth homonic number and p_L^m is a lower bound of $(\mathrm{OPT}_k^m/H(n))^3$, and they set

$$r = \left\lceil \frac{12}{q\epsilon^2} \ln\left(2n^3\left(1+\left\lfloor \log_{1+\gamma}\frac{1}{p_L^c}\right\rfloor\right)\right)\right\rceil \quad (2)$$

for the MCP algorithm, where p_L^c is assumed to be a known lower bound of $(\mathrm{OPT}_k^c)^2$. Their approximation ratios only hold under these parameter settings.

Based on the techniques described above, Ceccarello et al. [11] proved several theoretical bounds, as summarized below.

Results on Uncertain k-median. Ceccarello et al. [11] *conjecture* that uncertain k-median is NP-hard even when a connection oracle is available. They also prove that ACP returns a solution A to k-median satisfying

$$KM(A) \geq \left(\mathrm{OPT}_k^m/[(1+\gamma)H(n)]\right)^3 \quad (3)$$

when there is a connection oracle. This algorithm requires invoking the MIN-PARTIAL procedure at most $\lfloor \log_{1+\gamma}\frac{H(n)}{\mathrm{OPT}_k^m}\rfloor + 1$ times.

Without a connection oracle, Ceccarello et al. [11] show that ACP could utilize the Monte-Carlo sampling method described above to identify a solution A to k-median, such that

$$KM(A) \geq (1-\epsilon)\left(\frac{\mathrm{OPT}_k^m}{(1+\gamma)H(n)}\right)^3 \quad (4)$$

holds with probability of at least $1-\delta$. The algorithm invokes the MIN-PARTIAL procedure for $\lfloor \log_{1+\gamma}\frac{H(n)}{\mathrm{OPT}_k^m}\rfloor + 1$ times with high probability. Ceccarello et al. [11] have not presented the time complexity of ACP, so we provide the following theorem:

THEOREM 1. *ACP can be implemented in expected time complexity of* $\mathcal{O}\left(\frac{(1+\frac{1}{\gamma})^3(\ln n)^3kn^2}{\epsilon^2(\mathrm{OPT}_k^m)^3}\left(\ln\frac{n}{\delta}+\ln\log_{1+\gamma}\frac{n\ln n}{k}\right)\right)$ *when* OPT_k^m *is unknown.*

We roughly explain the time complexity of ACP as follows. The $\mathcal{O}(kn^2)$ factor comes from the MIN-PARTIAL procedure (as $\alpha = n$ in ACP). The other factors come from the fact that ACP terminates in $\lfloor \log_{1+\gamma}\frac{H(n)}{\mathrm{OPT}_k^m}\rfloor + 1$ iterations with high probability, and it generates r random samples according to Equation (1) to call the MIN-PARTIAL procedure in each iteration.

Note that the approximation guarantees in Equations (3) and (4) are rather weak, especially when n is large or OPT_k^m is small. In addition, the time complexity of ACP is prohibitive when n is large or OPT_k^m is small, and Ceccarello et al. [11] have not presented any efficient implementation to address the problem. Finally, when γ is small, ACP requires invoking the MIN-PARTIAL procedure a large number of times, which results in significant computation overheads.

Results on Uncertain k-center. When a connection oracle is available, Ceccarello et al. [11] prove that the k-center problem is NP-hard, and that MCP can find a solution B to k-center satisfying

$$KC(B) \geq (\mathrm{OPT}_k^c)^2/(1+\gamma). \quad (5)$$

This algorithm requires calling the MIN-PARTIAL procedure at most $\lfloor 2\log_{1+\gamma}\frac{1}{\mathrm{OPT}_k^c}\rfloor + 1$ times.

When using Monte-Carlo sampling to replace a connection oracle, Ceccarello et al. [11] show that MCP returns a solution B to k-center, such that the following ratio holds with high probability:

$$KC(B) \geq (1-\epsilon)(\mathrm{OPT}_k^c)^2/(1+\gamma) \quad (6)$$

It requires iteratively calling the MIN-PARTIAL procedure at most $\lfloor 2\log_{1+\gamma}\frac{1}{\mathrm{OPT}_k^c}\rfloor + 1$ times with high probability. Moreover, it requires a known lower bound of $(\mathrm{OPT}_k^c)^2$ (denoted by p_L^c) to achieve the approximation ratio shown in Equation (6).

The above results for uncertain k-center have two major deficiencies. First, when γ is small, the k-center algorithm in [11] incurs significant computation cost, due to the large number of invokations of the MIN-PARTIAL procedure. Second, the algorithm achieves the approximation guarantee in Equation (5) only when there is known lower bound p_L^c of $(\mathrm{OPT}_k^c)^2$. Although such a lower bound can be easily got under a connection oracle, there is no existing solution for computing a non-trivial lower bound of OPT_k^c without a connection oracle (note that a trivial lower bound k/n of OPT_k^m cannot be used as a lower bound of OPT_k^c). This severely undermines the practicability of the algorithm.

3. SOLVING THE K-MEDIAN PROBLEM

In this section, we first present algorithms for uncertain k-median with a connectivity oracle, and then propose an improved algorithm without assuming the oracle.

3.1 k-Median Algorithms with an Oracle

Prior work [11] conjectures that uncertain k-median is NP-hard even with a connectivity oracle. We prove that this conjecture holds, by a reduction from Dominating Set [59]:

THEOREM 2. *The k-median problem is NP-hard, even if* $\Pr[u \sim v]$ *can be computed in polynomial time for any* $u, v \in V$.

Although uncertain k-median with a connectivity oracle is NP-hard, we find that there exists a constant approximation using a greedy algorithm. Consider the following functions where C is any subset of V and v is any node in V:

$$f_v(C) = \max\{\Pr[u \sim v] \mid u \in C\}; \quad (7)$$

$$F(C) = \sum\nolimits_{v\in V} f_v(C)/n; \quad (8)$$

It can be verified that $F(C) \geq KM(A)$ for any $A \in \mathcal{S}_k^G$, as long as the set of center nodes in A is C. Moreover, given any $C \subseteq V$ with $|C| = k$, we can construct a k-clustering A such that $\Pr[u \sim v] = f_v(C)$ for any $(u, v) \in A$ and $u \in C$. (We refer to such a k-clustering A as "the k-clustering induced by C".) Therefore, the k-median problem is equivalent to the following optimization problem:

$$\begin{aligned} \textbf{Maximize} \quad & F(C) \\ \textbf{s.t.} \quad & |C| = k;\ C \subseteq V \end{aligned}$$

The above problem is a submodular maximization problem, as shown in Theorem 3. Intuitively, as $f_v(\cdot)$ is essentially a $\max(\cdot)$ function, its submodularity can be proved by directly using the definition of submodular functions. Moreover, the function $F(\cdot)$ is a linear combination of submodular functions $f_v(\cdot) : v \in V$, so $F(\cdot)$ is also submodular.

THEOREM 3. *The function $F(\cdot)$ is a non-negative, monotone, and submodular function on 2^V.*

PROOF. $F(\cdot)$ is non-negative by definition. For any $X \subseteq Y \subseteq V$ and any $x \in V \backslash Y$, we have

$$\begin{aligned} & f_v(X) = \max\{\Pr[u \sim v] \mid u \in X\} \\ \leq \quad & \max\{\Pr[u \sim v] \mid u \in Y\} = f_v(Y) \end{aligned} \tag{9}$$

Hence, $f_v(\cdot)$ is monotone. Note that $f_v(X \cup \{x\}) = \max\{\Pr[x \sim v], f_v(X)\}$. Therefore, we have the following:

If $\Pr[x \sim v] \geq f_v(Y)$, then we can get $\Pr[x \sim v] \geq f_v(X)$ due to $f_v(Y) \geq f_v(X)$. Hence, we have

$$\begin{aligned} & f_v(X \cup \{x\}) - f_v(X) = \Pr[x \sim v] - f_v(X) \\ \geq \quad & \Pr[x \sim v] - f_v(Y) = f_v(Y \cup \{x\}) - f_v(Y) \end{aligned} \tag{10}$$

If $\Pr[x \sim v] < f_v(Y)$, then we have $f_v(Y \cup \{x\}) = f_v(Y)$. Hence, we also have

$$f_v(X \cup \{x\}) - f_v(X) \geq 0 = f_v(Y \cup \{x\}) - f_v(Y) \tag{11}$$

According to the above reasoning and the definition of submodular functions [37], we know that $f_v(\cdot)$ is a monotone and submodular function for any $v \in V$. Finally, as $F(\cdot)$ is a linear combination of $f_v(\cdot) : v \in V$, it is also monotone and submodular. □

It is well known that monotone submodular maximization problems can be addressed by a greedy algorithm with a $1 - 1/e$ approximation ratio [37]. For completeness, we present such a greedy algorithm in Algorithm 5, and provide the following theorem:

THEOREM 4. *Let $C = \mathsf{Greedy}(G, k, F(\cdot))$, and A be the k-clustering induced by C. Then, A is a solution with a $1 - 1/e$ approximation ratio to the uncertain k-median problem.*

Note that the conventional metric k-median problem [60] is a minimization problem while our uncertain k-median problem is a maximization problem. The work in [27] has proved the NP-hardness of approximating metric k-median within a factor of $1 + 2/e$ [27], and the best-known approximation ratio for metric k-median is $2.675 + \epsilon$ [10]. Thus, our uncertain k-median is intrinsically different from the conventional metric k-median problem.

3.2 k-Median Algorithms without Oracle

In this section, we consider a more practical setting where the connectivity oracle is absent. We first present a graph sampling algorithm to address uncertain k-median, and then provide optimizations that lead to improved practical efficiency.

3.2.1 Sampling for k-Median

Although it is #P-hard to compute the connection probability between any two nodes in G, we may use random graph samples to estimate the probability with specified precision. Towards this end, we first introduce some concepts and definitions on graph sampling in what follows.

A random sample R of G is a graph generated by removing each edge e in G with probability of $1 - p(e)$. For any $u, v \in V$ and any random sample R of G, we define a function $X_R(u \sim v)$ such that $X_R(u \sim v) = 1$ if u and v is connected via a path in R, and $X_R(u \sim v) = 0$ otherwise. For any set $\mathcal{R}$ of random samples of G and any k-clustering $A \in \mathcal{S}_k^G$, we define

$$\widehat{\Pr}[\mathcal{R}, u \sim v] = \sum\nolimits_{R \in \mathcal{R}} X_R(u \sim v)/|\mathcal{R}|; \tag{12}$$

$$\widehat{KM}(\mathcal{R}, A) = \sum\nolimits_{(u,v) \in A} \widehat{\Pr}[\mathcal{R}, u \sim v]/n. \tag{13}$$

It can be verified 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 $A \in \mathcal{S}_k^G$, any $C \subseteq V$ and any $v \in V$, we define

$$\widehat{f}_v(\mathcal{R}, C) = \max\{\widehat{\Pr}[\mathcal{R}, u \sim v] \mid u \in C\}; \tag{14}$$

$$\widehat{F}(\mathcal{R}, C) = \sum\nolimits_{v \in V} \widehat{f}_v(\mathcal{R}, C). \tag{15}$$

Using similar reasoning with that in Theorem 3, we can prove that $\widehat{F}(\mathcal{R}, \cdot)$ is a monotone and submodular function. For any $C \subseteq V$ with $|C| = k$ and any set $\mathcal{R}$ of random samples of G, we use $D(\mathcal{R}, C)$ to denote the k-clustering in $\mathcal{S}_k^G$ such that (i) C is the set of center nodes in $D(\mathcal{R}, C)$, and (ii) any cluster link $(c, v) \in D(\mathcal{R}, C)$ (with $c \in C$) satisfies $\widehat{\Pr}[\mathcal{R}, c \sim v] = \widehat{f}_v(\mathcal{R}, C)$.

With the above definitions, we present the SearchKM method in Algorithm 4 for identifying an approximate solution for uncertain k-median. Given a set $\mathcal{R}$ of random samples, SearchKM first invokes the Greedy algorithm (with the function $F(\cdot)$ replaced by $\widehat{F}(\mathcal{R}, \cdot)$) to find a set C^* of center nodes. After that, it returns $A^* = D(\mathcal{R}, C^*)$ as an approximate solution. To ensure that A^* has a good approximation ratio, we present the following theorem to determine an upper-bound for the number of random samples needed by SearchKM:

THEOREM 5. *If $|\mathcal{R}| = T_{max}$ where*

$$T_{max} = \left\lceil \frac{2(7 - 7/e - 4\epsilon)(2 - 1/e)}{3\epsilon^2 \mathrm{OPT}_k^m} \ln \frac{2n^2}{\delta} \right\rceil, \tag{16}$$

then $\mathsf{SearchKM}(G, k, \mathcal{R})$ returns a $(1 - 1/e - \epsilon)$-approximate solution A^* to the k-median problem with probability of at least $1 - \delta$ under the time complexity of $\mathcal{O}\left(\frac{n^2}{\epsilon^2 \mathrm{OPT}_k^m} \ln \frac{n}{\delta} + kn^2\right)$.

The proof of Theorem 5 is non-trivial. Its main idea is to carefully bound the errors of estimating $\Pr[u \sim v]$ using $\widehat{\Pr}[\mathcal{R}, u \sim v]$ for all $u, v \in V$, such that A^* admits the $1 - 1/e - \epsilon$ approximation ratio. The details can be found in [1].

3.2.2 Acceleration through Lazy Evaluations

It can be seen that the main operation in SearchKM is to greedily select $v^* = \arg\max_{v \in V \backslash C^*} \Delta(v|C^*)$ at each step, where C^* is the set of currently selected center nodes in A^*, and $\Delta(v|C^*) = \widehat{F}(\mathcal{R}, C^* \cup \{v\}) - \widehat{F}(\mathcal{R}, C^*)$ is the *marginal gain* of v with respect to C^*. SearchKM finds v^* by computing the exact values of marginal gains for all $v \in V \backslash C^*$, which results in quadratic time complexity with respect to n. In this section, we propose several optimized algorithms with better empirical time efficiency than SearchKM, while still keeping the $1 - 1/e - \epsilon$ approximation guarantee. The main idea of these algorithms is to correctly find v^* by computing fewer number of marginal gains.

Lazy Evaluation of the Marginal Gains. Our first optimization is shown in the LazyGreedy method in Algorithm 6, which (i) is functionally equivalent to the SearchKM algorithm and (ii) adopts the

Algorithm 4: $\mathsf{SearchKM}(G, k, \mathcal{R})$

```
C* ← Greedy(G, k, F̂(R, ·));  A* ← D(R, C*)
return A*
```

Algorithm 5: $\mathsf{Greedy}(G, k, F(\cdot))$

```
C ← ∅
while |C| < k do
    Find u ∈ V\C such that F(C ∪ {u}) − F(C) is maximized;
    C ← C ∪ {u}
return C
```

Algorithm 6: LazyGreedy$(G, k, \mathcal{R})$

```
1  C* ← ∅;
2  for (i = 1; i ≤ k; i = i + 1) do
3      if i = 1 then
4          foreach v ∈ V do UB(v) ← F̂(R, {v}) ;
5      else
6          Sort the nodes in V\C* into ⟨w_1, ··· , w_{n−i+1}⟩ such that
             UB(w_1) ≥ UB(w_2) ≥ ··· ≥ UB(w_{n−i+1})
7          for (j = 1; j ≤ n − i + 1; j = j + 1) do
8              UB(w_j) ← F̂(R, C* ∪ {w_j}) − F̂(R, C*)
9              if UB(w_j) ≥ UB(w_{j+1}) then break;
10     Find v* ∈ V\C* such that UB(v) is maximized;
11     C* ← C* ∪ {v*}; A* ← D(R, C*)
12 return A*
```

Algorithm 7: LazierGreedy$(G, k, \mathcal{R})$

```
1  C* ← ∅;
2  for (i = 1; i ≤ k; i = i + 1) do
3      if i = 1 then
4          foreach v ∈ V do
5              l ← the summation of the sizes of the connected
                components in I(R) that contain v;
6              UB(v) ← l/|R|
7      else
8          Call the procedure GetUB(C*, R) to update the value of
             UB(v) for all v ∈ V\C*
9          Run Lines 6-9 of LazyGreedy(G, k, R)
10     Find v* ∈ V\C* such that UB(v*) is maximized;
11     C* ← C* ∪ {v*}; A* ← D(R, C*)
12 return A*
```

Algorithm 8: GetUB$(C^*, \mathcal{R})$

```
1  foreach v ∈ V\C* do
2      Z ← the set of all connected components in I(R) that overlap
         C* ∪ {v};
3      h ← the summation of the sizes of the connected components in
         Z
4      UB(v) ← min{h/|R| − F̂(R, C*), UB(v)}
```

CELF framework proposed in [38]. Compared with SearchKM, LazyGreedy adopts a more efficient strategy to find v^*, as explained below.

LazyGreedy maintains a value $UB(v)$ for each $v \in V$, which serves as an upper bound of $\Delta(v|C^*)$. To find v^*, LazyGreedy evaluates $\Delta(v|C^*)$ for each $v \in V\backslash C^*$ based on the non-increasing order of $UB(v)$, and terminates this evaluation process immediately when $\Delta(v|C^*)$ is no less than the upper bound of the marginal gains of the unevaluated nodes (Lines 6-9). As such, LazyGreedy usually does not need to compute the marginal gains of all nodes in $V\backslash C^*$, and hence, achieves better time efficiency than SearchKM does.

LazyGreedy maintains the upper bounds of the marginal gains as follows. Initially, it sets $UB(v) = \widehat{F}(\mathcal{R}, \{v\})$ for all $v \in V$, which implies $UB(v) = \Delta(v|C^*)$ due to $C^* = \emptyset$ (Line 4). After that, it only updates $UB(v)$ when $\Delta(v|C^*)$ is re-computed (Line 8). This "lazy update" strategy ensures that $UB(v)$ is always an upper bound of $\Delta(v|C^*)$ when C^* changes, as $\Delta(v|C^*)$ is non-increasing due to the submodularity of function $\widehat{F}(\mathcal{R}, \cdot)$.

Enhancing LazyGreedy. LazyGreedy's benefit on time efficiency is mainly determined by its strategy on maintaining the values of $UB(v) : v \in V\backslash C^*$. Ideally, such a strategy should satisfy the following two requirements:

- Updating $UB(v)$ should be more efficient than directly computing the marginal gain of v;
- $UB(v)$ should be as tight as possible (i.e., it should be close to the marginal gain of v); Otherwise, LazyGreedy would still evaluate a large number of marginal gains.

Unfortunately, LazyGreedy often fails to satisfy the above requirements, due to which it still incurs considerable overheads on large graphs. Specifically, LazyGreedy does not provide an efficient method to compute $UB(v)$. For example, if we adopt a straightforward approach to compute $UB(u) = \widehat{F}(\mathcal{R}, \{u\})$ for any $u \in V$ in the initialization phase, we would need to evaluate $\widehat{\Pr}[u \sim u']$ for all $u' \in V\backslash\{u\}$, which incurs $\mathcal{O}(n^2|\mathcal{R}|)$ cost for computing $UB(v)$ for all $v \in V$. In addition, the value of $UB(v)$ is a very loose upper bound of $\Delta(v|C^*)$ in LazyGreedy, as it adopts the lazy update strategy described before. Due to this reason, we find in our experiments that, when selecting a new node, LazyGreedy still computes a sizable number of marginal gains of the nodes in V, and hence, offers inferior efficiency.

Based on the above observation, we propose a a new algorithm named LazierGreedy, which addresses the deficiencies of LazyGreedy in two aspects:

First, LazierGreedy provides an efficient method to compute $UB(v) : \forall v \in V$ in the initialization phase (Lines 4-6). More specifically, LazierGreedy initializes $UB(v)$ for all $v \in V$ by computing the summation of the sizes of the connected components that v lies in (Lines 5-6), which costs at most $\mathcal{O}(|\mathcal{R}|n)$ time. This process is correct because two nodes u and v are connected in a random sample if and only if they are in the same connected component.

Second, LazierGreedy adopts a more aggressive method for updating $UB(v) : \forall v \in V$ to improve time efficiency. Before selecting the ith node for any $i > 1$, LazierGreedy invokes the GetUB procedure to compute new upper bounds of the marginal gains for all $v \in V$ (Line 8). More specifically, GetUB first computes a value h by summing the sizes of the connected components overlapping $C^* \cup \{v\}$, which guarantees that $\widehat{F}(\mathcal{R}, C^* \cup \{v\}) \leq h/|\mathcal{R}|$, and then set $UB(v) = \min\{h/|\mathcal{R}| - \widehat{F}(\mathcal{R}, C^*), UB(v)\}$ (see Line 4 of Algorithm 8).

Compared with the "lazy updating" method of LazyGreedy described before, LazierGreedy updates $UB(v) : \forall v \in V$ immediately when a new node is added into C^*, so the upper bounds got by GetUB can be tighter than those used in LazyGreedy. Moreover, the time complexity of GetUB is only $\mathcal{O}(|\mathcal{R}|n)$, which is much less than the quadratic time complexity required to compute the exact marginal gains of the nodes in $V\backslash C^*$. After getting the upper bounds, LazierGreedy selects v^* by computing the exact marginal gains in the same way as that in LazyGreedy (Line 9), so the number of exact marginal gains computed in LazierGreedy could be less than that in LazyGreedy due to the tighter values of $UB(v) : v \in V$.

The following theorem establishes the correctness and time complexity of LazierGreedy:

THEOREM 6. *The* LazierGreedy *algorithm is correct, and has worst-case time complexity of* $\mathcal{O}(n^2|\mathcal{R}| + kn^2)$.

Lazy Sampling. According to Theorem 5, a straightforward approach is to invoke LazierGreedy or LazyGreedy using a set $\mathcal{R}$

Algorithm 9: SearchKM+$(G, k, \epsilon, \delta, \lambda)$

Input: G,k, $\epsilon, \delta, \lambda$
Output: A solution A^* to the k-median problem

1 $T_{max} \leftarrow \left\lceil \frac{2(7-7/e-4\epsilon)(2-1/e)n}{3\epsilon^2 k} \ln \frac{2n^2}{\delta} \right\rceil$; $T \leftarrow \lambda$
2 Generate two sets $\mathcal{R}_1$ and $\mathcal{R}_2$ of random samples of G, such that $|\mathcal{R}_1| = |\mathcal{R}_2| = T$;
3 $i_{max} \leftarrow \lceil \log_2(T_{max}/T) \rceil$;
4 **for** $i \leftarrow 1$ **to** i_{max} **do**
5 Call LazyGreedy$(G, k, \mathcal{R}_1)$ or LazierGreedy$(G, k, \mathcal{R}_1)$ to find a k-clustering A^*
6 $a \leftarrow \ln(3i_{max}/\delta)$; $\theta \leftarrow |\mathcal{R}_1|$
7 $\mathrm{lb}(A^*) \leftarrow \left(\sqrt{\widehat{KM}(\mathcal{R}_2, A^*) + \frac{2a}{9\theta}} - \sqrt{\frac{a}{2\theta}} \right)^2 - \frac{a}{18\theta}$
8 $\mathrm{ub}(A^o) \leftarrow \left(\sqrt{\frac{\widehat{KM}(\mathcal{R}_1, A^*)}{1-1/e} + \frac{8a}{9\theta}} + \sqrt{\frac{a}{2\theta}} \right)^2 - \frac{a}{18\theta}$
9 **if** $\mathrm{lb}(A^*)/\mathrm{ub}(A^o) \geq 1 - 1/e - \epsilon$ **or** $i = i_{max}$ **then**
10 **return** A^*
11 double the sizes of $\mathcal{R}_1$ and $\mathcal{R}_2$ with new random samples;

of T_{max} random samples (with OPT_k^m replaced by k/n), which has a worst-case time complexity of $\mathcal{O}(\frac{n^3}{k\epsilon^2} \ln \frac{n}{\delta} + kn^2)$. The time efficiency of this approach, however, can be further improved by adopting the OPIM framework [55] for submodular maximization to our problem, and the resulting algorithm is referred to as SearchKM+, which is shown by Algorithm 9.

The SearchKM+ algorithm uses a "lazy sampling" technique described as follows. It first sets an upper bound T_{max} for the number of generated random samples according to Theorem 5, and then generates two sets of random samples $\mathcal{R}_1$ and $\mathcal{R}_2$ such that $|\mathcal{R}_1| = |\mathcal{R}_2| = \lambda$, where λ can be any positive integer smaller than T_{max}. After that, SearchKM+ calls LazyGreedy or LazierGreedy using $\mathcal{R}_1$ to find a k-clustering A^*, and then uses $\mathcal{R}_1$ and $\mathcal{R}_2$ to find $\mathrm{ub}(A^o)$ and $\mathrm{lb}(A^*)$, respectively, where $\mathrm{ub}(A^o)$ denotes an upper bound of $KM(A^o)$ and $\mathrm{lb}(A^*)$ denotes a lower bound of $KM(A^*)$ (Lines 5-8). If $\mathrm{lb}(A^*)/\mathrm{ub}(A^o) \geq 1 - 1/e - \epsilon$ or $|\mathcal{R}_1| \geq T_{max}$, then SearchKM+ terminates and returns A^* (Line 10), otherwise it doubles the sizes of $\mathcal{R}_1$ and $\mathcal{R}_2$ and repeats the above process. The correctness of SearchKM+ relies on the property that A^* has a provable performance bound in the "sampling space", i.e., we have $\widehat{F}(\mathcal{R}, A^*) \geq (1-1/e)\widehat{F}(\mathcal{R}, \widehat{A}^o)$, where $\widehat{A}^o$ denotes a k-clustering such that $\widehat{F}(\mathcal{R}, \widehat{A}^o)$ is maximized.

It can proved that SearchKM+ achieves the $1-1/e-\epsilon$ approximation ratio with probability of at least $1-1/n$, under an expected time complexity of $\mathcal{O}\big(\frac{n^2}{\epsilon^2 \mathrm{OPT}_k^m} \ln \frac{n}{\delta} + kn^2 \ln n\big)$. This time complexity is derived based on the $\mathcal{O}(n^2|\mathcal{R}| + kn^2)$ time complexity of LazierGreedy and the fact that the expected number of generated random samples in SearchKM+ is $\mathcal{O}\big(\frac{1}{\epsilon^2 \mathrm{OPT}_k^m} \ln \frac{n}{\delta}\big)$. The full proofs for these theoretical bounds can be found in our technical report [1].

In summary, the time efficiency of SearchKM+ is optimized in two major aspects: 1) The "lazy evaluation" method adopted by LazierGreedy can greatly reduce the number of exact marginal gains to be evaluated; 2) The "lazy sampling" method described above makes it possible for SearchKM+ to find a satisfying solution using less than T_{max} random samples. Due to these reasons, the practical running time of SearchKM+ in experiments is much smaller than a cubic running time with respect to n.

3.3 Comparison with Existing Work

The results described above advance the state of the art [11] in three aspects. First, Ceccarello et al. [11] conjecture that the k-median problem is NP-hard when there is a connectivity oracle. We prove that this conjecture is true in Theorem 2.

Second, when a connection oracle is unavailable, our algorithm offers an approxiation ratio of $1 - 1/e - \epsilon$, which is significantly better than the $\mathcal{O}(\frac{(\mathrm{OPT}_k^m)^2}{(1+\gamma)^3 \ln^3 n})$ approximation ratio achieved by the solutions in [11]. For example, if $n = 100$ and $\mathrm{OPT}_k^m = 0.9$, then the approximation ratio in [11] is no more than 0.01, which is much smaller than $1 - 1/e - \epsilon$ for reasonable ϵ.

Third, in practice, the k-median algorithm proposed in [11] (i.e., ACP) incurs prohibitive running time to achieve non-trivial approximations, even for moderate graphs with about $20K$ nodes (see Section 6). In contrast, our algorithm runs much faster than ACP while achieving $(1-1/e-\epsilon)$-approximations, due to the optimization techniques discussed previously.

4. SOLVING THE K-CENTER PROBLEM

This section addresses the uncertain k-center problem. Section 4.1 presents an approximation algorithm assuming a connectivity oracle, while Section 4.2 proposes a solution without the oracle. Section 4.3 compares our solutions with those in [11].

4.1 k-Center Algorithms with an Oracle

Ceccarello et al. [11] present a rather complex algorithm for uncertain k-center that repeatedly identifies candidate k-clusterings until a satisfying solution is found (see Sec. 2.2). In this section, we propose a much simpler algorithm with better approximation guarantees. The main idea of our algorithm is to transform the uncertain k-center problem into a conventional metric k-center problem.

To explain, we first revisit a relevant result in [11], which shows that although the connection probabilities in uncertain graphs do not obey the triangle inequality, they have the following property:

$$\forall u, v, w \in V : \Pr[u \sim w] \geq \Pr[u \sim v] \cdot \Pr[v \sim w] \qquad (17)$$

Based on Eqn. (17), we define $d(u, v) = -\ln \Pr[u \sim v]$ and $d_v(C) = \min_{u \in C} d(u, v)$ for any $u, v \in V$ and any $C \subseteq V$. It can be seen from Eqn. (17) that $d(\cdot)$ is a metric, i.e.,

$$\forall u, v, w \in V : d(u, w) \leq d(u, v) + d(v, w) \qquad (18)$$

Now consider the following problem:

$$\textbf{Minimize } \max_{v \in V} d_v(C); \quad \textbf{s.t. } |C| = k;\ C \subseteq V \qquad (19)$$

As $d(\cdot)$ is a metric, the above problem is exactly a traditional metric k-center problem, which admits 2-approximations [18, 60]. Moreover, it can be verified that C_{kc}^o is also an optimal solution to the above problem. Therefore, we can use any existing 2-approximate metric k-center algorithm to address uncertain k-center, as shown in the following theorem.

THEOREM 7. *Let C^* be a 2-approximate solution to the problem in Equation* (19)*, and B^* be the k-clustering induced by C^*. Then, we have $KC(B^*) \geq (\mathrm{OPT}_k^c)^2$.*

PROOF. Note that $d_v(C^*) = -\ln f_v(C^*)$ for any $v \in V$. Let $C^\dagger$ denote an optimal solution to the problem in Equation (19). Then, we have

$$\max_{v \in V} d_v(C^*) \leq 2 \max_{v \in V} d_v(C^\dagger) \leq 2 \max_{v \in V} d_v(C_{kc}^o), \qquad (20)$$

which implies:

$$\max\nolimits_{v \in V} [1/f_v(C^*)] \leq \max\nolimits_{v \in V} [1/f_v(C_{kc}^o)]^2,$$
$$\min\nolimits_{v \in V} f_v(C^*) \geq \min\nolimits_{v \in V} [f_v(C_{kc}^o)]^2,$$

As $\min_{v\in V} f_v(C^*) = KC(B^*)$ and $\min_{v\in V}[f_v(C^o_{kc})]^2 = (\mathrm{OPT}^c_k)^2$, the theorem follows. □

Note that the 2-approximation ratio for traditional metric k-center problem is tight unless P=NP [60]. As we have transformed the k-center problem in uncertain graphs into a metric k-center problem, we conjecture that the approximation ratio shown in Theorem 7 is also optimal.

4.1.1 Hardness on Better Approximation Ratio

The approximation ratio proposed in Theorem 7 is OPT^c_k, which depends on G and could be small for certain input graphs. Therefore, we ask whether there could exist an algorithm with a data-independent approximation ratio for the k-center problem. Unfortunately, we have the following negative result.

THEOREM 8. *Unless P=NP, no polynomial-time algorithm with the connectivity oracle can find a α-approximate solution to uncertain k-center, for any positive α.*

As Theorem 8 proves that the uncertain k-center problem is NP-hard to approximate within any given $\alpha > 0$, we further ask whether there exists a *bi-criteria approximation algorithm* for the problem. That is, if we allow the algorithm to use more than k center nodes to find a clustering B, is possible that $KC(B)$ could be close to OPT^c_k. Unfortunately, the following theorem shows that if we are to ensure that $KC(B)$ is close to OPT^c_k, we have to increase the number of center nodes to at least $\Omega(k \log n)$.

THEOREM 9. *Assume that there is a connectivity oracle and P≠NP. Let B be a clustering found by any polynomial-time bi-criteria approximation algorithm for uncertain k-center. If $KC(B) \geq \mathrm{OPT}^c_k$, then the number of center nodes in B is at least $\Omega(k \log n)$.*

Due to the above results, we do not investigate data-independent or bi-criteria approximation algorithms for uncertain k-center.

4.2 k-Center Algorithms without an Oracle

In this section, we present uncertain k-center algorithms without a connectivity oracle.

4.2.1 A Greedy Algorithm

A straightforward idea is that we can first generate a set of random samples of G, and then follow the method in Section 4.1 to transform the k-center problem into a traditional metric k-center problem using these samples. This idea, however, does not work, since the metric property no longer holds on random samples of G. In other words, without a connectivity oracle, the uncertain k-center problem is drastically different from the metric k-center problem. More specifically, it is possible that

$$\exists u, v, w \in V : \widehat{\Pr}[\mathcal{R}, u \sim w] < \widehat{\Pr}[\mathcal{R}, u \sim v] \cdot \widehat{\Pr}[\mathcal{R}, v \sim w]$$

holds for any set $\mathcal{R}$ of random samples of G, which is to the contrary of Equation (17). For example, suppose that either u is connected to v or v is connected to w in each random sample in $\mathcal{R}$, but u is never connected to w in these samples. In this case, we have $\widehat{\Pr}[\mathcal{R}, u \sim w] = 0$ and $\widehat{\Pr}[\mathcal{R}, u \sim v] \cdot \widehat{\Pr}[\mathcal{R}, v \sim w] > 0$.

Fortunately, although our problem cannot be transformed into a metric k-center problem without a connectivity oracle, we find that a greedy algorithm SearchKC (shown by Algorithm 10) can achieve an approximation ratio close to the one with a connectivity oracle. The intuition of SearchKC is as follows. It first adds an arbitrary node v^* into the set C^*, and then greedily selects a node that is "most unlikely" to connect to the nodes in C^* (according to the observations on $\mathcal{R}$, see Line 3). When $|C^*| = k$, SearchKC returns $B^* = D(\mathcal{R}, C^*)$ as the approximate solution.

Algorithm 10: SearchKC$(G, k, \mathcal{R})$

```
1  Select an arbitrary node v ∈ V and set C* = {v}
2  while |C*| < k do
3      Find v* ∈ V\C* such that f̂_{v*}(R, C*) is minimized;
4      C* ← C* ∪ {v*}
5  B* ← D(R, C*)
6  return C*, B*
```

Algorithm 11: SearchKC+(G, k, ϵ, δ)

```
1  p_min = ∏_{e∈E}(p(e))^2; R_0 ← ∅; U_0 ← ∅; i ← 0
2  repeat
3      i ← i + 1; q_i ← 2^{-i}; δ_i ← 3δ/(π^2 i^2); R_i ← R_{i-1} ∪ U_{i-1}
4      ℓ(i) ← 4(6+ε)/(3ε^2(1-ε)q_i) ln n(n-1)/δ_i;
5      if |R_i| < ℓ(i) then
6          Add more random samples into R_i until |R_i| ≥ ℓ(i)
7      (C*_i, B*_i) ← SearchKC(G, k, R_i)
8      Generate another set U_i of random samples such that |U_i| = |R_i|
9      foreach (u, v) ∈ B*_i ∧ u ≠ v do
10         γ_i ← ln(C(n,2)/δ_i)/|U_i|
11         z(u, v) ← ( sqrt(P̂r[U_i, u ∼ v] + 2γ_i/9) − sqrt(γ_i/2) )^2 − γ_i/18
12     l_i ← min{z(u, v) | (u, v) ∈ B*_i ∧ u ≠ v}
13 until l_i ≥ (1 − ε)q_i ∨ q_i ≤ p_min ;
14 return (C*, B*) = (C*_i, B*_i)
```

We note that the SearchKC algorithm is similar in spirit to a metric k-center algorithm proposed in [18]. However, as explained above, we do not have a metric without a connectivity oracle, and hence, the performance analysis in [18] cannot be applied for SearchKC. Instead, we provide the following theorem on the approximation guarantee of SearchKC.

THEOREM 10. *Given any $\epsilon, \delta \in (0, 1)$ and any set $\mathcal{R}$ of random samples of G satisfying*

$$|\mathcal{R}| \geq \frac{4(6+\epsilon)}{3\epsilon^2(\mathrm{OPT}^c_k)^2} \ln \frac{n(n-1)}{\delta}, \tag{21}$$

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

The intuition of Theorem 10 can be roughly explained as follows. When $|C^*|$ increases, the value of $\widehat{f}_v(\mathcal{R}, C^*)$ is non-decreasing for any $v \in V$. Thus, when $|C^*| = k$, $KC(B^*) = \min_{v\in V} \widehat{f}_v(\mathcal{R}, C^*)$ should be sufficiently large. Indeed, as $|\mathcal{R}|$ is also sufficiently large (according to Eqn. (21)), we can use the greedy selection rule in Line 3 and some concentration bounds to prove that $\min_{v\in V} \widehat{f}_v(\mathcal{R}, C^*) \geq (1 - \epsilon/2)(\mathrm{OPT}^c_k)^2$ with high probability. With this result, we then use the concentration bounds again to prove that $KC(B^*) \geq (1-\epsilon)(\mathrm{OPT}^c_k)^2$ w.h.p.

As OPT^c_k is unknown, a possible way to implement SearchKC using Theorem 10 is to find some trivial lower bound of $(\mathrm{OPT}^c_k)^2$. For example, it can be verified that $p_{min} = \prod_{e\in E}(p(e))^2$ is a lower bound of $(\mathrm{OPT}^c_k)^2$. However, such lower bounds are excessively loose, and hence, could result in a prohibitive number of random samples. In the sequel, we will study how to make further optimizations to address this problem.

4.2.2 Optimizations

One possible optimization idea is that we could leverage the OPIM framework [55] to reduce the number of generated random samples. Specifically, we could set a large upper bound of $|\mathcal{R}|$ using p_{min}, and then progressively generate random samples until a satisfying solution is found. However, as described in Sec. 3.2, the stopping rule of OPIM requires that we have a provable approximation ratio in the "sampling space", which implies that $\min_{v\in V} \widehat{f}_v(\mathcal{R}, C^*)$ should have a provable approximation guarantee with respect to $\min_{v\in V} \widehat{f}_v(\mathcal{R}, \widehat{C}^o)$, where C^* denotes the set of center nodes returned by SearchKC$(G, k, \mathcal{R})$ and

$$\widehat{C}^o = \arg\max_{C\subseteq V\wedge|C|=k} \min_{v\in V} \widehat{f}_v(\mathcal{R}, C).$$

Unfortunately, we do not have such an approximation assurance in the sampling space, as we do not have a metric without the connectivity oracle (see Sec. 4.2.1). Therefore, the OPIM framework cannot be applied to the k-center problem.

Based on the above discussion, we design a new "trial and error" algorithm named SearchKC+. In each iteration i, SearchKC+ uses $q_i = 2^{-i}$ as a guess of the lower bound of $(\mathrm{OPT}_k^c)^2$, and then generates $\ell(i)$ random samples according to Theorem 10 to find a solution B_i^* using SearchKC (Lines 3-7). After that, the SearchKC+ algorithm generates another set $\mathcal{U}_i$ of random samples to derive a lower bound l_i of $KC(B_i^*)$ (Lines 8-12), where l_i is derived according to the concentration bounds. The algorithm returns B_i^* as an approximate solution when $l_i \geq (1-\epsilon)q_i$ or $q_i \leq p_{min}$ (Line 13). Note that SearchKC+ differs from SearchKM+ in both the "trial phase" (i.e., how to find a approximate solution B_i^*) and the "error phase" (i.e., how to judge whether B_i^* is a good solution). The following theorem shows the correctness of SearchKC+:

THEOREM 11. *With probability of at least $1-\delta$,* SearchKC+ *returns a k-clustering B^* satisfying $KC(B^*) \geq (1-\epsilon)(\mathrm{OPT}_k^c)^2$.*

The correctness of SearchKC+ can be roughly explained as follows. Note that the stopping rule of SearchKC+ is $l_i \geq (1-\epsilon)q_i \vee q_i \leq p_{min}$ (Line 13). If SearchKC+ stops with $q_i \leq p_{min}$, then the set of generated random samples in SearchKC+ satisfies Equation (21) due to $p_{min} \leq (\mathrm{OPT}_k^c)^2$, and hence SearchKC+ must return a satisfying solution according to Theorem 10. If SearchKC+ stops with $q_i > p_{min}$, then we must have $l_i \geq (1-\epsilon)q_i$ and q_i must satisfy one of the following cases: (case 1) $q_i \geq (\mathrm{OPT}_k^c)^2$: in this case, $KC(B_i^*) \geq (1-\epsilon)(\mathrm{OPT}_k^c)^2$ must hold with high probability, as l_i can be proved to be a lower bound of $KC(B_i^*)$ with high probability; (case 2) $q_i < (\mathrm{OPT}_k^c)^2$: in this case, the set of generated random samples in SearchKC+ must satisfy Equation (21) due to Line 4, so we also have $KC(B_i^*) \geq (1-\epsilon)(\mathrm{OPT}_k^c)^2$ w.h.p. due to Theorem 10. In summary, SearchKC+ always returns a solution satisfying Theorem 11.

Finally, we ask whether SearchKC+ could generate too many random samples of G. Intuitively, when $q_i < (\mathrm{OPT}_k^c)^2$, we must have $KC(B_i^*) \geq (1-\epsilon)(\mathrm{OPT}_k^c)^2 \geq (1-\epsilon)q_i$ with high probability, and hence, the probability that SearchKC+ does not stop would decrease exponentially with $(\mathrm{OPT}_k^c)^2/q_i$ according to the concentration bounds. Therefore, we can prove that the expected number of generated random samples in SearchKC+ is close to the lower bound proposed in Equation (21).

THEOREM 12. *When $\epsilon \leq 1/2$ and $\delta \leq 1/3$, the expected number of generated random samples in Algorithm 11 is at most* $\mathcal{O}\left(\frac{1}{\epsilon^2(\mathrm{OPT}_k^c)^2}\left(\ln\frac{n}{\delta} + \ln\ln\frac{1}{\mathrm{OPT}_k^c}\right)\right)$

Based on Theorem 12, we also provide the time complexity of the SearchKC+ algorithm.

THEOREM 13. *The expected time complexity of* SearchKC+ *is* $\mathcal{O}\left(\frac{kn+m}{\epsilon^2(\mathrm{OPT}_k^c)^2}\left(\ln\frac{n}{\delta} + \ln\ln\frac{1}{\mathrm{OPT}_k^c}\right)\right)$ *when $\epsilon \leq \frac{1}{2}$ and $\delta \leq \frac{1}{3}$.*

We roughly explain Theorem 13 as follows. The SearchKC algorithm can be run in $\mathcal{O}(kn|\mathcal{R}|)$ time and generating a random sample causes $\mathcal{O}(n+m)$ time. The expected number of generated random samples is upper-bounded by Theorem 12. So the time complexity of SearchKC+ can be derived by using these results.

4.3 Comparison with Existing Work

Compared with the solutions in [11], our results for uncertain k-center are considerably improved in the following aspects. First, [11] only proves the hardness of uncertain k-center, where we further establish the inapproximability of the problem.

Second, our k-center algorithms are much simpler than the solutions in [11], and our approximation guarantees are strictly better than the $\frac{\mathrm{OPT}_k^c}{1+\gamma}$ approximation ratio in [11], since setting $\gamma = 0$ in their algorithm results in infinite running time.

Third, [11] does not provide a practical algorithm to achieve their approximation ratio, since they require a known lower bound of OPT_k^c, which is impractical. In contrast, we present a practical algorithm (i.e., Algorithm 11) to achieve our approximation guarantee without prior knowledge of OPT_k^c. In addition, we present an upper bound of the expected number of random samples generated in our algorithm, and also analyze our algorithm's time complexity.

5. RELATED WORK

Recently, there has been growing interest on querying and mining uncertain graphs. Studies in this area have investigated various topics such as reliable queries, pattern matching and similarity search [29,33,34,49], but there are relatively few studies on clustering uncertain graphs. Kollios et al. [36] aim to find a *cluster graph* of an uncertain graph G, such that the expected graph edit distance between the cluster graph and the input graph is minimized. However, the problem definition in [36] is very different from ours, as the cluster graph in their work is defined as a clique-cover of the nodes of the input graph, and the number of cliques in their problem can be arbitrary. Gu et al. [20] re-investigate the problem in [36] under a more general uncertain graph model with correlated edges. Liu et al. [40] use entropy to measure "purity" and "size balance" of a k-clustering, where "purity" reflects the adherence of the clustering to the fragments of a random possible world of G, and "size balance" reflects the distribution of node numbers in each cluster. They propose algorithms to seek a k-clustering such that the difference of purity and size balance is minimized, but their algorithms do not have guaranteed performance bounds and have scalability issues [11]. To the best of our knowledge, the most close work to our paper is [11], which is discussed extensively in Sec. 2.2.

The influence maximization (IM) [8, 55–58] problem is also defined on uncertain graphs, where the goal is to find k "seed nodes" to maximize the expected number of nodes reachable from the seed nodes. Note that the IM problem is essentially different from our problems, and none of the existing IM algorithms has proposed a method to partition the graph nodes into k clusters to optimize the objective values of k-median or k-center problems. Therefore, although we have adapted the OPIM framework [55] for the IM problem to reduce the number of generated random samples in the k-median problem, the focus of our approach is to reveal the submodularity of the k-median problem and propose efficient and practical algorithms for it with a provable performance ratio.

There also exist a lot of studies on submodular optimization techniques, and a good survey can be found in [37]. In particular, Leskovec et al. [38, 45] present some methods for speeding up the

Table 3: Characteristics of the datasets

Dataset	$\|V\|$	$\|E\|$
FruitFly	3.8K	3.7K
Dartmouth	6.1K	1.8M
Biomine	8.7K	14.4K
Flixster	95.9K	484.8K
DBLP	636.7K	2.4M

greedy algorithm for submodular maximization. However, the algorithms proposed in [45] are totally different from ours and only has an expected approximation ratio. Besides, although the CELF framework [38] is adopted by LazyGreedy in Sec. 3.2.2, the practical performance of LazyGreedy is far from satisfactory (see Sec. 6) due to the reasons explained in Sec. 3.2.2. Therefore, the focus of our Sec. 3.2.2 is to design a novel LazierGreedy procedure with much better time efficiency. Due to the wide applications of submodular functions, we believe that the idea of our LazierGreedy procedure has the potential to be applied to other problems.

6. PERFORMANCE EVALUATION

6.1 Experimental Setting

Datasets: We use 5 datasets in our experiments. FruitFly [26] and BioMine [26,34] are both real Protein-Protein Interaction networks where the nodes represent proteins and the edges represent the interactions between proteins. The probability associated with each edge denotes the confidence that the interaction actually exists. The Flixster dataset [34, 51] comes from a social movie website (flixster.com), where each node represents a user and each edge models the interactions between two nodes. Following [41,51], the probability associated with each edge in Flixster is learned from the users' action logs using the method proposed in [19]. As Flixster is a directed graph, we turn it into an undirected one by removing the edge directions and selecting an edge uniformly at random if there are two edges between any two nodes. Dartmouth [23] is a real mobility trace dataset which uses the SNMP logs of WiFi LAN in Dartmouth college to record the events that two mobile devices appeared at the same location. Following [23], we model each mobile device in Dartmouth as a node, and the probability associated with each edge is the frequency that the two mobile devices meet (i.e., appear in the same location) according to the logs. DBLP is a collaboration network where each node represents an author and co-authors are adjacent in the network. Following a large body of work on uncertain graphs [7, 11, 26, 29, 40, 46, 49], the probability associated with each edge (u, v) in DBLP is set to $1 - \exp\{-x/2\}$, where x is the number of papers co-authored by u and v. Following [11], our experiments are conducted on a connected component of each dataset, and the characteristics of these connected components are listed in Table 3.

Implemented Algorithms: To the best of our knowledge, only [11] has proposed k-median/k-center algorithms for uncertain graphs. So we use their algorithms (i.e., ACP and MCP) as baselines. The codes of ACP and MCP are provided by the authors of [11] and can be downloaded from [2].

We also implement three of our algorithms, including Lazy_SearchKM+, Lazier_SearchKM+ and SearchKC+. Both Lazy_SearchKM+ and Lazier_SearchKM+ adopt the lazy sampling framework SearchKM+ described in Sec. 3.2.2, with the difference that Lazy_SearchKM+ calls the lazyGreedy procedure while Lazier_SearchKM+ calls lazierGreedy. SearchKC+ is the algorithm shown in Algorithm 11.

Parameters: Note that our k-median and k-center algorithms achieve the $1 - 1/e - \epsilon$ and $(1 - \epsilon)\mathrm{OPT}_k^c$ approximation ratios, respectively, with probability of at least $1 - \delta$. In the experiments, we set $\epsilon = 0.1$ and $\delta = 1/n$ for all our algorithms. The parameter λ in SearchKM+ is set to 1000.

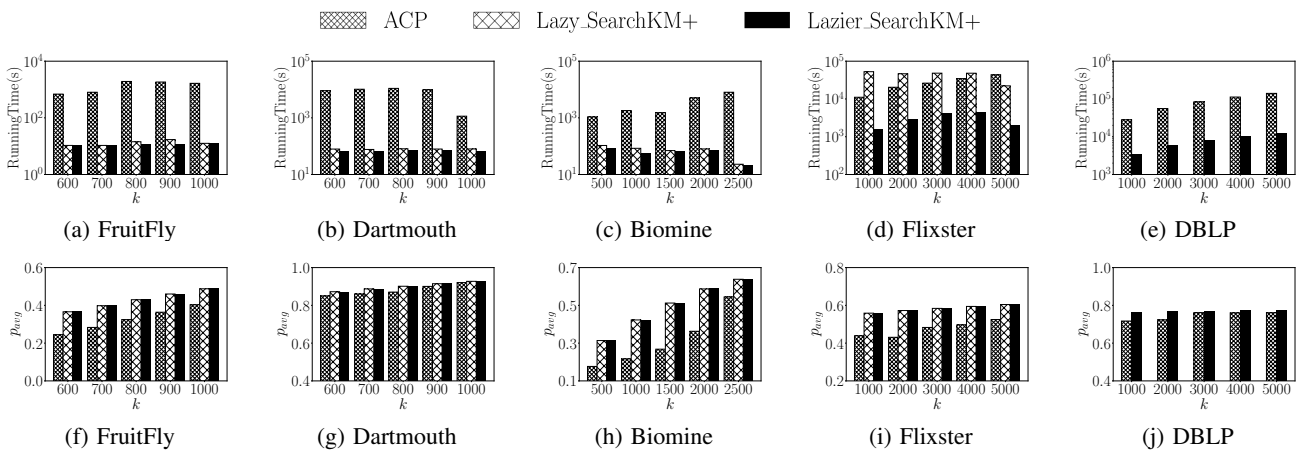
Recall that the approximation ratios of the ACP and MCP algorithms are shown in Table 1, and achieving these approximation ratios requires some lower bounds for OPT_k^m and OPT_k^c. Following the suggestions in [11], we use k/n as the lower bound for OPT_k^m in the ACP algorithm, and use $p_L = 10^{-4}$ as the lower bound of $(\mathrm{OPT}_k^c)^2$ in the MCP algorithm. We also follow [11] to set $\gamma = 0.1$ in ACP and MCP. For fair comparison, we set $\epsilon = 0.01$ in MCP such that MCP achieves the same approximation ratio with SearchKC+ when $p_L \leq (\mathrm{OPT}_k^c)^2$. However, the ACP algorithm cannot achieve the same approximation ratio with Lazy_SearchKM+ or Lazier_SearchKM+ no matter how we set the value of ϵ in ACP, as its approximation ratio is much worse than ours. Therefore, we follow [2] to set $\epsilon = 0.1$ in ACP.

As described in Sec. 2.2, both ACP and MCP use the input parameter α to control the size of their searching space. Following [11], we set $\alpha = 1$ for MCP. However, we should set $\alpha = n$ in ACP to achieve its approximation ratio shown in Table 1 [11], which results in prohibitive running time of ACP. Indeed, ACP cannot return a solution within 72 hours for the Dartmouth dataset, and its running time is even longer for the Biomine, Flixster and DBLP datasets with $\alpha = n$. Therefore, we only set $\alpha = n$ in ACP for the FruitFly dataset, and set $\alpha = 1$ in ACP for the other datasets, such that ACP can return a solution in reasonable time for comparison.

As explained in Sec. 2.2, the ACP algorithm presented in [11] requires that the number r of random samples generated in each iteration should satisfy Eqn. (1). However, in their implementation code [2], they have replaced this equation by $r = \frac{1}{10q\epsilon^2} \ln 100$. This replacement drastically reduce the random samples generated in ACP (hence the running time) but cannot lead to any performance guarantee. A similar problem also occurs in the implementation of MCP in [2]. In our experiments, we have fixed these problems by honestly setting r according to the equations presented in [11], which reveals that the running time of ACP and MCP is much larger than that reported in [11].

Other Settings: Following [11], we implement all the algorithms using OpenMP [3], a multi-threaded coding interface, and the maximum number of threads used by each algorithm is set to 8 for fair comparison. All our experiments are conducted on a Linux server with Intel(R) Xeon(R) E5-2650 v2 2.6GHz CPU and 128GB memory. When any implemented algorithm exceeds the memory limit, it returns the best solution found so far. For each solution returned by any implemented algorithm, we use 10,000 times of monte-carlo simulations to unbiasedly evaluate the solution's objective function value. All the reported data in our figures are the average of 5 runs.

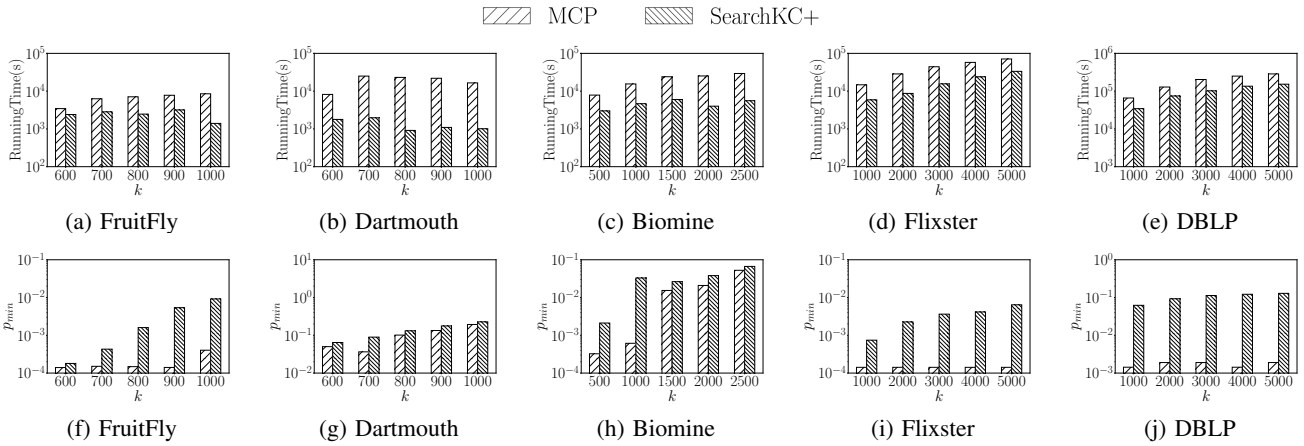
All our algorithms are implemented using C++. To facilitate the time efficiency of our algorithms, we have used the following data structures to store the nodes and the generated random samples. Each connected component C in the generated random samples is stored as a dynamic array (i.e., a *vector* object in C++) that contains pointers to all the vector objects storing the nodes in C. Meanwhile, each node $v \in V$ is stored as a vector object that contains the pointers to all the vector objects storing the connected components that v lies in. By such a double-linked structure as an "index", we can efficiently implement some key operations in our algorithms. For example, when we compute the summation of the sizes of connected components containing node v in Algorithm 7, we can directly find the connected components containing v and



(a) FruitFly (b) Dartmouth (c) Biomine (d) Flixster (e) DBLP

(f) FruitFly (g) Dartmouth (h) Biomine (i) Flixster (j) DBLP

Figure 1: Compare p_{avg} and running time of the implemented algorithms for the k-median problem



(a) FruitFly (b) Dartmouth (c) Biomine (d) Flixster (e) DBLP

(f) FruitFly (g) Dartmouth (h) Biomine (i) Flixster (j) DBLP

Figure 2: Compare p_{min} and running time of the implemented algorithms for the k-center problem

get the sizes of those components, without checking every generated random samples from scratch.

6.2 Experimental Results

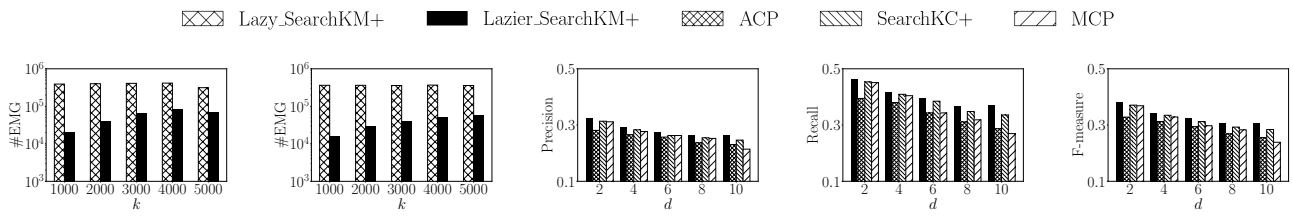
6.2.1 Overall Performance

In Figs. 1(a)-1(e), we compare the running time of the implemented k-median algorithms. As described in Sec. 6.1, we have adopted a parameter setting favorable to ACP's running time. Even under this setting, Lazier_SearchKM+ runs much faster than ACP, as our algorithms generate fewer random samples and also leverage the lazy evaluation methods described in Sec. 3.2.2 for acceleration. Moreover, it can be seen that Lazy_SearchKM+ runs much slower than Lazier_SearchKM+, and Lazy_SearchKM+ even cannot finish within 72 hours for the largest dataset DBLP. This demonstrates the effectiveness of the LazierGreedy algorithm described in Sec. 3.2.2, as Lazier_SearchKM+ leverages LazierGreedy to significantly reduce the number of marginal gains to be evaluated.

In Figs. 1(f)-1(j), we study the performance of ACP, Lazy_SearchKM+ and Lazier_SearchKM+ on the objective value of the k-median problem, which is denoted by P_{avg}. We note that ACP cannot achieve its claimed approximation ratio for the Dartmouth, Biomine, Flixster and DBLP datasets due to its prohibitive running time and memory overflow issues. In contrast, Lazier_SearchKM+ achieves the $1 - 1/e - \epsilon$ approximation ratio with high probability for all the 5 datasets. The results in Figs. 1(f)-1(j) also show that the P_{avg} values of Lazy_SearchKM+ and Lazier_SearchKM+ are both larger than those of ACP, irrespective of whether ACP achieves its claimed approximation ratio or not. Note that P_{avg} is the average connection probability between each node $v \in V$ and its center node. Therefore, even a slight advantage on P_{avg} implies that there are a lot of nodes in V whose connection probabilities to their center nodes are increased. Moreover, Figs. 1(f)-1(j) reveal that Lazy_SearchKM+ and Lazier_SearchKM+ achieve almost identical performance on P_{avg}, as these two algorithms are functionally equivalent.

In Figs. 2(a)-2(e), we compare the running time of the two implemented k-center algorithms, and the experimental results show that SearchKC+ runs much faster than MCP on all the five datasets. This can be explained by the reason that SearchKC+ generates much fewer random samples to achieve its approximation ratio, and it also does not need to iteratively call a time-consuming "partial clustering" procedure as that in MCP to seek the optimal solution.

In Figs. 2(f)-2(j), we compare the performance of SearchKC+ and MCP on the P_{min} value, which denotes the objective function value of the k-center problem. We find that MCP cannot achieve guaranteed performance ratio due to memory issues for all the 5 datasets except FruitFly. In contrast, SearchKC+ achieves the $(1 - \epsilon)\mathrm{OPT}_k^c$ approximation ratio with high probability for all datasets except DBLP, as the value of $(\mathrm{OPT}_k^c)^2$ is too small in DBLP and hence cause memory overflow issue to SearchKC+. Nevertheless, SearchKC+ outperforms MCP on the P_{min} value for all the 5 datasets.



(a) Flixster (b) DBLP (a) Precision (b) Recall (c) F-measure

Figure 3: The number of evaluated EMGs

Figure 4: Comparisons on predicting protein complexes

In Fig. 3, we plot the numbers of Exact Marginal Gains (EMGs) evaluated by Lazier_SearchKM+ and Lazy_SearchKM+ in their greedy searching process, using the Flixster and DBLP datasets. As Lazy_SearchKM+ has prohibitive running time on the DBLP dataset, we limit its running time to 48 hours on DBLP, so the true numbers of EMGs evaluated by Lazy_SearchKM+ on DBLP are even larger than those plotted in Fig. 3. It can be seen that Lazier_SearchKM+ computes much fewer EMGs than Lazy_SearchKM+ does, which corroborates the analysis in Sec. 3.2.2 that Lazier_SearchKM+ achieves better time efficiency by evaluating EMGs in a "lazier" way.

In summary, the experimental results show that our algorithms greatly outperform the algorithms in [11] both on the running time and on the quality of clustering results, and also corroborate the effectiveness of the proposed optimization methods for clustering.

6.2.2 Case Study on Detecting Protein Complexes

In this section, we consider the scenario of detecting protein complexes in Protein Protein Interaction (PPI) networks, and provide a ground-truth based performance evaluation of our algorithms. We use a popular dataset named Gavin [17], which is a real PPI network on yeast Saccharomyces cerevisiae consisting of 1430 proteins and 6531 interactions among the proteins. We also adopt the manually curated protein complexes published by CYC2008 [50] as the ground truth, which has been widely used in the literature as the golden standard for detecting protein complexes in yeast [25, 39, 47, 61].

We adopt several widely-acknowledged metrics including Precision, Recall and F-Measure [9, 13, 39] to measure the quality of detected protein complexes, as explained below. Suppose that P is the collection of detected complexes and B is the collection of ground-truth complexes. For any $p \in P$ and $b \in B$, the "precision-recall product score" $PR(p,b) = \frac{|p \cap b|^2}{|p| \times |b|}$ measures how well p matches b. Given a threshold ω, p and b are considered to be matched if $PR(p,b) \geq \omega$. Following many related studies (e.g., [4, 24, 39]), we set $\omega = 0.2$. The Precision, Recall and F-measure with respect to P and B are then defined as:

$$\text{Precision} = |\{p | p \in P, \exists b \in B, PR(p,b) \geqslant \omega\}|/|P| \quad (22)$$

$$\text{Recall} = |\{b | b \in B, \exists p \in P, PR(p,b) \geqslant \omega\}|/|B| \quad (23)$$

$$\text{F-Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (24)$$

Ceccarello et al. [11] have extended their ACP/MCP algorithms to address the "d-hop constrained uncertain k-median and k-center problems", where the only discrepancy in problem definition is that $\Pr[u \sim v](\forall u, v \in V)$ is replaced by the probability that there is a path with length no more than d between node u and node v. They have proved some performance bounds for their extended algorithms, and shown that these algorithms perform well on detecting protein complexes. Following [11], we have also extended our algorithms to the d-hop constrained problems mentioned above and got better performance bounds than those in [11] (more details can be found in [1]). In Fig. 4, we compare our extended algorithms with those in [11] on detecting the complexes in Gavin dataset, where we follow [17] to set $k = 491$ and scale d from 2 to 10. It can be seen that our extended Lazier_SearchKM+ and SearchKC+ algorithms outperform the extended ACP and MCP algorithms, respectively, on all the considered metrics including Precision, Recall and F-measure. This demonstrates the superiority of our approach. Besides, Fig. 4 also shows that all the algorithms perform better when d gets smaller. This phenomenon is similar to that reported in [11], which demonstrates the benefit of holistically considering topological distances and connection probabilities in the scenario of detecting protein complexes with PPI networks.

6.2.3 Other Comparisons

We also use several metrics including DaviesBouldin index, Dunn index and Silhouette coefficient [22] to compare our algorithms with those in [11]. The experimental results show that the overall performance of our algorithms are better than ACP/MCP, and both the performance of our algorithms and ACP/MCP on these metrics could be affected by the specific datasets used. Due to the space constraints, the detailed experimental results on these metrics can be found in our technical report [1].

7. CONCLUSION

We have studied the k-median and k-center problems in uncertain graphs. We have thoroughly analyzed the hardness of these problems and proposed several novel algorithms for them with provable performance bounds. Compared to the existing work, our algorithms achieve better approximation ratios and are also more efficient. Finally, we conduct extensive experiments using public datasets to compare our algorithms with the existing ones, and the experimental results demonstrate the superiorities of our algorithms both on the running time and on the quality of clustering results.

Following some related studies [6, 11, 26, 29, 30, 46, 49], we have only considered edge uncertainty in this work. In our future work, we plan to study more clustering problems in uncertain graphs with node uncertainty or attribute uncertainty.

8. ACKNOWLEDGEMENT

This work is partially supported by National Natural Science Foundation of China under grant No. 61772491, No. 61472460, No. U1709217, No. 61873177, No. 61572342, by Natural Science Foundation of Jiangsu Province under grant No. BK20161256, by MOE, Singapore under grant MOE2017-T1-002-024, MOE2015-T2-2-069, by NUS, Singapore under an SUG, by NRF, Singapore under grant NRF-RSS2016-004, and by Anhui Initiative in Quantum Information Technologies under grant AHY150300.

9. REFERENCES

[1] https://www.dropbox.com/s/i12see0ochh26gw/CLUGFull.pdf.

[2] https://github.com/Cecca/ugraph.

[3] https://www.openmp.org/.

[4] G. D. Bader and C. W. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC bioinformatics*, 4:2, 2003.

[5] M. O. Ball. Computational complexity of network reliability analysis: An overview. *IEEE Transactions on Reliability*, 35(3):230–239, 1986.

[6] P. Boldi, F. Bonchi, A. Gionis, and T. Tassa. Injecting uncertainty in graphs for identity obfuscation. *PVLDB*, 5(11):1376–1387, 2012.

[7] F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich. Core decomposition of uncertain graphs. In *KDD*, pages 1316–1325, 2014.

[8] C. Borgs, M. Brautbar, J. T. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *SODA*, pages 946–957, 2014.

[9] S. Brohee and J. Van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC bioinformatics*, 7:488, 2006.

[10] J. Byrka, T. Pensyl, B. Rybicki, A. Srinivasan, and K. Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *SODA*, pages 737–756, 2015.

[11] M. Ceccarello, C. Fantozzi, A. Pietracaprina, G. Pucci, and F. Vandin. Clustering uncertain graphs. *PVLDB*, 11(4):472–484, 2017.

[12] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *SODA*, pages 642–651, 2001.

[13] H. N. Chua, K. Ning, W.-K. Sung, H. W. Leong, and L. Wong. Using indirect protein–protein interactions for protein complex prediction. *Journal of bioinformatics and computational biology*, 6(3):435–466, 2008.

[14] F. Chung and L. Lu. Concentration inequalities and martingale inequalities: a survey. *Internet Mathematics*, 3(1):79–127, 2006.

[15] K. Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM*, pages 27–34, 2003.

[16] X. Gao, Z. Chen, F. Wu, and G. Chen. Energy efficient algorithms for k -sink minimum movement target coverage problem in mobile sensor network. *IEEE/ACM Transactions on Networking*, 25(6):3616–3627, Dec 2017.

[17] A.-C. Gavin, M. Bösche, R. Krause, P. Grandi, M. Marzioch, A. Bauer, J. Schultz, J. M. Rick, A.-M. Michon, C.-M. Cruciat, et al. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415:141–147, 2002.

[18] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[19] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. In *WSDM*, pages 241–250, 2010.

[20] Y. Gu, C. Gao, G. Cong, and G. Yu. Effective and efficient clustering methods for correlated probabilistic graphs. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1117–1130, 2014.

[21] S. Guo, Y. Gu, B. Jiang, and T. He. Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links. In *MobiCom*, pages 133–144, 2009.

[22] Z. Halim and J. H. Khattak. Density-based clustering of big probabilistic graphs. *Evolving Systems*, pages 1–18, Mar 2018.

[23] B. Han, J. Li, and A. Srinivasan. Your friends have more friends than you do: Identifying influential mobile users through random-walk sampling. *IEEE/ACM Transactions on Networking*, 22(5):1389–1400, 2014.

[24] A. L. Hu and K. C. Chan. Utilizing both topological and attribute information for protein complex identification in ppi networks. *IEEE/ACM transactions on computational biology and bioinformatics*, 10(3):780–792, 2013.

[25] L. Hu, X. Yuan, X. Liu, S. Xiong, and X. Luo. Efficiently detecting protein complexes from protein interaction networks via alternating direction method of multipliers. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2018.

[26] X. Huang, W. Lu, and L. V. Lakshmanan. Truss decomposition of probabilistic graphs: Semantics and algorithms. In *SIGMOD*, pages 77–90, 2016.

[27] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *STOC*, pages 731–740, 2002.

[28] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *SIGCOMM*, pages 145–158, 2004.

[29] R. Jin, L. Liu, and C. C. Aggarwal. Discovering highly reliable subgraphs in uncertain graphs. In *KDD*, pages 992–1000, 2011.

[30] R. Jin, L. Liu, B. Ding, and H. Wang. Distance-constraint reachability computation in uncertain graphs. *PVLDB*, 4(9):551–562, 2011.

[31] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. In *ASPLOS*, pages 96–107, 2002.

[32] A. Khan, F. Bonchi, A. Gionis, and F. Gullo. Fast reliability search in uncertain graphs. In *EDBT*, pages 535–546, 2014.

[33] A. Khan and L. Chen. On uncertain graphs modeling and queries. *PVLDB*, 8(12):2042–2043, 2015.

[34] A. Khan, F. Gullo, T. Wohler, and F. Bonchi. Top-k reliable edge colors in uncertain graphs. In *CIKM*, pages 1851–1854, 2015.

[35] D. Kim, W. Wang, N. Sohaee, C. Ma, W. Wu, W. Lee, and D. Du. Minimum data-latency-bound$k$-sink placement problem in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 19(5):1344–1353, Oct 2011.

[36] G. Kollios, M. Potamias, and E. Terzi. Clustering large probabilistic graphs. *IEEE Transactions on Knowledge and Data Engineering*, 25(2):325–336, 2013.

[37] A. Krause and D. Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*, pages 71–104. Cambridge University Press, 2014.

[38] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.

[39] X. Li, M. Wu, C.-K. Kwoh, and S.-K. Ng. Computational approaches for detecting protein complexes from protein interaction networks: a survey. *BMC genomics*, 11(1):S3, 2010.

[40] L. Liu, R. Jin, C. Aggarwal, and Y. Shen. Reliable clustering on uncertain graphs. In *ICDM*, pages 459–468, 2012.

[41] W. Lu, W. Chen, and L. V. Lakshmanan. From competition to complementarity: comparative influence diffusion and maximization. *PVLDB*, 9(2):60–71, 2015.

[42] Z. Lu, X. Sun, and T. L. Porta. Cooperative data offloading in opportunistic mobile networks. In *INFOCOM*, pages 1–9, 2016.

[43] Z. Lu, Y. Wen, and G. Cao. Information diffusion in mobile social networks: The speed perspective. In *INFOCOM*, pages 1932–1940, 2014.

[44] Z. Lu, Y. Wen, W. Zhang, Q. Zheng, and G. Cao. Towards information diffusion in mobile social networks. *IEEE Transactions on Mobile Computing*, 15(5):1292–1304, 2016.

[45] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause. Lazier than lazy greedy. In *AAAI*, pages 1812–1818, 2015.

[46] P. Parchas, F. Gullo, D. Papadias, and F. Bonchi. The pursuit of a good possible world: extracting representative instances of uncertain graphs. In *SIGMOD*, pages 967–978, 2014.

[47] M. Pellegrini, M. Baglioni, and F. Geraci. Protein complex prediction for large protein protein interaction networks with the core&peel method. *BMC bioinformatics*, 17(12):372, 2016.

[48] L. Pelusi, A. Passarella, and M. Conti. Opportunistic networking: Data forwarding in disconnected mobile ad hoc networks. *IEEE Communications*, 44(11):134–141, 2006.

[49] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios. K-nearest neighbors in uncertain graphs. *PVLDB*, 3(1):997–1008, 2010.

[50] S. Pu, J. Wong, B. Turner, E. Cho, and S. J. Wodak. Up-to-date catalogues of yeast protein complexes. *Nucleic acids research*, 37(3):825–831, 2008.

[51] M. Purohit, B. A. Prakash, C. Kang, Y. Zhang, and V. Subrahmanian. Fast influence-based coarsening for large networks. In *KDD*, pages 1296–1305, 2014.

[52] S. Rajasegarar, C. Leckie, and M. Palaniswami. Anomaly detection in wireless sensor networks. *IEEE Wireless Communications*, 15(4):34–40, Aug 2008.

[53] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *STOC*, pages 475–484, 1997.

[54] S. Srihari and H. W. Leong. A survey of computational methods for protein complex prediction from protein interaction networks. *Journal of Bioinformatics and Computational Biology*, 11(2), 2013.

[55] J. Tang, X. Tang, X. Xiao, and J. Yuan. Online processing algorithms for influence maximization. In *SIGMOD*, pages 991–1005, 2018.

[56] J. Tang, X. Tang, and J. Yuan. Influence maximization meets efficiency and effectiveness: A hop-based approach. In *ASONAM*, pages 64–71, 2017.

[57] J. Tang, X. Tang, and J. Yuan. An efficient and effective hop-based approach for inluence maximization in social networks. *Social Network Analysis and Mining*, 8:10, 2018.

[58] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: A martingale approach. In *SIGMOD*, pages 1539–1554, 2015.

[59] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin, 2001.

[60] D. P. Williamson and D. B. Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

[61] B. Xu, H. Lin, and Z. Yang. Ontology integration to identify protein complex in protein interaction networks. *Proteome science*, 9(1):S7, 2011.

[62] Y. Yuan, G. Wang, L. Chen, and H. Wang. Efficient subgraph similarity search on large probabilistic graph databases. *PVLDB*, 5(9):800–811, 2012.

[63] Y. Yuan, G. Wang, H. Wang, and L. Chen. Efficient subgraph search over large uncertain graphs. *PVLDB*, 4(11):876–886, 2011.

[64] F. Zhao and A. K. Tung. Large scale cohesive subgraphs discovery for social network visual analysis. *PVLDB*, 6(2):85–96, 2012.

[65] W. Zheng, L. Zou, X. Lian, J. X. Yu, S. Song, and D. Zhao. How to build templates for rdf question/answering: An uncertain graph similarity join approach. In *SIGMOD*, pages 1809–1824, 2015.

[66] Z. Zou, H. Gao, and J. Li. Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. In *KDD*, pages 633–642, 2010.