Enumeration of Maximal Cliques from an Uncertain Graph

Arko Provo Mukherjee [#1], Pan Xu [*2], Srikanta Tirthapura [#3]

[#] *Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA*
Coover Hall, Ames, IA, USA

[1] arko@iastate.edu
[3] snt@iastate.edu

[*] *Department of Computer Science, University of Maryland*
A.V. Williams Building, College Park, MD, USA

[2] panxu@cs.umd.edu

Abstract—We consider the enumeration of dense substructures (maximal cliques) from an uncertain graph. For parameter $0 < \alpha < 1$, we define the notion of an α-maximal clique in an uncertain graph. We present matching upper and lower bounds on the number of α-maximal cliques possible within a (uncertain) graph. We present an algorithm to enumerate α-maximal cliques whose worst-case runtime is near-optimal, and an experimental evaluation showing the practical utility of the algorithm.

Index Terms—Graph Mining, Uncertain Graph, Maximal Clique, Dense Substructure

1 Introduction

Large datasets often contain information that is uncertain in nature. For example, given people A and B, it may not be possible to definitively assert a relation of the form "A knows B" using available information. Our confidence in such relations are commonly quantified using probability, and we say that the relation exists with a probability of p, for some value p determined from the available information. In this work, we focus on *uncertain graphs*, where our knowledge is represented as a graph, and there is uncertainty in the presence of each edge in the graph. Uncertain graphs have been used extensively in modeling, for example, in communication networks [1, 2, 3], social networks [4, 5, 6, 7, 8, 9], protein interaction networks [10, 11, 12], and regulatory networks in biological systems [13].

Identification of dense substructures within a graph is a fundamental task, with numerous applications in data mining, including in clustering and community detection in social and biological networks [14], the study of the co-expression of genes under stress [15], integrating different types of genome mapping data [16]. Perhaps the most elementary dense substructure in a graph, also probably the most commonly used, is a clique, a completely connected subgraph. We are typically interested in a *maximal clique*, which is a clique that is not contained within any other clique. Enumerating all maximal cliques from a graph is one of the most basic problems in graph mining, and has been applied in many settings, including in finding overlapping communities from social networks [14, 17, 18, 19], finding overlapping multiple protein complexes [20], analysis of email networks [21] and other problems in bioinformatics [22, 23, 24].

While the notion of a dense substructure and methods for enumerating dense substructures are well understood in a deterministic graph, the same is not true in the case of an uncertain graph. This is an important open problem today, given that many datasets increasingly incorporate data that is noisy and uncertain in nature. Uncertainty can result from a lack of data. For example, in constructing a social network from data collected through sensors, some communications between individuals maybe missed, or maybe anonymized [4]. In some cases, relationships themselves are probabilistic in nature; for example, the relation of one person influencing another in a social network [25]. In biological networks such as protein–protein interaction networks, it is known that there are frequent errors in finding interactions and our knowledge is best modeled probabilistically [10].

In this work, we consider the analog of a maximal clique in an uncertain graph. Intuitively, a clique in an uncertain graph is a set of vertices that has a high probability of being a completely connected subgraph. In other words, when we

sample from the uncertain graph, this set is likely to form a (deterministic) clique. Finding such sets of vertices enables us to unearth robust communities within an uncertain graph, for example, a group of proteins such that it is likely that each protein interacts with each other protein. We present a systematic study of the problem of identifying cliques within an uncertain graph. A preliminary version of this work appeared in [26].

1.1 Our Contributions

First, we present a precise definition of a maximal clique in an uncertain graph, leading to the notion of an α-maximal clique, for parameter $0 < \alpha \leq 1$. A set of vertices U in an uncertain graph is an α-maximal clique if U is a clique with probability at least α, and there does not exist a vertex set U' such that $U \subset U'$ and U' is a clique with probability at least α. When $\alpha = 1$, the above definition reduces to the well understood notion of a maximal clique in a deterministic graph.

Number of Maximal Cliques. We first consider a basic question on maximal cliques in an uncertain graph: *how many α-maximal cliques can be present within an uncertain graph?* For deterministic graphs, this question was first considered by Moon and Moser [27] in 1965, who presented matching upper and lower bounds for the largest number of maximal cliques within a graph; on a graph with n vertices, the largest possible number of maximal cliques is $3^{\frac{n}{3}}$[1]. For the case of uncertain graphs, we present the first matching upper and lower bounds for the largest number of α-maximal cliques in a graph on n vertices. We show that for any $0 < \alpha < 1$, the maximum number of α-maximal cliques possible in an uncertain graph is $\binom{n}{\lfloor n/2 \rfloor}$, i.e. there is an uncertain graph on n vertices with $\binom{n}{\lfloor n/2 \rfloor}$ uncertain maximal cliques and no uncertain graph on n vertices can have more than $\binom{n}{\lfloor n/2 \rfloor}$ α-maximal cliques.

Algorithm for Enumerating All Maximal Cliques. We present a novel algorithm, *MULE* (*M*aximal *U*ncertain c*L*ique *E*numeration), for enumerating all α-maximal cliques within an uncertain graph. MULE is based on a depth-first-search of the graph, combined with optimizations for limiting exploration of the search space, and a fast way to check for maximality based on an incremental computation of clique probabilities. We present a theoretical analysis showing that the worst-case runtime of MULE is $O\left(n \cdot 2^n\right)$, where n is the number of vertices. This is nearly the best possible dependence on n, since our analysis of the number of maximal cliques shows that the size of the output can be as much as $O(\sqrt{n} \cdot 2^n)$. Such worst-case behavior occurs only in graphs that are very dense; for typical graphs, we can expect the runtime of MULE to be far better, as we show in our experimental evaluation. We also present an extension of MULE to efficiently enumerate only large maximal cliques.

Note that the worst–case runtime of our algorithm is not the same as an exhaustive search. The cost of checking whether an uncertain clique is maximal or not can be as large as $\Theta(n^2)$. Considering that there are 2^n subsets of vertices of the graph, exhaustive search has a worst-case runtime of $O\left(n^2 \cdot 2^n\right)$, which is worse than our algorithm by a factor of $O(n)$.

Experimental Evaluation. We present an experimental evaluation of MULE using synthetic as well as real-world uncertain graphs. Our evaluation shows that MULE is practical and can enumerate maximal cliques in an uncertain graph with tens of thousands of vertices, more than hundred thousand edges and more than two million α-maximal cliques. Interestingly, the observed runtime of this algorithm is proportional to the size of the output. The real-world graphs included a protein–protein interaction network, and a collaboration network inferred from DBLP.

1.2 Related Work

There has been much recent work on mining from uncertain graphs, including computing shortest paths [28], nearest neighbors [29], clustering [30], enumerating frequent and reliable subgraphs [31, 32, 33, 34, 35, 36], and distance-constrained reachability [37]. The problem of enumerating dense substructures is different from the above. In particular, the problem of finding reliable subgraphs is one of finding subgraphs that are connected with a high probability. However, these individual subgraphs are not required to be dense and may be sparse. In contrast, we are interested in finding subgraphs that are not just connected, but also fully connected with a high probability. The most closely related work to ours is on mining cliques from an uncertain graph by Zou et. al [38]. Our work is different from theirs in significant ways as elaborated below.

- While we focus on enumerating all α-maximal cliques in a graph, they focus on a different problem, that of enumerating the k cliques with the highest probability of existence.
- We present bounds on the number of such cliques that could exist, while by definition, their problem requires them to output no more than k cliques.
- We provide a runtime complexity analysis of our algorithm and show that it is near optimal. No runtime complexity analysis was provided for the algorithm presented in [38].
- We also provide an algorithm to enumerate only large maximal uncertain cliques.

There is substantial prior work on maximal clique enumeration from a deterministic graph. A popular algorithm for maximal clique enumeration problem is the Bron-Kerbosch algorithm [39], also based on depth-first-search. Tomita et al. [40] improved the depth-first-search approach

1. This assumes that 3 divides n. If not, the expressions are slightly different

through a better strategy for pivot selection; their resulting algorithm runs in time $O(3^{\frac{n}{3}})$, which is worst-case optimal, due to the bound on the number of maximal cliques possible [27]. Further work on enumeration of maximal cliques includes [41, 42, 43, 44, 45]. There is work on parallel methods for enumerating maximal cliques and bicliques from a large graph [46, 47].

Our algorithm uses the general structure of search presented in [39, 40]. However, unlike the case of a deterministic maximal clique where it is easy to incrementally maintain the set of vertices that can be added to the clique, for an uncertain graph, this is more complex, since we need to be aware of the change in clique probabilities. Recomputing these can be expensive, and our algorithms reduce this cost through an incremental computation. Our runtime analysis and correctness proof need to take this into account, and do not follow from the analysis in [39] or [40].

Roadmap. We present a problem definition in Section 2, bounds on the number of α-maximal cliques in Section 3, an algorithm to enumerate all α-maximal cliques in Section 4, followed by experimental results in Section 5.

2 Problem Definition

An uncertain graph is a probability distribution over a set of deterministic graphs. We deal with undirected simple graphs, i.e. there are no self-loops or multiple edges. An uncertain graph is a triple $\mathcal{G} = (V, E, p)$, where V is a set of vertices, $E \subseteq V \times V$ is a set of (possible) edges, and $p : E \rightarrow (0, 1]$ is a function that assigns a probability of existence $p(e)$ to each edge $e \in E$. As in prior work on uncertain graphs, we assume that the existence of different edges are mutually independent events.

Let $n = |V|$ and $m = |E|$. Note that $\mathcal{G}$ is a distribution over 2^m deterministic graphs, each of which is a subgraph of the undirected graph (V, E). This set of possible deterministic graphs is called the set of "possible graphs" of the uncertain graph $\mathcal{G}$, and is denoted by $D(\mathcal{G})$. Note that in order to sample from an uncertain graph $\mathcal{G}$, it is sufficient to sample each edge $e \in E$ independently with a probability $p(e)$.

In an uncertain graph $\mathcal{G} = (V, E, p)$, two vertices u and v are said to be adjacent if there exists an edge $\{u, v\}$ in E. Let the neighborhood of vertex u, denoted $\Gamma(u)$, be the set of all vertices that are adjacent to u in $\mathcal{G}$. The next two definitions are standard, and apply not to uncertain graphs, but to deterministic graphs.

Definition 1. *A set of vertices $C \subseteq V$ is a clique in a graph $G = (V, E)$, if every pair of vertices in C is connected by an edge in E.*

Definition 2. *A set of vertices $M \subseteq V$ is a maximal clique in a graph $G = (V, E)$, if (1) M is a clique in G and (2) There is no vertex $v \in V \setminus M$ such that $M \cup \{v\}$ is a clique in G.*

Definition 3. *In an uncertain graph $\mathcal{G}$, for a set of vertices $C \subseteq V$, the clique probability of C, denoted by $clq(C, \mathcal{G})$, is defined as the probability that in a graph sampled from $\mathcal{G}$, C is a clique. For parameter $0 \leq \alpha \leq 1$, C is called an α-clique if $clq(C, \mathcal{G}) \geq \alpha$.*

For any set of vertices $C \subseteq V$, let E_C denote the set of edges $\{e = \{u, v\} | e \in E, u, v \in C \text{ and } u \neq v\}$, i.e. the set of edges connecting vertices in C.

Observation 1. *For any set of vertices $C \subseteq V$ in $\mathcal{G} = (V, E, p)$, such that C is a clique in $G = (V, E)$, $clq(C, \mathcal{G}) = \prod_{e \in E_C} p(e)$.*

Proof. Let G be a graph sampled from $\mathcal{G}$. The set C will be a clique in G iff every edge in E_C is present in G. Since the events of selecting different edges are independent of each other, the observation follows. □

Definition 4. *Given an uncertain graph $\mathcal{G} = (V, E, p)$, and a parameter $0 \leq \alpha \leq 1$, a set $M \subseteq V$ is defined as an α-maximal clique if (1) M is an α-clique in $\mathcal{G}$, and (2) There is no vertex $v \in (V \setminus M)$ such that $M \cup \{v\}$ is an α-clique in $\mathcal{G}$.*

Definition 5. *The* **Maximal Clique Enumeration** *problem in an Uncertain Graph $\mathcal{G}$ is to enumerate all vertex sets $M \subseteq V$ such that M is an α-maximal clique in $\mathcal{G}$.*

The following two observations follow directly from Observation 1.

Observation 2. *For any two vertex sets A, B in $\mathcal{G}$, if $B \subset A$ then, $clq(B, \mathcal{G}) \geq clq(A, \mathcal{G})$.*

Observation 3. *Let C be an α-clique in $\mathcal{G}$. Then for all $e \in E_C$ we have $p(e) \geq \alpha$.*

3 Number of Maximal Cliques

The maximum number of maximal cliques in a deterministic graph on n vertices is known exactly due to a result by Moon and Moser [27]. If $n \mod 3 = 0$, this number is $3^{\frac{n}{3}}$. If $n \mod 3 = 1$, then it is $4 \cdot 3^{\frac{n-4}{3}}$, and if $n \mod 3 = 2$, then it is $2 \cdot 3^{\frac{n-2}{3}}$. The graphs that have the maximum number of maximal cliques are known as Moon-Moser graphs.

For uncertain cliques, no such bound was known so far. In this section, we establish a bound on the maximum number of α-maximal cliques in an uncertain graph. For $0 < \alpha < 1$, let $f(n, \alpha)$ be the maximum number of α-maximal cliques in any uncertain graph with n nodes, without any assumption about the assignments of edge probabilities. The following theorem is the main result of this section.

Theorem 1. *Let $n \geq 2$, and $0 < \alpha < 1$. Then: $f(n, \alpha) = \binom{n}{\lfloor n/2 \rfloor}$*

Proof. We can easily verify that the theorem holds for $n = 2$. for $n \geq 3$, let $g(n) = \binom{n}{\lfloor n/2 \rfloor}$. We show $f(n, \alpha)$ is at

least $g(n)$ in Lemma 1, and then show that $f(n,\alpha)$ is no more than $g(n)$ in Lemma 2. □

Lemma 1. *For any $n \geq 3$, and any $\alpha, 0 < \alpha < 1$, there exists an uncertain graph $\mathcal{G} = (V, E, p)$ with n nodes which has $g(n)$ α-maximal cliques.*

Proof. First, we assume that n is even. Consider $\mathcal{G} = (V, E, p)$, where $E = V \times V$. Let $\kappa = \binom{n/2}{2}$. For each $e \in E$, let $p(e) = q$ where $q^\kappa = \alpha$. We have $0 < q < 1$ since $0 < \alpha < 1$. Let S be an arbitrary subset of V such that $|S| = n/2$. We can verify that S is an α-maximal clique since (1) the probability that S is a clique is $q^\kappa = \alpha$ and (2) for any set $S' \supsetneq S, S' \subseteq V$, the probability that S' is a clique is at most $qq^\kappa = q\alpha < \alpha$. We can also observe that for any subset $S \subseteq V$, S cannot be an α-maximal clique if $|S| < n/2$ or $|S| > n/2$. Thus we conclude that a subset $S \subseteq V$ is an α-maximal clique iff $|S| = n/2$ which implies that the total number of α-maximal cliques in $\mathcal{G}$ is $\binom{n}{n/2}$. A similar proof applies when n is odd. □

Note that our construction in the Lemma above employs the condition that $n \geq 3$ and $0 < \alpha < 1$. When $\alpha = 1$, the upper bound is from the result of Moon and Moser for deterministic graphs, and in this case $f(n,\alpha) = 3^{\frac{n}{3}}$ and is smaller than $g(n)$. Next we present a useful definition required for proving the next Lemma.

Definition 6. *A collection of sets $\mathcal{C}$ is said to be non-redundant if for any pair $S_1, S_2 \in \mathcal{C}$, $S_1 \neq S_2$, we have $S_1 \nsubseteq S_2$ and $S_2 \nsubseteq S_1$.*

Lemma 2. *$g(n)$ is an upper bound on $f(n,\alpha)$.*

Proof. Let $\mathcal{C}^\alpha(\mathcal{G})$ be the collection of all α-maximal cliques in $\mathcal{G}$. Note that by the definition of α-maximal cliques, any α-maximal clique S in $\mathcal{G}$ can not be a proper subset of any other α-maximal clique in $\mathcal{G}$. Thus from Definition 6, for any uncertain graph $\mathcal{G}$, $\mathcal{C}^\alpha(\mathcal{G})$ is a non-redundant collection. Hence, it is clear that the largest number of α-maximal cliques in $\mathcal{G}$ should be upper bounded by the size of a largest non-redundant collection of subsets of V.

Let $\mathcal{C}$ be the collection of all subsets of V. Based on $\mathcal{C}$, we construct such an undirected graph $\widehat{G} = (\mathcal{C}, \widehat{E})$ where for any two nodes $S_1 \in \mathcal{C}, S_2 \in \mathcal{C}$, there is an edge connecting S_1 and S_2 iff $S_1 \subseteq S_2$ or $S_2 \subseteq S_1$. It can be verified that a sub-collection $\mathcal{C}' \subseteq \mathcal{C}$ is a non-redundant iff $\mathcal{C}'$ is an independent set in $\widehat{G}$. In Lemma 3, we show that $g(n)$ is the size of a largest independent set of $\widehat{G}$, which implies that $g(n)$ is an upper bound for the number of α-maximal cliques in $\mathcal{G}$. □

Let $\mathcal{C}^*$ be a largest independent set in $\widehat{G}$. Also, let $\mathcal{C}_k \subseteq \mathcal{C}, 0 \leq k \leq n$ be the collection of subsets of V with the size of k. Observe that for each $0 \leq k \leq n$, $\mathcal{C}_k$ is an independent set of $\widehat{G}$. Also let $L(n)$ and $U(n)$ be respectively the minimum and maximum size of sets in $\mathcal{C}^*$. We can show that $L(n)$ and $U(n)$ can be bounded as shown in Lemma 4 and Lemma 5 respectively.

Lemma 3. *For any $n \geq 3$, $|\mathcal{C}^*| = g(n)$.*

Proof. We first consider the case when n is even. By Lemmas 4 and 5, we know $n/2 \leq L(n) \leq U(n) \leq n/2$. Thus we have $L(n) = U(n) = n/2$ which implies $\mathcal{C}^* = \mathcal{C}^*_{n/2}$. Recall that $\mathcal{C}_k \subseteq \mathcal{C}, 0 \leq k \leq n$ is the collection of subsets of V with the size of k.

We have (1) $\mathcal{C}^* = \mathcal{C}^*_{n/2} \subseteq \mathcal{C}_{n/2}$ and (2) $|\mathcal{C}^*| \geq |\mathcal{C}_{n/2}|$ since $\mathcal{C}^*$ is a largest independent set of $\widehat{G}$. Thus we conclude $\mathcal{C}^* = \mathcal{C}_{n/2}$ which has the size of $\binom{n}{(n/2)} = g(n)$.

We next consider the case when n is odd. From Lemmas 4 and 5, we know $(n-1)/2 \leq L(n) \leq U(n) \leq (n+1)/2$. Thus we have $\mathcal{C}^* = \mathcal{C}^*_{(n-1)/2} \bigcup \mathcal{C}^*_{(n+1)/2}$. For notation convenience, we set $n_1 = (n-1)/2, n_2 = (n+1)/2$. Let $\widehat{G}(\mathcal{C}_{n_1}, \mathcal{C}_{n_2})$ be the subgraph of $\widehat{G}$ induced by $\mathcal{C}_{n_1} \cup \mathcal{C}_{n_2}$. We can view $\widehat{G}(\mathcal{C}_{n_1}, \mathcal{C}_{n_2})$ as a bipartite graph with two disjoint vertex sets $\mathcal{C}_{n_1}$ and $\mathcal{C}_{n_2}$ respectively. Observe that $\mathcal{C}^*_{n_1} \subseteq \mathcal{C}_{n_1}$ and $\mathcal{C}^*_{n_2} \subseteq \mathcal{C}_{n_2}$. Let $\widehat{E}(\mathcal{C}^*_{n_1})$ be the set of edges induced by $\mathcal{C}^*_{n_1}$ in $\widehat{G}(\mathcal{C}_{n_1}, \mathcal{C}_{n_2})$. Since $\mathcal{C}^*$ is an independent set of $\widehat{G}$, none of the edges in $\widehat{E}(\mathcal{C}^*_{n_1})$ will have an end in a node of $\mathcal{C}^*_{n_2}$, i.e, all the edges of $\widehat{E}(\mathcal{C}^*_{n_1})$ should have an end falling in $\mathcal{C}_{n_2} \setminus \mathcal{C}^*_{n_2}$. Note that in $\widehat{G}(\mathcal{C}_{n_1}, \mathcal{C}_{n_2})$, all nodes have a degree of n_2. Thus we have:

$$|\widehat{E}(\mathcal{C}^*_{n_1})| = |\mathcal{C}^*_{n_1}| * n_2 \leq |\mathcal{C}_{n_2} \setminus \mathcal{C}^*_{n_2}| * n_2 = (|\mathcal{C}_{n_2}| - |\mathcal{C}^*_{n_2}|) * n_2$$

from which we obtain $|\mathcal{C}^*| = |\mathcal{C}^*_{n_1}| + |\mathcal{C}^*_{n_2}| \leq |\mathcal{C}_{n_2}| = \binom{n}{n_2}$. Note that $\mathcal{C}_{n_2}$ itself is an independent set of $\widehat{G}$ with size $\binom{n}{n_2}$. Thus we conclude that $|\mathcal{C}^*| = \binom{n}{n_2} = g(n)$. □

Lemma 4. $L(n) \geq \lfloor n/2 \rfloor$

Proof. Let us assume n is an even number. We prove by contradiction as follows. Suppose $L(n) = \ell \leq n/2 - 1$. Let $\mathcal{C}^*_k \subseteq \mathcal{C}^*, L(n) \leq k \leq U(n)$ be the collection of all sets in $\mathcal{C}^*$ which has the size of k, i.e, $\mathcal{C}^*_k = \{S \in \mathcal{C}^* || S| = k\}$. In the following we construct a new collection $\mathcal{C}_{new} \subseteq \mathcal{C}$ which proves to be an independent set in $\widehat{G}$ with the size being strictly larger than $\mathcal{C}^*$. For each $S \in \mathcal{C}^*_\ell$, we add to $\mathcal{C}^*$ all subsets of V which has the form as $S \cup \{i\}$ where $i \in V \setminus S$ and remove S from $\mathcal{C}^*$ meanwhile. Let $\mathcal{C}_{new}$ be the collection obtained after we process the same route for all $S \in \mathcal{C}^*_\ell$. Mathematically, we have: $\mathcal{C}_{new} = \mathcal{C}_1 \bigcup \mathcal{C}_2$ where $\mathcal{C}_1 = \bigcup_{S \in \mathcal{C}^*_\ell} \bigcup_{i \in V \setminus S} \{S \cup \{i\}\}, \mathcal{C}_2 = \mathcal{C}^* \setminus \mathcal{C}^*_\ell$. First we show $\mathcal{C}_{new}$ is an independent set of $\widehat{G}$. Arbitrarily choose two distinct sets, say $S_1 \in \mathcal{C}_{new}, S_2 \in \mathcal{C}_{new}, S_1 \neq S_2$. We check all the possible cases one by one:

- $S_1 \in \mathcal{C}_1, S_2 \in \mathcal{C}_1$. We observe that $|S_1| = |S_2| = \ell + 1$ and $S_1 \neq S_2$. Thus no inclusion relation could exist between S_1 and S_2.

- $S_1 \in \mathcal{C}_2, S_2 \in \mathcal{C}_2$. In this case no inclusion relation can exist between S_1 and S_2 since $\mathcal{C}_2$ is an independent set of $\widehat{G}$.
- $S_1 \in \mathcal{C}_1, S_2 \in \mathcal{C}_2$. Since $\mathcal{C}_\ell^*$ is the collection of sets in $\mathcal{C}^*$ which has the smallest size ℓ, we get that $|S_2| \geq \ell + 1 = |S_1|$. Therefore there is only one possible inclusion relation existing here, that is $S_1 \subset S_2$. Suppose $S_1 = S_1' \cup \{i_1\} \subset S_2$ for some $S_1' \in \mathcal{C}_\ell^*$. Thus we get that $S_1' \subset S_2$ which implies $\mathcal{C}^*$ is not an independent set of $\widehat{G}$. Hence we conclude that no inclusion relation could exist between S_1 and S_2.

Summarizing the analysis above, we get that no inclusion relation could exist between S_1 and S_2 which yields $\mathcal{C}_{new}$ is an independent set of $\widehat{G}$.

Now we prove that $|\mathcal{C}_{new}| > |\mathcal{C}^*|$. Observe that $\mathcal{C}_1$ and $\mathcal{C}_2$ are disjoint from each other; otherwise $\mathcal{C}^*$ is not an independent set. So we have $|\mathcal{C}_{new}| = |\mathcal{C}_1| + |\mathcal{C}_2|$. Note that $|\mathcal{C}^*| = |\mathcal{C}_\ell^*| + |\mathcal{C}_2|$ since $\mathcal{C}^*$ is the union of the two disjoint parts $\mathcal{C}_\ell^*$ and $\mathcal{C}_2$. Therefore $|\mathcal{C}_{new}| > |\mathcal{C}^*|$ is equivalent to $|\mathcal{C}_1| > |\mathcal{C}_\ell^*|$. Let $\widehat{G}(\mathcal{C}_\ell^*, \mathcal{C}_1)$ be the induced subgraph graph of $\widehat{G}$ by $\mathcal{C}_\ell^* \bigcup \mathcal{C}_1$. Note that $\widehat{G}(\mathcal{C}_\ell^*, \mathcal{C}_1)$ can be viewed as a bipartite graph where the two disjoint vertex sets are $\mathcal{C}_\ell^*$ and $\mathcal{C}_1$ respectively. In $\widehat{G}(\mathcal{C}_\ell^*, \mathcal{C}_1)$ we observe that (1) for each node $S_1 \in \mathcal{C}_\ell^*$, its degree $d(S_1) = n - \ell$; (2) for each node $S_2 \in \mathcal{C}_1$, its degree $d(S_2) \leq \ell + 1$. Thus we get that $|\widetilde{E}| = |\mathcal{C}_\ell^*|(n - \ell) \leq |\mathcal{C}_1|(\ell + 1)$. According to our assumption we have $\ell \leq n/2 - 1$. Thus we have $|\mathcal{C}_\ell^*|/|\mathcal{C}_1| \leq (\ell + 1)/(n - \ell) \leq (n/2)/(n/2 + 1) < 1$, yielding $|\mathcal{C}_\ell^*| < |\mathcal{C}_1|$ which is equivalent to $|\mathcal{C}^*| < |\mathcal{C}_{new}|$.

So far we have successfully constructed a new collection $\mathcal{C}_{new} \subseteq \mathcal{C}$ such that (1) it is an independent set of $\widehat{G}$ and (2) $|\mathcal{C}_{new}| > |\mathcal{C}^*|$. That contradicts with the fact that $\mathcal{C}^*$ is a largest independent set of $\widehat{G}$. Thus our assumption $\ell \leq n/2 - 1$ does not hold, which yields $\ell \geq n/2$. For the case when n is odd, we can process essentially the same analysis as above and get $\ell \geq (n - 1)/2$. □

Lemma 5. $U(n) \leq \lceil n/2 \rceil$

Proof. Let us assume n is an even number. Based on $\mathcal{C}^*$, we construct a dual collection $\mathcal{C}_{dual}^*$ as follows: Initialize $\mathcal{C}_{dual}^*$ as an empty collection. For each $S \in \mathcal{C}^*$, we add $V \setminus S$ into $\mathcal{C}_{dual}^*$. Mathematically, we have: $\mathcal{C}_{dual}^* = \bigcup_{S \subseteq \mathcal{C}^*} \{V \setminus S\}$. First we show $\mathcal{C}_{dual}^*$ is an independent set of $\widehat{G}$. Arbitrarily choose two distinct sets, say $V \setminus S_1 \in \mathcal{C}_{dual}^*, V \setminus S_2 \in \mathcal{C}_{dual}^*$, where $S_1 \in \mathcal{C}^*, S_2 \in \mathcal{C}^*, S_1 \neq S_2$. Note that

$$V \setminus S_1 \subset V \setminus S_2 \Leftrightarrow S_1 \supset S_2, V \setminus S_2 \subset V \setminus S_1 \Leftrightarrow S_2 \supset S_1$$

Thus we have that no inclusion relation could exist between $V \setminus S_1$ and $V \setminus S_2$ since no inclusion relation exists between S_1 and S_2 resulting from the fact that $\mathcal{C}^*$ is an independent set of $\widehat{G}$. So we get $\mathcal{C}_{dual}^*$ is an independent set as well.

We can verify that $|\mathcal{C}_{dual}^*| = |\mathcal{C}^*|$. Therefore we can conclude $\mathcal{C}_{dual}^*$ is a largest independent set of $\widehat{G}$. By Lemma 4, we get to know the minimum size of sets in $\mathcal{C}_{dual}^*$ should be at least $n/2$, which yields the maximum size of of sets in $\mathcal{C}^*$ should be at most $n/2$. For the case when n is odd, we can analyze essentially the same as above. □

4 Enumeration Algorithm

In this section, we present **MULE** (Maximal Uncertain cLique Enumeration), an algorithm for enumerating all α-maximal cliques in an uncertain graph $\mathcal{G}$, followed by a proof of correctness and an analysis of the runtime. We assume that $\mathcal{G}$ has no edges e such that $p(e) < \alpha$. If there are any such edges, they can be pruned away without losing any α-maximal cliques, using Observation 3. Let the vertex identifiers in $\mathcal{G}$ be $1, 2, \ldots, n$. For clique C, let $\max(C)$ denote the largest vertex in C. For ease of notation, let $\max(\emptyset) = 0$, and let $clq(\emptyset, \mathcal{G}) = 1$.

Intuition. We first describe a basic approach to enumeration using depth-first-search (DFS) with backtracking. The algorithm starts with a set of vertices C (initialized to an empty set) that is an α-clique and incrementally adds vertices to C, while retaining the property of C being an α-clique, until we can add no more vertices to C. At this point, we have an α-maximal clique. Upon finding a clique that is α-maximal, the algorithm backtracks to explore other possible vertices that can be used to extend C, until all possible search paths have been explored. To avoid exploring the same set C more than once, we add vertices in increasing order of the vertex id. For instance, if C was currently the vertex set $\{1, 3, 4\}$, we do not consider adding vertex 2 to C, since the resulting clique $\{1, 2, 3, 4\}$ will also be reached by the search path by adding vertices $1, 2, 3, 4$ in that order.

MULE improves over the above basic DFS approach in the following ways. First, given a current α-clique C, the set of vertices that can be added to extend C includes only those vertices that are already connected to every vertex within C. Instead of considering every vertex that is greater than $\max(C)$, it is more efficient to track these vertices as the recursive algorithm progresses – this will save the effort of needing to check if a new vertex v can actually be used to extend C. This leads us to incrementally track vertices that can still be used to extend C.

Second, note that not all vertices that extend C into a clique preserve the property of C being an α-clique. In particular, adding a new vertex v to C decreases the clique probability of C by a factor equal to the product of the edge probabilities between v and every vertex in C. So, in considering vertex v for addition to C, we need to compute the factor by which the clique probability will fall. This computation can itself take $\Theta(n)$ time since the size of C can be $\Theta(n)$, and there can be $\Theta(n)$ edges to consider in adding v. A key insight is to reduce this time to $O(1)$ by

incrementally maintaining this factor for each vertex v still under consideration. The recursive subproblem contains, in addition to current clique C, a set I consisting of pairs (u, r) such that $u > \max(C)$, u can extend C into an α-clique, and adding u will multiply the clique probability of C by a factor of r. This set I is incrementally maintained and supplied to further recursive calls.

Finally, there is the cost of checking maximality. Suppose that at a juncture in the algorithm we found that I was empty, i.e. there are no more vertices greater than $\max(C)$ that can extend C into an α-clique. This does not yet mean that C is an α-maximal clique, since it is possible there are vertices less than $\max(C)$, but not in C, which can extend C to an α-maximal clique (note that such an α-maximal clique will be found through a different search path). This means that we have to run another check to see if C is an α-maximal clique. Note that even checking if a set of vertices C is an α-maximal clique can be a $\Theta(n^2)$ operation, since there can be as many as $\Theta(n)$ vertices to be potentially added to C, and $\Theta(n^2)$ edge interactions to be considered. We reduce the time for searching such vertices by maintaining the set X of vertices that can extend C, but will be explored in a different search path. By incrementally maintaining probabilities with vertices in I and X, we can reduce the time for checking maximality of C to $\Theta(n)$.

MULE incorporates the above ideas and is described in Algorithm 1.

Algorithm 1: MULE($\mathcal{G}, \alpha$)

Input: $\mathcal{G}$ is the input uncertain graph,
$\alpha, 0 < \alpha < 1$ is the user provided probability threshold

1 $\hat{I} \leftarrow \emptyset$
2 **forall the** $u \in V$ **do**
3 $\quad \hat{I} \leftarrow \hat{I} \cup \{(u, 1)\}$
4 Enum–Uncertain–MC($\emptyset$, 1 ,$\hat{I}$, $\emptyset$)

4.1 Proof of Correctness

In this section we prove the correctness of MULE.

Theorem 2. *MULE (Algorithm 1) enumerates all α-maximal cliques from an input uncertain graph $\mathcal{G}$.*

Proof. To prove the theorem we need to show the following. First, if C is a clique emitted by Algorithm 1, then C must be an α-maximal clique. Next, if C is an α-maximal clique, then it will be emitted by Algorithm 1. We prove them in Lemmas 8 and 9 respectively. □

Before proving Lemmas 8 and 9, we prove some properties of Algorithm 2.

Lemma 6. *When Algorithm 2 is called with C' in line 10, I' is a set of all tuples $(u'r')$, where $u' \in V$ and $0 < r' \leq 1$, such that $u' > max(C')$, and $clq(C' \cup \{u'\}, \mathcal{G}) = q' \cdot r' \geq \alpha$, i.e. $C' \cup \{u'\}$ is an α-clique in $\mathcal{G}$.*

Algorithm 2: Enum–Uncertain–MC(C, q, I, X)

Input: We assume $\mathcal{G}$ and α are available as immutable global variables
C is the current Uncertain Clique being processed
$q = clq(C, \mathcal{G})$, maintained incrementally
I is a set of all tuples (u, r), such that $u > max(C)$, and $C \cup \{u\}$ is an α-clique in $\mathcal{G}$
X is a set of all tuples (v, s), such that $v \notin C$, $v < max(C)$, and $C \cup \{v\}$ is an α-clique in $\mathcal{G}$

1 **if** $I = \emptyset$ *and* $X = \emptyset$ **then**
2 $\quad$ Output C as α-maximal clique
3 $\quad$ **return**
4 **forall the** $(u, r) \in I$ *considered in lexicographical ordering over* u **do**
5 $\quad C' \leftarrow C \cup \{u\}$ `// Lemma 6`
6 $\quad m = max(C') = u$
7 $\quad q' \leftarrow q \cdot r$ // $clq(C \cup \{v\}, \mathcal{G})$
8 $\quad I' \leftarrow GenerateI(C', q', I)$
9 $\quad X' \leftarrow GenerateX(C', q', X)$
10 $\quad$ Enum–Uncertain–MC(C', q', I', X')
11 $\quad X \leftarrow X \cup \{(u, r)\}$

Algorithm 3: GenerateI(C', q', I)

Input: We assume $\mathcal{G}$ and α are available as immutable global variables

1 $m \leftarrow max(C')$, $I' \leftarrow \emptyset$, $S \leftarrow \emptyset$
2 **forall the** $(u, r) \in I$ **do**
3 $\quad S \leftarrow S \cup \{u\}$
4 $S \leftarrow S \cap \{\Gamma(m)\}$
5 **forall the** $(u, r) \in I$ **do**
6 $\quad$ **if** $u > m$ *and* $u \in S$ **then**
7 $\quad\quad clq(C' \cup \{u\}, \mathcal{G}) \leftarrow q' \cdot r \cdot p(\{u, m\})$
8 $\quad\quad$ **if** $(clq(C' \cup \{u\}, \mathcal{G})) \geq \alpha$ **then**
9 $\quad\quad\quad u' \leftarrow u$
10 $\quad\quad\quad r' \leftarrow r \cdot p(\{u, m\})$
11 $\quad\quad\quad I' \leftarrow I' \cup \{(u', r')\}$
12 **return** I'

Proof. Let $u' \in V$ be a vertex such that (1) $u' > \max(C')$, and (2) $C' \cup \{u'\}$ is an α-clique in $\mathcal{G}$. We need to show that $(u', r') \in I'$ such that $clq(C' \cup \{u'\}, \mathcal{G}) = q' \cdot r'$.

Let C' be a clique being called by Enum–Uncertain–MC with I'. Note that each call of the method adds one vertex $u \in I$ to the current clique C such $u > \max(C)$. Since the vertices are added in the lexicographical ordering, there is an unique sequence of calls to the method Enum–Uncertain–MC such that we reach a point in execution of Algorithm 2 where Enum–Uncertain–MC is called with C'. We call this sequence of calls as Call-0, Call-1, ..., Call-$|C'|$. Also, let C_i be the clique used by method Enum–Uncertain–MC during Call-i.

Algorithm 4: GenerateX(C', q', X)

Input: We assume $\mathcal{G}$ and α are available as immutable global variables

1 $m \leftarrow max(C')$, $X' \leftarrow \emptyset$, $S \leftarrow \emptyset$
2 **forall the** $(v, s) \in I$ **do**
3 $S \leftarrow S \cup \{v\}$
4 $S \leftarrow S \cap \{\Gamma(m)\}$
5 **forall the** $(v, s) \in X$ **do**
6 **if** $v \in S$ **then**
7 $clq(C' \cup \{v\}, \mathcal{G}) \leftarrow q' \cdot s \cdot p(\{v, m\})$
8 **if** $(clq(C' \cup \{v\}, \mathcal{G}) \geq \alpha$ **then**
9 $v' \leftarrow v$
10 $s' \leftarrow s \cdot p(\{v, m\})$
11 $X' \leftarrow X' \cup \{(v', s')\}$
12 **return** X'

We prove by induction. First consider the base case. For that consider the first call made to Algorithm 2, i.e. Call-0. We know that C is initialized as $\emptyset$. During the first call made, all vertices in V satisfy conditions (1) and (2). This is because, first $max(\emptyset) = 0$. Second any single vertex can be considered as a clique with probability 1. $\hat{I}$ is initialized such that all r in $\hat{I}$ are $1 \geq \alpha$. Thus for all u such that $(u, r) \in \hat{I}$, $u > \max(C)$. This proves the base case.

For the inductive step, consider a recursive call to the method Call-i which calls Call-$(i+1)$. For every case expect initialization, I' is generated from I by line 8 of Algorithm 2 which in turn calls Algorithm 3. In Algorithm 3, only vertices in I that are greater than C' are added to I'. Thus all vertices in I that satisfy (1) are added to I'. Next every vertex in I is connected to C. We need to show that all vertices in I' are connected to C'. In line 4 of Algorithm 3, we prune out any vertex in I that is not connected to $m = \max(C')$. Assume that u' extends C such that $clq(C \cup \{u'\}, \mathcal{G}) = r$. Now let $c = \{C' \setminus C\}$. Note that c is a single vertex. Also, assume $u' > c$. From line 4, we know that $q' \cdot r' \geq \alpha$ Also from line 6 of Algorithm 3, $r' = r \cdot p(\{c, u'\})$. Now $clq(C' \cup \{u'\}, \mathcal{G}) = q' \cdot r \cdot p(\{c, u'\}) = q' \cdot r'$, Now in line 8 of Algorithm 3 we add u' to I' only if $r' \geq \alpha$ thus proving the inductive step. □

The following observation follows from Lemma 6.

Observation 4. *The input C to Algorithm 2 is an α-clique.*

Lemma 7. *When Algorithm 2 is called with C' in line 10, X' is a set of all tuples (v', s'), where $v' \in V$ and $0 < s' \leq 1$, such that, $\forall (v', s') \in X'$, we have $v' \notin C'$, $v' < \max(C')$, and $(clq(C' \cup \{v'\}, \mathcal{G}) = q' \cdot s') \geq \alpha$, i.e. $C' \cup \{v'\}$ is an α-clique in $\mathcal{G}$.*

Proof. Let $m = \max(C')$ and $C = C' \setminus \{m\}$. Since Algorithm 2 was called with C', it must have been called with C. This is because the working clique is always extended by adding vertices from I, and from Lemma 6, I only contains vertices that are greater than the maximum vertex in C. Let X be the corresponding set of tuples used when the call was made to Enum–Uncertain–MC with C. Let $u > \max(C)$ be a vertex such that $clq(C' \cup \{u\}, \mathcal{G}) \geq \alpha$ and $u < m$. Note that $u \notin C'$, $u < \max(C')$, and $C' \cup \{u\}$ is an α-clique in $\mathcal{G}$. This means u satisfies all conditions for $u \in X'$. We need to show that when Enum–Uncertain–MC is called with C', the generated X' which is passed in Enum–Uncertain–MC contains u.

Firstly, note that since $C' \cup \{u\}$ is α-clique in $\mathcal{G}$, we have $clq(C \cup \{u\}, \mathcal{G}) \geq \alpha$ (from Observation 2). Since $u > \max(C)$ and $clq(C \cup \{u\}, \mathcal{G}) \geq \alpha$, from Lemma 6, u will be used in line 4 to call Enum–Uncertain–MC using $C \cup \{u\}$. Once this call is returned, u is added to X in line 11. Note that since the loop at line 4 add vertices in lexicographical order, m will be added to C after u. Thus u will be in X, when m is used to extend C. Next we show that if $u \in X$, after execution of line 9, $u \in X'$. We prove this as follows. Note that Algorithm 4 is used to generate X' from X. Note that X' is generated by Algorithm 4 by selectively adding vertices from X. A vertex is added to X' from X, only if $C' \cup \{u\}$ is α-clique in $\mathcal{G}$. From our initial assumptions, we know that u satisfies this condition and is hence added to X' and passed on to Enum–Uncertain–MC when it is called with C'.

Now let us consider v, such that v does not satisfy all the conditions for $v \in X'$. We need to show that $v \notin X'$. There are two cases. First, when $v \notin X$. This case is trivial as X' is constructed from X and hence if $v \notin X$, $v \notin X'$. For the second case, when $v \in X$, we need to show that v will not be added to X' in line 9 of Algorithm 2. Note that since $v \in X$, we know $v \notin C'$ and $v < \max(C')$. Thus, it must be that $C \cup \{m, v\}$ is not an α-clique in $\mathcal{G}$. Algorithm 4 will add v to X' only if $C \cup \{m, v\}$ is α-clique in $\mathcal{G}$. But from our previous discussion, we know that this condition doesn't hold. Hence, v will not be added to X'. Thus only vertices that satisfy all three conditions are in X'. □

Lemma 8. *Let C be a clique emitted by Algorithm 2. Then C is an α-maximal clique.*

Proof. Algorithm 2 emits C in Line 2. From Observation 4, we know that C is an α-clique. We need to show that C is α-maximal. We use proof by contradiction. Suppose C is non-maximal. This means that there exists a vertex $u \in V$, such that $C \cup \{u\}$ is an α-clique. We know that $I = \emptyset$ when C is emitted. From Lemma 6, we know that there exists no vertex $u \in V$ such that $u > max(C)$ that can extend C. Again, we know that $X = \emptyset$ when C is emitted. Thus from Lemma 7, we know that there exists no vertex $v \in V$ such that $v < \max(C)$ that can extend C. This is a contradiction and hence C is an α-maximal clique. □

Lemma 9. *Let C be an α-maximal clique in $\mathcal{G}$. Then C is emitted by Algorithm 2.*

Proof. We first show that a call to method Enum–Uncertain–MC with α-clique C enumerates all α-maximal cliques C' in $\mathcal{G}$, such that for all $c \in \{C' \setminus C\}$, $c > \max(C)$.

Without loss of generality, consider a α-maximal clique C' in $\mathcal{G}$ such that $\forall c \in \{C' \setminus C\}$, $c > \max(C)$. Note that C' will be emitted as an α-maximal clique by the method Enum–Uncertain–MC when called with C, if the following holds: (1) A call to method Enum–Uncertain–MC is made with C', (2) When this call is made, $I' = \emptyset$, and $X' = \emptyset$. Since C' is α-maximal clique in $\mathcal{G}$, the second point follows from Lemmas 6 and 7. Thus we need to show that a call to Enum–Uncertain–MC is made with C'.

We prove this by induction. Let $\hat{C} = \{C' \setminus C\}$. Let c_i represent the ith element in $\hat{C}$ in lexicographical order. Also let $C_i = C \cup \{c_1, c_2, \ldots, c_i\}$. For the base case, we show that if a call to Enum–Uncertain–MC is made with C, a call will be made with $C_1 = C \cup \{c_1\}$. This is because, line 4 of the method loops over every vertex $u \in I$ thus implying $u > \max(C)$ and $clq(C \cup \{u\}, \mathcal{G}) \geq \alpha$. Since C' is an α-maximal clique, c_1 will satisfy both these conditions and hence a call to Enum–Uncertain–MC is made with $C \cup \{c_1\}$. Now for the inductive step we show that if a call is made with clique C_i, then this call will in turn call the method with clique C_{i+1}. Again, c_{i+1} is greater than $\max(C_i)$ and $clq(C_i \cup \{c_{i+1}\}, \mathcal{G}) \geq \alpha$. Thus $c_{i+1} \in I$ when the call is made to Enum–Uncertain–MC with C_i. Hence using the previous argument, in line 4, c_{i+1} will be used as a vertex in the loop which would in turn make a call to Enum–Uncertain–MC with C_{i+1}.

Now without any loss of generality, consider an α-maximal clique in $\mathcal{G}$. We know that $C \supset \emptyset$. Thus the proof follows. □

4.2 Runtime Complexity

Theorem 3. *The runtime of MULE (Algorithm 1) on an input graph of n vertices is $O(n \cdot 2^n)$.*

Proof. MULE initializes variables and calls to Algorithm 2, hence we analyze the runtime of Algorithm 2. An execution of the recursive Algorithm 2 can be viewed as a search tree as follows. Each call to Enum–Uncertain–MC is a node of this search tree. The first call to the method is the root node. A node in this search tree is either an internal node that makes one or more recursive calls, or a leaf node that does not make further recursive calls. To analyze the runtime of Algorithm 2, we consider the time spent at internal nodes as well as leaf nodes.

The runtime at each leaf node is $O(1)$. For a leaf node, the parameter $I = \emptyset$, and there are no further recursive calls. This implies that either C is α-maximal ($X = \emptyset$) and is emitted in line 2 or it is non-maximal ($X \neq \emptyset$) but cannot be extended by the loop in line 4 as $I = \emptyset$. Checking the sizes of I and X takes constant time.

We next consider the time taken at each internal node. Instead of adding up the times at different internal nodes, we equivalently add up the cost of the different edges in the search tree. At each internal node, the cost of making a recursive call can be analyzed as follows. Line 5 takes $O(n)$ time as we add all vertices in C to C' and also u. Line 7 takes constant time. Lines 8 and 9 take $O(n)$ time (Lemmas 10 and 11 respectively). Note that lines 5 to 9 can get executed only once in between the two calls. Thus total runtime for each edge of the search tree is $O(n)$.

Note that the total number of calls made to the method method Enum–Uncertain–MC is no more than the possible number of unique subsets of V, which is $O(2^n)$. We see that for internal nodes, time complexity is $O(n)$ and for leaf nodes it is $O(1)$. Hence the time complexity of Algorithm 2 is $O(n \cdot 2^n)$. □

Thus now we need to prove that lines 8 and 9 take $O(n)$ time. This implies that time complexity of Algorithms 3 and 4 is $O(n)$. We prove the same in Lemmas 10 and 11 respectively.

Lemma 10. *The runtime of Algorithm 3 is $O(n)$.*

Proof. First note that lines 1-6 takes $O(n)$ time. This is because $|I| = O(n)$, and hence the loop at line 4 of Algorithm 3 can take $O(n)$ time. Further the set intersection at line 6 also takes $O(n)$ time. We need to show that the for loop in line 7 is $O(n)$, that is each iteration of the loop takes $O(1)$ time. Assume that it takes constant time to find out the probability of an edge. This is a valid assumption, as the edge probabilities can be stored as a HashMap and hence for an edge e, in constant time we can find out $p(e)$. With this assumption, it is easy to show that lines 8-13 takes constant time. This is because, they are either constant number of multiplications, or adding one element to a set. Thus total time complexity is $O(n)$. □

Lemma 11. *The runtime of Algorithm 4 is $O(n)$.*

We omit the proof of the above lemma since it is similar to the proof of Lemma 10.

Observation 5. *The worst-case runtime of any algorithm that can output all maximal cliques of an uncertain graph on n vertices is $\Omega(\sqrt{n} \cdot 2^n)$.*

Proof. From Theorem 1, we know that the number of maximal uncertain cliques can be as much as $\binom{n}{\lfloor n/2 \rfloor} = \Theta\left(\frac{2^n}{\sqrt{n}}\right)$ (using Stirling's Approximation). Since the size of each uncertain clique can be $\Theta(n)$, the total output size can be $\Omega(\sqrt{n} \cdot 2^n)$, which is a lower bound on the runtime of any algorithm. □

Lemma 12. *The worst-case runtime of MULE on an n vertex graph is within a $O(\sqrt{n})$ factor of the runtime of an optimal algorithm for Maximal Clique Enumeration on an uncertain graph.*

Proof. The proof follows from Theorem 3 and Observation 5. □

4.3 Enumerating Only Large Maximal Cliques

For a typical input graph, many maximal cliques are small, and may not be interesting to the user. Hence it is helpful to have an algorithm that can enumerate only large maximal cliques efficiently, rather than enumerate all maximal cliques. We now describe an algorithm that enumerates every α-maximal clique with more than t vertices, where t is an user provided parameter.

As a first step, we prune the input uncertain graph $\mathcal{G} = (V, E, p)$ by employing techniques described by Modani and Dey [42]. We apply the "Shared Neighborhood Filtering" where edges are recursively checked and removed as follows. First drop all edges $\{u, v\} \in E$, such that $|\Gamma(u) \cap \Gamma(v)| < (t-2)$. Next drop every vertex $v \in V$, that doesn't satisfy the following condition. For vertex $v \in V$, there must exist at least $(t-1)$ vertices in $\Gamma(v)$, such that for $u \in \Gamma(v)$, $|\Gamma(u) \cap \Gamma(v)| < (t-2)$. Let $\mathcal{G}'$ denote the graph resulting from $\mathcal{G}$ after the pruning step.

Algorithm 5 runs on the pruned uncertain graph $\mathcal{G}'$ to enumerate only large maximal cliques. The recursive method in Algorithm 6 differs from Algorithm 2 as follows. Before each recursive call to method Enum–Uncertain–MC-Large (Algorithm 6), the algorithm checks if the sum of the sizes of the current working clique C' and the candidate vertex set I' are greater than the size threshold t. If not, the recursive method is not called. This optimization leads to a substantial pruning of the search space and hence a reduction in runtime.

Algorithm 5: LARGE–MULE($\mathcal{G}, \alpha, t$)

Input: $\mathcal{G}'$ is the input uncertain graph post pruning
$\alpha, 0 < \alpha < 1$ is the user provided probability threshold
$t, t \geq 2$ is the user provided size threshold

1 $\hat{I} \leftarrow \emptyset$
2 **forall the** $u \in V$ **do**
3 $\quad \hat{I} \leftarrow \hat{I} \cup \{(u, 1)\}$
4 Enum–Uncertain–MC–Large($\emptyset$, 1 ,$\hat{I}$, $\emptyset$,t)

Lemma 13. *Given an input graph $\mathcal{G}$, LARGE–MULE (Algorithm 5) enumerates every α-maximal clique with more than t vertices.*

Proof. First we prove that no maximal clique of size less than t is enumerated by Algorithm 6. Consider an α-maximal clique C_1 in $\mathcal{G}$ with less than t vertices. Also let $m_1 = \max(C_1)$ and $C_1' = C_1 \setminus \{m_1\}$. Note that if C_1 is emitted by Algorithm 6, then a call must be made to Enum–Uncertain–MC–Large with C_1. Since the Algorithm adds vertices in lexicographical ordering, this implies that a call must be made to Enum–Uncertain–MC–Large with C_1' before the call is made with C_1. In the worst case, let us consider that the search tree reaches the execution point where Enum–Uncertain–MC–Large is called with C_1'. Consider the execution of the algorithm where m_1 is added to $C = C_1'$ to form $C' = C_1$. Since C_1 is an α-maximal clique, I' will become NULL which implies $|I'| = 0$. We know that $|C_1| < t$. Thus $|C_1 + I'|$ will also be less than t and the If condition (line 8) will succeed. This will result in the execution of the continue statement. Thus Enum–Uncertain–MC-Large will not be called with C_1 implying that C_1 is not enumerated.

Next we show that any maximal clique of size at least t is enumerated by Algorithm 6. Consider an α-maximal clique C_2 in $\mathcal{G}$ of size at least t. We note that the "If" condition in line 8 is never satisfied in the search path ending with C_2 and hence a call is made to the method with Enum-Uncertain-MC-Large with C_2. This is easy to see as whenever a call is made to Enum-Uncertain-MC-Large with any $C \subseteq C_2$, since C_2 is large, we always have $|C| + |I| \geq t$. □

Algorithm 6: Enum–Uncertain–MC–Large(C, q, I, X, t)

Input: We assume $\mathcal{G}$ and α are available as immutable global variables
C is the current Uncertain Clique being processed
$q = clq(C, \mathcal{G})$, maintained incrementally
I is a set of all tuples (u, r), such that $u > max(C)$, and $C \cup \{u\}$ is an α-clique in $\mathcal{G}$
X is a set of all tuples (v, s), such that $v \notin C$, $v < max(C)$, and $C \cup \{v\}$ is an α-clique in $\mathcal{G}$
t is the user provided size threshold

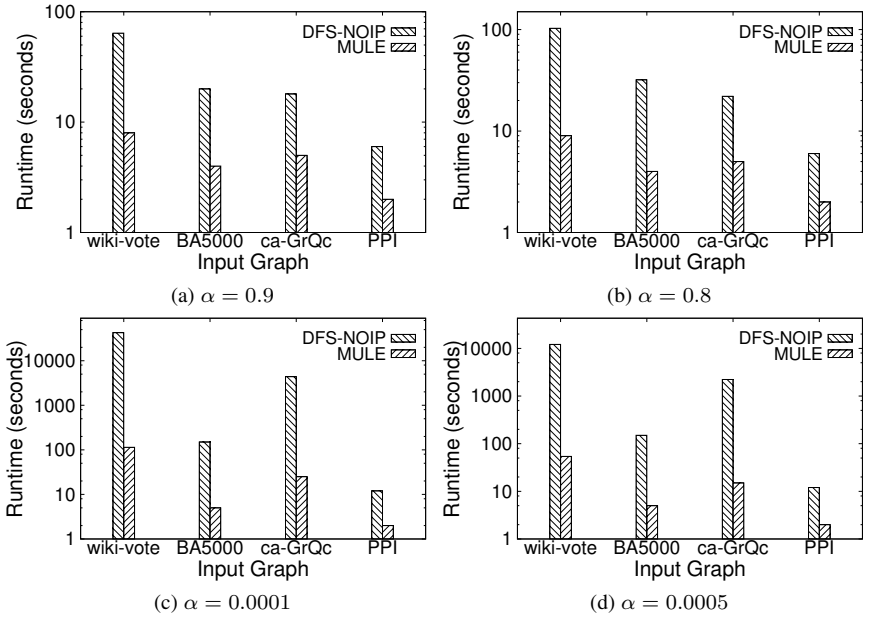
1 **if** $I = \emptyset$ *and* $X = \emptyset$ **then**
2 $\quad$ Output C as α-maximal clique
3 $\quad$ **return**
4 **forall the** $u, r \in I$ *considered in lexicographical ordering over* u **do**
5 $\quad C' \leftarrow C \cup \{u\}$ `// Lemma 6`
6 $\quad m = max(C') = u$
7 $\quad q' \leftarrow q \cdot r$ `//` $clq(C \cup \{v\}, \mathcal{G})$
8 $\quad I' \leftarrow GenerateI(C', q', I)$
9 $\quad$ **if** $|C'| + |I'| < t$ **then**
10 $\quad\quad$ continue
11 $\quad X' \leftarrow GenerateX(C', q', X)$
12 $\quad$ Enum–Uncertain–MC–Large(C', q', I', X', t)
13 $\quad X \leftarrow X \cup \{(u, r)\}$

5 Experimental Results

We report the results of an experimental evaluation of our algorithm. We implemented the algorithm using Java and ran all experiments on a system with a 3.19 GHz Intel(R) Core(TM) i5 processor and 4 GB of RAM, with heap space configured at 1.5GB.

TABLE 1: Input Graphs

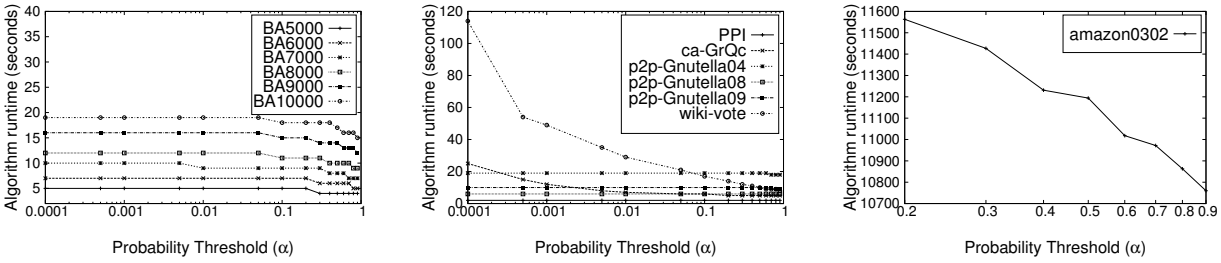
Input Graph	Category	Description	# Vertices	# Edges
Fruit-Fly	Protein Protein Interaction network	PPI for Fruit Fly from STRING Database	3751	3692
DBLP10	Social network	Collaboration network from DBLP	684911	2284991
amazon–0302	Product co-purchasing network	March 2003 Amazon co-purchase network	262111	1234877
p2p-Gnutella08	Internet peer-to-peer networks	Gnutella network August 8 2002	6301	20777
p2p-Gnutella04	Internet peer-to-peer networks	Gnutella network August 4 2003	10879	39994
p2p-Gnutella09	Internet peer-to-peer networks	Gnutella network August 9 2003	8114	26013
ca-GrQc	Collaboration networks	Arxiv General Relativity	5242	28980
wiki-vote	Social networks	wikipedia who-votes-whom network	7118	103689
BA5000	Barabási−Albert random graphs	Random graph with 5K vertices	5000	50032
BA6000	Barabási−Albert random graphs	Random graph with 6K vertices	6000	60129
BA7000	Barabási−Albert random graphs	Random graph with 7K vertices	7000	70204
BA8000	Barabási−Albert random graphs	Random graph with 8K vertices	8000	80185
BA9000	Barabási−Albert random graphs	Random graph with 9K vertices	9000	90418
BA10000	Barabási−Albert random graphs	Random graph with 10K vertices	10000	99194



(a) $\alpha = 0.9$ (b) $\alpha = 0.8$

(c) $\alpha = 0.0001$ (d) $\alpha = 0.0005$

Fig. 1: Comparison of Simple and Optimized Depth First Search approaches. The y–axis is in log–scale.



(a) Random Graphs (b) Semi–synthetic and Real Graphs (c) amazon0302

Fig. 2: Runtime vs Probability threshold (α). The x–axis is in log–scale

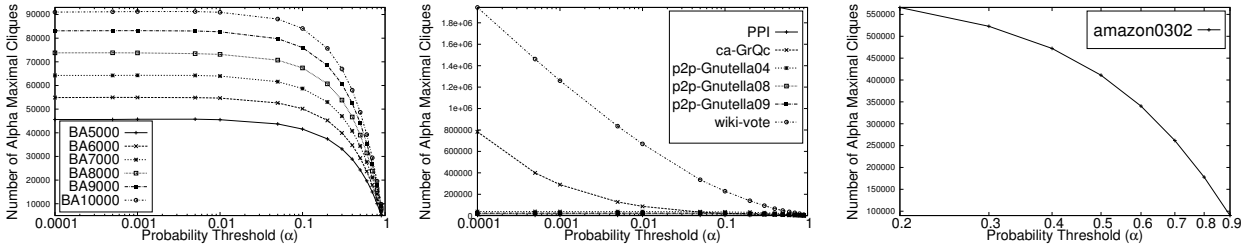
Input Data: Details of the input graphs that we used are shown in Table 1.

The first set of graphs consists of real world uncertain graphs, also used in [32] and [36]. These include a protein-protein interaction (PPI) network of a Fruit Fly obtained by integrating data from the BioGRID [2] database with that form the STRING [3] database, and the DBLP [4] dataset from authors of [36], which is an uncertain network predicting future co-authorship. The PPI network is an uncertain graph where each vertex represents a protein, and two vertices are connected by an edge with a probability representing the

2. http://thebiogrid.org/

3. http://string-db.org/

4. http://dblp.uni-trier.de/



(a) Random Graphs (b) Semi–synthetic and Real Graphs (c) amazon0302

Fig. 3: No of α-maximal cliques vs Probability threshold (α). The x–axis is in log–scale

likelihood of interaction between the the two proteins. The DBLP network represents co-authorship in academic articles. Each vertex in this network represents an author. Two vertices are connected by an edge with a probability that depends on the "strength" of their co-authorship, computed as $1-e^{-c/10}$, where c is the number of papers co–authored.

The second set of graphs was obtained from the Stanford Large Network Collection [48], and includes graphs representing Internet p2p networks, collaboration networks, and an online social network. The amazon0302 network is a product co-purchasing network where each node is a product and two nodes (products) are connected by an edge if they are frequently co-purchased. The p2p-Gnutella graphs represent peer to peer file sharing networks, where each vertex represents a computer and an edge represents an application level communication between computers. The p2p-Gnutella04, p2p-Gnutella08 and p2p-Gnutella09 graphs represent communications occurring on 4th, 8th and 9th of August, 2002 respectively. The ca-GrQc graph represents a collaboration network among scientists working on General Relativity and Quantum Cosmology. Each vertex in the graph is a scientist and two vertices are connected by an edge if the corresponding scientists have co-authored a paper. Finally the wiki-vote graph represents voting that occurs while selecting a new wikipedia administrator. Each vertex is either a wikipedia admin or wikipedia user and an edge represents a vote that an admin/user casts in favor of a candidate. For each graph in the second set, an uncertain graph was created by assigning edge probabilities uniformly at random. Hence these can be considered as semi–synthetic uncertain graphs.

The third set of input graphs was synthetically generated using the Barabási−Albert (BA) model for random graphs [49]. Then the edges were assigned probabilities uniformly at random from $[0, 1]$.

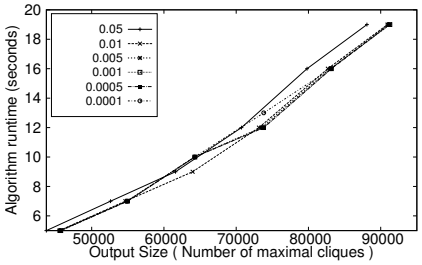
Comparison with other approaches. We compare our algorithm with another algorithm based on depth-first-search, which we call DFS-NOIP (DFS with NO Incremental Probability Computation). Similar to MULE, this algorithm performs a depth first search to enumerate all α–maximal cliques but unlike MULE, it does not compute the probabilities incrementally. The DFS-NOIP algorithm is described in supplementary materials, due to space constraints.

Figure 1 shows the runtime of MULE and of DFS–NOIP on different input graphs. The results show that MULE is much faster than DFS–NOIP. For instance, for the wiki–vote graph with $\alpha = 0.9$ DFS–NOIP took 64 seconds while MULE took only 8 seconds. The relative performance results hold true over a wide range of input graphs and values of α, including synthetic and real-world graphs, and small and large values of α. For $\alpha = 0.0001$, MULE took only 25 secs on ca-GrQc, while DFS–NOIP took over 4400 secs. On the wiki–vote input graph with probability threshold 0.9, MULE took 8 seconds while DFS–NOIP took 64 seconds. For the same graph, with probability threshold 0.0001, MULE took 114 seconds, while DFS–NOIP took more than 11 hours.

Dependence on α. We measured the runtime of enumeration as well as the output size (the number of α-maximal cliques that were output) as a function of α. The dependence of the runtime on α is shown in Figure 2, and the number of cliques as a function of α is shown in Figure 3. We note that as α increases, the number of maximal cliques, and the time of enumeration both drop sharply. The decrease in runtime is because with a larger value of α, the size of the output decreases and the algorithm is able to prune search paths aggressively early in the enumeration.

We note here that the number of α-maximal cliques does not necessarily decrease as α increases. Sometimes it is possible that the number of α-maximal cliques increases with α. For instance, a large maximal clique whose probability is above the threshold for a small value of α may not pass the threshold for a higher value of α, and may split into many smaller maximal cliques, thus leading to an increase in the number of maximal cliques. However, such instances seem to be rare and are not visible in the plots.

Dependence on Output Size Figure 4 shows the runtime as a function of the number of α-maximal cliques enumerated, for randomly generated graphs. It can be seen that the runtime of the algorithm increases with the number of maximal cliques in the output, and also that the runtime scales well with the output size. This comparison was not done for real world or semi–synthetic graphs as these graphs have different structural properties, hence different sizes of maximal cliques and thus there is no meaningful way to interpret the results.



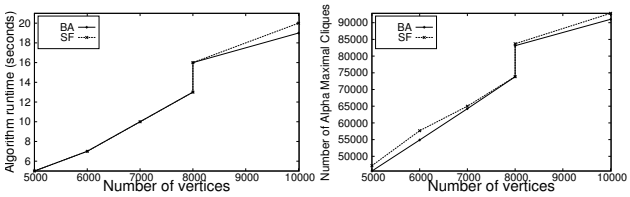
(a) Random Graphs

Fig. 4: Runtime vs Output size

Enumerating Large Maximal Cliques. Figures 5 and 6 show the runtime of LARGE–MULE (Algorithm 5) and the output size respectively as a function of t, the minimum threshold for the size of the α-maximal clique. As t increases, both the runtime and output size decrease substantially. For instance, MULE takes 76797 seconds to enumerate all uncertain maximal cliques from the DBLP dataset, for probability threshold 0.9. However, LARGE–MULE takes only 32 seconds when $t = 3$. Similarly, for input graph ca-GrQc and $\alpha = 0.0001$, MULE takes 125 seconds, while LARGE–MULE takes 10 seconds when $t = 6$ and 6 seconds when $t = 7$.

Dependence on Number of Uncertain Edges. Figure 7 shows the change in runtime as a function of the graph uncertainty, for the ca–GrQc and DBLP10 networks. We consider the same underlying network with difference in the number of uncertain edges. We use the following three cases: 1) when all edges are uncertain, 2) when two–thirds of all the edges are uncertain, and 3) when one–third of all the edges are uncertain.

We can see that as the number of uncertain edges in the graph increases, the runtime decreases. Considering that there can be many more uncertain maximal cliques than just maximal cliques, Figure 7 shows that our algorithm employs effective pruning techniques that help us to quickly identify all maximal uncertain cliques. For example, for the graph ca-GrQc and $\alpha = 0.5$, MULE took 6 and 7 seconds for all uncertain and two–third uncertain edges respectively. However, for one–third uncertain edges, it took 124 seconds. Again for the graph DBLP10 and $\alpha = 0.9$, MULE (with size threshold 4), could find all maximal cliques in 6 seconds with all uncertain edges. But with only two–third uncertain edges, the same graph took over 42 minutes with one–third uncertain edges, we could not process the graph within 4 hours. Thus, the graph processing time required by our method reduces as the uncertainty of the graph increases. Note that there are two ways in which uncertain cliques can be handled – first by using deterministic MCE and then finding embedded uncertain cliques, and second, by directly incorporating uncertainty in pruning, as we do. From Figure 7 we can see that for multiple values of α, as the number of uncertain edges increase, runtime of MULE decreases. This implies that when we directly model uncertainty and use our Algorithm, we can find structures much faster than finding all maximal cliques followed by pruning on basis of probability to find maximal uncertain cliques.



(a) Runtime ($\alpha = 0.0001$) (b) Output Size ($\alpha = 0.0001$)

Fig. 8: BA vs Two–Level graphs

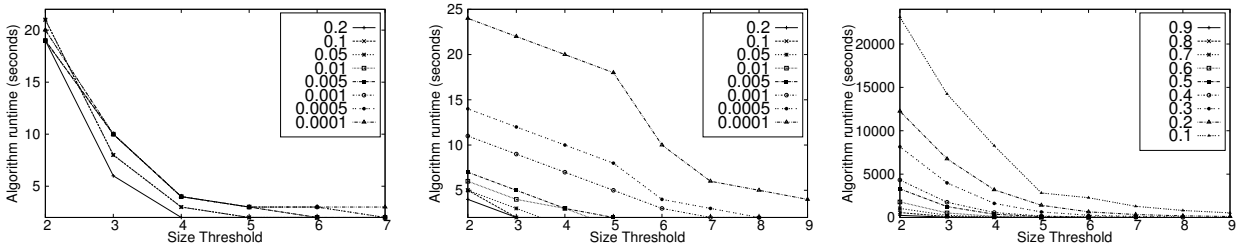
Scale-Free Graph Models. In order to see the performance on different types of scale-free graph models, in addition to the random graphs generated using the Barabási−Albert (BA) model, which is a type of a scale free (SF) model, we constructed random graphs using the two–level scale free model [50], to compare against the graphs generated using the BA model. For each BA input graph that we used, we generated a SF graph with the same number of vertices as the BA graph, and compared the runtime of our algorithm and the output size. Due to lack of space, we present only a subset of results in Figure 8. We observe from the Figure that there is little difference in processing runtime or output size, between the BA graphs and the two-level SF graphs. Both BA as well as two-level SF graphs, took approximately the same amount of time to be processed and had very similar number of maximal uncertain cliques. We observed the same behavior for different values of α that we tried.

6 Conclusion

We present a systematic study of the enumeration of maximal cliques from an uncertain graph, starting from a precise definition of the notion of an α-maximal clique, followed by a proof showing that the maximum number of α-maximal cliques in a graph on n vertices is exactly $\binom{n}{\lfloor n/2 \rfloor}$, for $0 < \alpha < 1$. We present a novel algorithm, MULE, for enumerating the set of all α-maximal cliques from a graph, and an analysis showing that the worst-case runtime of this algorithm is $O\left(n \cdot 2^n\right)$. We present an experimental evaluation of MULE showing its performance, and an extension for faster enumeration of large maximal cliques.

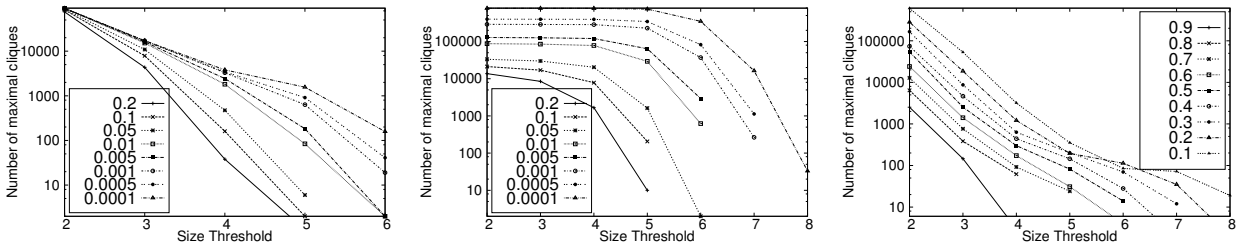
An interesting open problem is to design an algorithm for enumerating maximal cliques from an uncertain graph whose time complexity is worst-case optimal, $O\left(\sqrt{n} \cdot 2^n\right)$. Finally, there are various dense substructures that can be found in a network. Some examples include bicliques, quasi–cliques and k-cores. Finding these dense substructures in the context of uncertain graphs can be an important future direction of work.

Acknowledgments: This work was partially funded by an IBM Faculty Award and by grant 0831903 from the National Science Foundation.



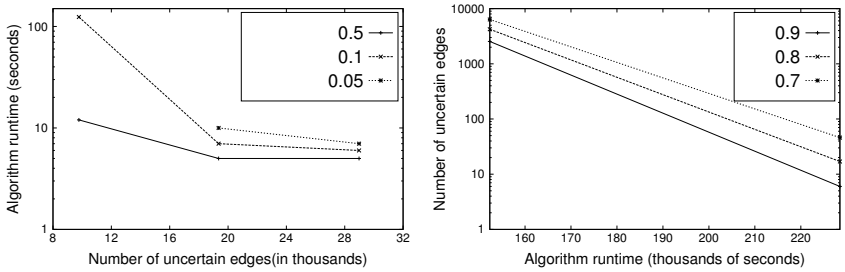
(a) BA10000 (b) ca-GrQc (c) DBLP

Fig. 5: Runtime vs Size threshold of enumerated uncertain maximal cliques



(a) BA10000 (b) ca-GrQc (c) DBLP

Fig. 6: Number of α-maximal cliques vs Size threshold of enumerated uncertain maximal cliques



(a) ca–GrQc (b) DBLP10 (s=4)

Fig. 7: Runtime vs the number of uncertain edges in the graph. The y–axis is in log scale. Observe that the runtime decreases when the number of uncertain edges in the graph increases.

REFERENCES

[1] J. Ghosh, H. Ngo, S. Yoon, and C. Qiao, "On a routing problem within probabilistic graphs and its application to intermittently connected networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, 2007, pp. 1721–1729.

[2] S. Biswas and R. Morris, "Exor: opportunistic multi-hop routing for wireless networks," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 133–144, Aug. 2005.

[3] H. Kawahigashi, Y. Terashima, N. Miyauchi, and T. Nakakawaji, "Modeling ad hoc sensor networks using random graph theory," in *Second IEEE Consumer Communications and Networking Conference*, 2005, pp. 104–109.

[4] E. Adar and C. Re, "Managing uncertainty in social networks," *IEEE Data Engineering Bulletin*, vol. 30, no. 2, pp. 15–22, 2007.

[5] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *Proceedings of the 13th International conference on World Wide Web*, ser. WWW '04. New York, NY, USA: ACM, 2004, pp. 403–412.

[6] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the 9th ACM SIGKDD International conference on Knowledge discovery and data mining*, ser. KDD '03. New York, NY, USA: ACM, 2003, pp. 137–146.

[7] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proceedings of the 12th International conference on Information and knowledge management*, ser. CIKM '03. New York, NY, USA: ACM, 2003, pp. 556–559.

[8] U. Kuter and J. Golbeck, "Using probabilistic confidence models for trust inference in web-based social networks," *ACM Transactions on Internet Technology*, vol. 10, no. 2, pp. 8:1–8:23, Jun. 2010.

[9] P. Boldi, F. Bonchi, A. Gionis, and T. Tassa, "Injecting uncertainty in graphs for identity obfuscation," *Proceedings of the VLDB Endowment*, vol. 5, no. 11, pp. 1376–1387, 2012.

[10] S. Asthana, O. D. King, F. D. Gibbons, and F. P. Roth, "Predicting protein complex membership using probabilistic network reliability," *Genome Research*, vol. 14, pp. 1170–1175, 2004.

[11] J. Bader, A. Chaudhuri, J. Rothberg, and J. Chant, "Gaining confidence in high-throughput protein interaction networks." *Nature Biotechnology*, vol. 22, no. 1, pp. 78–85, 2004.

[12] D. R. Rhodes, S. A. Tomlins, S. Varambally, V. Mahavisno, T. Barrette, S. Kalyana-Sundaram, D. Ghosh, A. Pandey, and A. M. Chinnaiyan, "Probabilistic model of the human protein-protein interaction network," *Nature Biotechnology*, vol. 23, no. 8, pp. 951–959, 2005.

[13] R. Jiang, Z. Tu, T. Chen, and F. Sun, "Network motif identification in stochastic networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 25, pp. 9404–9409, 2006.

[14] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society." *Nature*, vol. 435, no. 7043, pp. 814 – 818, 2005.

[15] O. Rokhlenko, Y. Wexler, and Z. Yakhini, "Similarities and differences of gene expression in yeast stress conditions," *Bioinformatics*, vol. 23, no. 2, pp. 184–190, 2007.

[16] E. Harley and A. Bonner, "Uniform integration of genome mapping data using intersection graphs," *Bioinformatics*, vol. 17, no. 6, pp. 487–494, 2001.

[17] J. Mcauley and J. Leskovec, "Discovering social circles in ego networks,"

ACM Transactions on Knowledge Discovery from Data, vol. 8, no. 1, pp. 4:1–4:28, Feb. 2014.
[18] H. Bernard, P. D. Killworth, and L. Sailer, "Informant accuracy in social network data iv: a comparison of clique-level structure in behavioral and cognitive network data," *Social Networks*, vol. 2, no. 3, pp. 191 – 218, 19791980.
[19] J. Pattillo, N. Youssef, and S. Butenko, "Clique relaxation models in social network analysis," in *Handbook of Optimization in Complex Networks*, ser. Springer Optimization and Its Applications. Springer New York, 2012, pp. 143–162.
[20] G. A. et al., "Functional organization of the yeast proteome by systematic analysis of protein complexes," *Nature*, vol. 415, no. 6868, pp. 141 – 147, 2002.
[21] N. Pathak, S. Mane, and J. Srivastava, "Who thinks who knows who? socio-cognitive analysis of email networks," in *Sixth International Conference on Data Mining*, 2006, pp. 466–477.
[22] E. Harley, A. Bonner, and N. Goodman, "Uniform integration of genome mapping data using intersection graphs," *Bioinformatics*, vol. 17, no. 6, pp. 487–494, 2001.
[23] H. M. Grindley, P. J. Artymiuk, D. W. Rice, and P. Willett, "Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm," *Journal of Molecular Biology*, vol. 229, no. 3, pp. 707–721, 1993.
[24] B. Zhang, B.-H. Park, T. Karpinets, and N. F. Samatova, "From pull-down data to protein interaction networks and complexes with biological relevance," *Bioinformatics*, vol. 24, no. 7, pp. 979–986, 2008.
[25] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 199–208.
[26] A. P. Mukherjee, P. Xu, and S. Tirthapura, "Mining maximal cliques from an uncertain graph," in *Proceedings of the Thirty–First IEEE International Conference Data Engineering*, ser. ICDE '15. IEEE, 2015.
[27] J. Moon and L. Moser, "On cliques in graphs," *Israel Journal of Mathematics*, vol. 3, pp. 23–28, 1965.
[28] Y. Yuan, L. Chen, and G. Wang, "Efficiently answering probability threshold-based shortest path queries over uncertain graphs," in *Database Systems for Advanced Applications*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 5981, pp. 155–170.
[29] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios, "k-nearest neighbors in uncertain graphs," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 997–1008, September 2010.
[30] G. Kollios, M. Potamias, and E. Terzi, "Clustering large probabilistic graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 2, pp. 325–336, 2013.
[31] P. Hintsanen and H. Toivonen, "Finding reliable subgraphs from large probabilistic graphs," *Data Mining and Knowledge Discovery*, vol. 17, no. 1, pp. 3–23, August 2008.
[32] Z. Zou, J. Li, H. Gao, and S. Zhang, "Mining frequent subgraph patterns from uncertain graph data," *IEEE TKDE*, vol. 22, no. 9, pp. 1203–1218, 2010.
[33] R. Jin, L. Liu, and C. C. Aggarwal, "Discovering highly reliable subgraphs in uncertain graphs," in *Proceedings of the 17th ACM SIGKDD International conference on Knowledge discovery and data mining*, ser. KDD '11. New York, NY, USA: ACM, 2011, pp. 992–1000.
[34] Z. Zou, H. Gao, and J. Li, "Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics," in *Proceedings of the 16th ACM SIGKDD International conference on Knowledge discovery and data mining*, ser. KDD '10. New York, NY, USA: ACM, 2010, pp. 633–642.
[35] L. Liu, R. Jin, C. Aggarwal, and Y. Shen, "Reliable clustering on uncertain graphs," in *IEEE 12th International Conference on Data Mining (ICDM)*, 2012, pp. 459–468.
[36] A. Khan, F. Bonchi, A. Gionis, and F. Gullo, "Fast reliability search in uncertain graphs," in *Proceedings of the 16th International Conference on Extending Database Technology*, ser. EDBT '14. New York, NY, USA: ACM, 2014, pp. 535–546.
[37] R. Jin, L. Liu, B. Ding, and H. Wang, "Distance-constraint reachability computation in uncertain graphs," *Proceedings of the VLDB Endowment*, vol. 4, no. 9, pp. 551–562, June 2011.
[38] Z. Zou, J. Li, H. Gao, and S. Zhang, "Finding top-k maximal cliques in an uncertain graph," in *IEEE 26th International Conference on Data Engineering (ICDE)*, 2010, pp. 649–652.
[39] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Communications of ACM*, vol. 16, no. 9, pp. 575–577, September 1973.
[40] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments," *Theoretical Computer Science*, vol. 363, pp. 28–42, October 2006.
[41] D. Eppstein and D. Strash, "Listing all maximal cliques in large sparse real-world graphs," in *Experimental Algorithms*, ser. Lecture Notes in Computer Science, P. Pardalos and S. Rebennack, Eds. Springer Berlin / Heidelberg, 2011, vol. 6630, pp. 364–375.
[42] N. Modani and K. Dey, "Large maximal cliques enumeration in sparse graphs," in *Proceedings of the 17th ACM conference on Information and knowledge management*, ser. CIKM '08. New York, NY, USA: ACM, 2008, pp. 1377–1378.
[43] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa, "A new algorithm for generating all the maximal independent sets," *SIAM Journal on Computing*, vol. 6, no. 3, pp. 505–517, 1977.
[44] N. Chiba and T. Nishizeki, "Arboricity and subgraph listing algorithms," *SIAM Journal on Computing*, vol. 14, pp. 210–223, February 1985.
[45] K. Makino and T. Uno, "New algorithms for enumerating all maximal cliques," in *Algorithm Theory - SWAT 2004*, ser. Lecture Notes in Computer Science, T. Hagerup and J. Katajainen, Eds. Springer Berlin / Heidelberg, 2004, vol. 3111, pp. 260–272.
[46] M. Svendsen, A. P. Mukherjee, and S. Tirthapura, "Mining maximal cliques from a large graph using mapreduce: Tackling highly uneven subproblem sizes," *J. Parallel Distrib. Comput.*, vol. 79, pp. 104–114, 2015.
[47] A. P. Mukherjee and S. Tirthapura, "Enumerating maximal bicliques from a large graph using mapreduce," in *2014 IEEE International Congress on Big Data, Anchorage, AK, USA, June 27 - July 2, 2014*, 2014, pp. 707–716.
[48] J. Leskovec. Stanford large network dataset collection.
[49] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, pp. 47–97, Jan 2002.
[50] C. Dangalchev, "Generation models for scale-free networks," *Physica A: Statistical Mechanics and its Applications*, vol. 338, no. 34, pp. 659 – 671, 2004.

PLACE PHOTO HERE

Arko Provo Mukherjee received his Ph.D. in Computer Engineering from Iowa State University in 2015. He received his Baccalaureate Degree from National Institute of Technology, Durgapur, India, and worked for 3 years as an application developer at IBM. His research interests are in algorithms for large graph mining and tools for large-scale data analytics.

PLACE PHOTO HERE

Pan Xu is a Ph.D. student in Computer Science at University of Maryland, College Park. He received his PhD in Industrial Engineering from Iowa State University.

PLACE PHOTO HERE

Srikanta Tirthapura received his Ph.D. in Computer Science from Brown University in 2002, and his B.Tech. in Computer Science and Engineering from IIT Madras in 1996. He is an Associate Professor in the department of Electrical and Computer Engineering at Iowa State University. His research is concerned with algorithms for big data, including streaming algorithms and parallel and distributed algorithms, and applications to cybersecurity and transportation.