# Fast Maximal Clique Enumeration on Uncertain Graphs: A Pivot-based Approach


## ABSTRACT

Maximal clique enumeration on uncertain graphs is a fundamental problem in uncertain graph analysis. In this paper, we study a problem of enumerating all maximal $(k, \eta)$-cliques on an uncertain graph $\mathcal{G}$, where a vertex set $H$ of $\mathcal{G}$ is a maximal $(k, \eta)$-clique if (1) $H$ ($|H| \geq k$) is a clique with probability no less than $\eta$, and (2) $H$ is a maximal vertex set satisfying (1). The state-of-the-art algorithms for enumerating all maximal $(k, \eta)$-cliques are based on a set enumeration technique which are often very costly. This is because the set enumeration based techniques may explore all subsets of a maximal $(k, \eta)$-clique, thus resulting in many unnecessary computations. To overcome this issue, we propose several novel and efficient pivot-based algorithms to enumerate all maximal $(k, \eta)$-cliques based on a newly-developed pivot-based pruning principle. Our pivot-based pruning principle is very general which can be applied to speed up the enumeration of any maximal subgraph that satisfies a hereditary property. Here the hereditary property means that if a maximal subgraph $H$ satisfies a property $\mathcal{P}$, any subgraph of $H$ also meets $\mathcal{P}$. To the best of our knowledge, our work is the first to systematically explore the idea of pivot for maximal clique enumeration on uncertain graphs. In addition, we also develop a nontrivial size-constraint based pruning technique and a new graph reduction technique to further improve the efficiency. Extensive experiments on nine real-world graphs demonstrate the efficiency, effectiveness, and scalability of the proposed algorithms.



**ACM Reference Format:**

. 2022. Fast Maximal Clique Enumeration on Uncertain Graphs: A Pivot-based Approach. In *Proceedings of SIGMOD '22: International Conference on Management of Data (SIGMOD '22).* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn


## 1 INTRODUCTION

Maximal clique enumeration is a fundamental problem in graph analysis. It is not only closely related to many other graph analysis problems such as maximal independent set [36, 53], graph coloring [2, 56], and maximal common induced subgraphs [28], but it also has numerous applications including social network analysis [38, 57], financial network analysis [5], and bioinformatics [23].

Many real-world networks, however, often contain uncertain information. For example, in Protein-Protein Interaction (PPI) networks, each edge represents an interaction between proteins which are collected by researchers through experiments with inferred links [14, 15, 50]. In social networks [1, 29], the collected relationships between individuals may be missed by automated sensors. In knowledge graphs [10, 11], many types of knowledge are uncertain in nature, where relation facts are usually associated with confidence scores to represent the likelihood of the relation fact being true. These real-world networks, in which each edge is associated with a probability of existence, are referred to as uncertain graphs [24, 25].

Recently, the problem of enumerating all maximal cliques on uncertain graphs [33, 40, 42, 43] has been well studied due to a large number of practical applications [19, 20, 33, 42, 45]. For example, as indicated in [42], the maximal cliques on uncertain graphs can be used to find overlapping multiple protein complexes [19], integrate genome mapping data [20], and analyze email networks [45]. In addition, three other specific applications include clustering quality testing in PPI networks (also tested in [33] partially), community detecting in uncertain knowledge graphs, and task-driven team formation problem, which we evaluated in this paper.

Given an uncertain graph $\mathcal{G}$ and two parameters $k$ and $\eta$, a set $H$ of vertices in $\mathcal{G}$ is called a maximal $(k, \eta)$-clique if (1) $H$ ($|H| \geq k$) is a clique with probability no less than $\eta$, and (2) $H$ is a maximal vertex set satisfying (1). In this paper, we focus on the problem of enumerating all maximal $(k, \eta)$-cliques in an uncertain graph $\mathcal{G}$. Similar to the classic maximal clique enumeration on deterministic graphs [7], the maximal $(k, \eta)$-clique enumeration is NP-hard as proved by Mukherjee et al. [43]. To solve this problem, Mukherjee et al. proposed a recursive backtracking enumeration algorithm based on a set enumeration technique. To boost the efficiency, Li et al. [33] proposed an improved maximal $(k, \eta)$-clique enumeration algorithm based on a carefully-designed graph reduction technique. Instead of enumerating maximal $(k, \eta)$-cliques on the original uncertain graph, their algorithm first prunes the vertices that are definitely not contained in any maximal $(k, \eta)$-clique, and then employs the same set enumeration based technique to find all maximal $(k, \eta)$-cliques on the reduced uncertain graph.

Although the state-of-the-art algorithm [33] can significantly improve the enumeration performance, it is still based on the classical set enumeration technique which often includes many unnecessary computations (because the set enumeration technique needs to explore all subsets of a given set). For example, consider an uncertain graph $\mathcal{G}$ induced by the vertices $\{v_4, \cdots, v_8\}$ in Fig. 1. Assume that $k = 1$ and $\eta = 0.5$. When using the state-of-the-art algorithm [33] to enumerate all maximal $(k, \eta)$-cliques in $\mathcal{G}$, a total number of 30 non-maximal $(k, \eta)$-cliques will be explored, because the set enumeration technique needs to explore all subsets of $\{v_4, \cdots, v_8\}$. However, in this example, there is only one maximal $(k, \eta)$-clique in $\mathcal{G}$ (i.e., $\{v_4, \cdots, v_8\}$), thus the set enumeration based algorithm includes many unnecessary computations.

Note that the traditional maximal clique enumeration problem can be efficiently solved by a classic Bron-Kerbosch algorithm with




*SIGMOD '22, June 12–17, 2022, Philadelphia, PA, USA*





ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

a pivot technique [7, 17, 52]. The main idea of such a pivot-based pruning technique is that every maximal clique either contains a vertex $v \in V$ or a non-neighbor of $v$. Thus, in the maximal clique enumeration procedure, if a vertex $v$, called a pivot vertex, is selected, then all its neighbors can be pruned in the current recursion, thus significantly reducing search branches. Unfortunately, we show that the idea of the previous pivot technique cannot be extended to solve the maximal $(k, \eta)$-clique enumeration problem. This is because in each recursion, the neighbors of a pivot vertex may form a $(k, \eta)$-clique with the current clique, thus cannot be skipped (details are given in Section 3). To overcome this problem, we need to seek different ideas to design novel pivot techniques.

To achieve this goal, we first propose a novel and general pivot technique which can be used to speed up the enumeration of maximal subgraphs that satisfy the hereditary property, where the hereditary property means that if a maximal subgraph satisfies a property $\mathcal{P}$, all its subgraphs also satisfy $\mathcal{P}$. For convenience, we refer to a subgraph as the $\mathcal{P}$-subgraph if it meets a hereditary property $\mathcal{P}$. Note that the idea of our general pivot technique is completely different from that of the traditional pivot technique. Specifically, the key idea of our general pivot technique is to identify a *periphery set* $P$ such that in each recursion, the current clique $R$ together with $P$ cannot contain any maximal $\mathcal{P}$-subgraph. Since $R \cup P$ does not contain any solutions, the enumeration algorithm can safely prune the vertices in $P$. Based on the general pivot technique, we develop a novel technique, called MM-pivot, to identify the periphery set $P$ for enumerating all maximal $(k, \eta)$-cliques, as the maximal $(k, \eta)$-clique satisfies the hereditary property. We show that the MM-pivot technique can substantially reduce the unnecessary recursive calls in the maximal $(k, \eta)$-clique enumeration procedure. We also propose a nontrivial size-constraint based punning technique and a new graph reduction technique to further improve the efficiency of our algorithms. To summarize, the main contributions of this paper are as follows.

**A general pivot principle.** We develop a novel and general pivot principle to efficiently enumerate all maximal $\mathcal{P}$-subgraphs in a graph. To justify the correctness, we prove sufficient and necessary conditions of the pivot principle for enumerating all maximal $\mathcal{P}$-subgraphs. To the best of our knowledge, we are the first to systematically develop pivot-based algorithms for enumerating all maximal $\mathcal{P}$-subgraphs. We believe that the proposed pivot principle could be of independent interests, which can be used for many other maximal hereditary subgraph enumeration problems.

**Novel pivot-based algorithms.** Based on the general pivot principle, we develop a novel pivot technique, called MM-pivot, to speed up the maximal $(k, \eta)$-clique enumeration. We also devise a MM-pivot based maximal $(k, \eta)$-clique enumeration algorithm which can significantly reduce the unnecessary computations of the state-of-the-art algorithm. More interestedly, we show that the MM-pivot technique can be progressively refined to enhance the pruning performance during the enumeration procedure.

**New optimization techniques.** To further improve the efficiency, we also develop a nontrivial size-constraint based pruning technique and a new graph reduction technique. Specifically, the size-constraint based pruning technique is based on the general pivot principle. That is, we construct a *specific* pivot-based pruning method based on the size constraint of the $(k, \eta)$-clique. The graph reduction technique is based on a newly-proposed concept called $(Top_k, \eta)$-triangle (more details in Section 5.2). We show that any maximal $(k, \eta)$-clique must be contained in the $(Top_{k-2}, \eta)$-triangle of the uncertain graph $\mathcal{G}$. The algorithm only needs to enumerate all maximal $(k, \eta)$-cliques in $(Top_{k-2}, \eta)$-triangle, instead of in the original uncertain graph $\mathcal{G}$, thus significantly improving the efficiency.

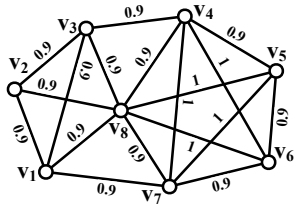



**Figure 1: Running example (an uncertain $\mathcal{G}$)**

**Extensive experiments.** We conduct comprehensive experiments using nine real-world graphs to evaluate the efficiency, effectiveness, and scalability of the proposed algorithms. The results show that our algorithms significantly outperform the state-of-the-art algorithm on all datasets. Moreover, our best algorithm can be up to two orders of magnitude faster than the state-of-the-art algorithm on large datasets. For reproducibility purposes, the source code of this paper is released at an anonymous source https://tinyurl.com/bde4wute.

## 2 PROBLEM STATEMENT

Let $\mathcal{G} = (V, E, p)$ be an uncertain graph, where $V$ is a set of vertices, $E$ is a set of edges, and $p : E \rightarrow (0, 1]$ is a function that assigns each edge $e \in E$ a probability indicating the likelihood of $e$'s existence. Let $n = |V|$ and $m = |E|$ be the number of vertices and edges of $\mathcal{G}$ respectively. Denote by $N_v(\mathcal{G}) = \{u | (u, v) \in E\}$ a set of neighbors of $v$ in $\mathcal{G}$. The degree of $v$, denoted by $d_v(\mathcal{G})$, is the cardinality of $N_v(\mathcal{G})$, i.e., $d_v(\mathcal{G}) = |N_v(\mathcal{G})|$. Following the standard uncertain graph model [24, 33, 43], we assume that the existing probabilities of edges in $\mathcal{G}$ are independent. Based on this assumption, a widely-used possible world model can be applied to analyze uncertain graphs. Specifically, a possible world of $\mathcal{G}$ is a deterministic graph, denoted by $G = (V, E_G)$, which is obtained by sampling an edge set $E_G$ from $E$ w.r.t. (with respect to) the probability function $p$. The probability of a possible world $G$, denoted by $Pr(G)$, is defined as

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

Clearly, the number of possible worlds of an uncertain graph $\mathcal{G}$ is $2^{|E|}$. Given a possible world $G$, a clique $H$ is a completed subgraph in $G$. If there does not exist a clique $H'$ such that $H \subset H'$, we say that $H$ is maximal in $G$. Below, we give the definition of clique probability [33, 43].

*Definition 1 (Clique probability). Given an uncertain graph $\mathcal{G}$ and a set of vertices $H$, the clique probability of $H$, denoted by $Pr(H, \mathcal{G})$, is defined as the probability that in a sampled possible world of $\mathcal{G}$, $H$ is a clique.*

Assume that the probability of $(u, v)$ is 0 if $(u, v) \notin E$ and $u, v \in V$. As shown in [43], the clique probability of $H$ can be computed by

$$Pr(H, \mathcal{G}) = \prod_{e \in \{(u,v) | u,v \in H, u \neq v\}} p_e. \quad (2)$$

Based on Definition 1, a concept of $\eta$-clique on an uncertain graph $\mathcal{G}$ can be defined as follows.

*Definition 2 ($\eta$-clique). Given an uncertain graph $\mathcal{G}$ and a probability threshold $\eta \in [0, 1]$, a set of vertices $H$ is an $\eta$-clique if $Pr(H, \mathcal{G}) \geq \eta$.*

A vertex set $H$ is a maximal $\eta$-clique if (1) it is a $\eta$-clique, and (2) there does not exist a vertex set $H' \supset H$ such that $H'$ is an $\eta$-clique. For a typical uncertain graph, there may exist many maximal $\eta$-cliques with small size which are often no practical use in real-world applications [33, 43]. Thus, it is more useful to detect large maximal $\eta$-cliques for real-world applications. Below, we give the definition of $(k, \eta)$-clique which is a maximal $\eta$-clique with at least $k$ vertices.

*Definition 3 (Maximal $(k, \eta)$-clique). Given an uncertain graph $\mathcal{G}$, a probability threshold $\eta \in [0, 1]$, and a positive integer $k \in \mathbb{N}^+$. A vertex set $H \subseteq V$ is a maximal $(k, \eta)$-clique if $|H| \geq k$ and $H$ is a maximal $\eta$-clique.*

**Problem definition.** Given an uncertain graph $\mathcal{G}$ and two parameters $\eta \in [0, 1]$ and $k \in \mathbb{N}^+$, our goal is to efficiently enumerate all maximal $(k, \eta)$-cliques in $\mathcal{G}$.

## 3 DEFECTS OF EXISTING SOLUTIONS

To enumerate all maximal $(k, \eta)$-cliques, Mukherjee et al. [43] proposed a recursive backtracking algorithm based on a set enumeration technique [48]. The main idea of this algorithm is to maintain three disjoint sets $R$, $C$, and $X$ in the recursive enumeration procedure, where $R$ is an $\eta$-clique, $C$ is a set of candidates that are to be added to $R$ to form a larger $\eta$-clique, and $X$ is a set of vertices that have already been explored from $C$. In each recursion, the three vertex sets always keep the invariance that $R \cup \{v\}$ is an $\eta$-clique if $v \in C \cup X$, and $R \cup \{v\}$ is not an $\eta$-clique if $v \notin C \cup X$. The algorithm iteratively processes the candidate vertices in $C$ following a lexicographical ordering to expand the current $\eta$-clique $R$. Before entering the next recursion, the sets $C$ and $X$ are updated to keep the above-mentioned invariant constraint. Whenever $C \cup X = \emptyset$ and $|R| \geq k$, the set $R$ is a valid $(k, \eta)$-clique. The detailed procedure is shown in Algorithm 1.

In Algorithm 1, it invokes the procedure MUCE to enumerate all maximal $(k, \eta)$-cliques for each connected component of $\mathcal{G}$ (lines 1-3), where MUCE admits six arguments: the $\eta$-clique $R$, the clique probability $q$ of $R$, the candidate set $C$, the set $X$, and two parameters $k$ and $\eta$. Note that every element in $C$ and $X$ is a pair of $(v, r_v)$ which is initialized by $(v, 1)$ [43], where $v$ is a vertex satisfying that $R \cup \{v\}$ is a $\eta$-clique, and $r_v$ is the product of probabilities of the edges connecting $v$ and the vertices in $R$. The algorithm iteratively processes each vertex $v \in C$ in a lexicographical ordering to expand the current $\eta$-clique $R$. In each iteration, the algorithm updates the sets $C$ and $X$ using Algorithm 2 to keep the invariance that $R' \cup \{u\}$ is an $\eta$-clique when $u \in C' \cup X'$ (lines 8-10). If there may be a $(k, \eta)$-clique containing $R'$, i.e., $|R'| + |C'| \geq k$, the algorithm continues the recursive calls (lines 11-12). After processing the vertex $v$, the algorithm removes $v$ from $C$ and adds it into $X$ (line 13). The algorithm terminates until all vertices in $C$ are processed.

An improved solution to enumerate all maximal $(k, \eta)$-cliques was developed by Li et al. [33]. Their algorithm first applies several effective pruning techniques to remove vertices that are definitely not included in any maximal $(k, \eta)$-clique, and then makes use of the same recursive backtracking technique as used in Algorithm 1 on the reduced graph to find all maximal $(k, \eta)$-cliques. Since the algorithm proposed by Li et al. [33] works on the reduced graph which is often much smaller than the original uncertain graph, it is much faster than the algorithm proposed by Mukherjee et al. [43].

**Algorithm 1:** MUC$(\mathcal{G}, k, \eta)$ [43]

**Input:** An uncertain graph $\mathcal{G}$, and two parameters $k$ and $\eta$.
**Output:** All maximal $(k, \eta)$-cliques in $\mathcal{G}$.

1 **foreach** *connected component $C$ in $\mathcal{G}$* **do**
2  Replace each $v$ in $C$ with $(v, 1)$;
3  MUCE$(\emptyset, 1, C, \emptyset, k, \eta)$;
4 **Procedure** MUCE$(R, q, C, X, k, \eta)$
5 **if** $C \cup X = \emptyset$ *and* $|R| \geq k$ **then**
6  Output $R$ as a maximal $(k, \eta)$-clique;
7 **foreach** $(v, r) \in C$ *in lexicographical ordering over* $v$ **do**
8  $R' \leftarrow R \cup \{v\}$; $q' \leftarrow q * r$;
9  $C' \leftarrow$ GenerateSet$(v, C, q', \eta)$; ▷ Algorithm 2
10  $X' \leftarrow$ GenerateSet$(v, X, q', \eta)$;
11  **if** $|R'| + |C'| \geq k$ **then**
12  MUCE$(R', q', C', X', k, \eta)$;
13  $C \leftarrow C \setminus \{(v, r)\}$; $X \leftarrow X \cup \{(v, r)\}$;

**Algorithm 2:** GenerateSet$(v, C, q, \eta)$ [43]

1 $C' \leftarrow \emptyset$;
2 **foreach** $(u, r) \in C$ **do**
3  **if** $u \in N_v(\mathcal{G})$ **then**
4  $q' \leftarrow q * r * p_{(u,v)}$;
5  **if** $q' \geq \eta$ **then** $C' \leftarrow C' \cup (u, r * p_{(u,v)})$ ;
6 **return** $C'$;

The main defect of two existing enumeration algorithms is that to enumerate a maximal $(k, \eta)$-clique $H$, they need to explore all the non-maximal $(k, \eta)$-cliques contained in $H$ based on the set enumeration technique (because the set enumeration technique needs to explore all subsets of a given set [48]), resulting in poor performance. For example, consider an uncertain graph $\mathcal{G}$ induced by the vertex set $\{v_4, \cdots, v_8\}$ in Fig. 1. Assume that $k = 1$ and $\eta = 0.5$. When using the algorithm proposed in [33] to enumerate all maximal $(k, \eta)$-cliques in $\mathcal{G}$, a total number of 31 $(k, \eta)$-cliques (all vertex subsets of $\{v_4, \cdots, v_8\}$) are explored by this algorithm). However, there is only one maximal $(k, \eta)$-clique in $\mathcal{G}$. Thus, many unnecessary computations exist in this algorithm. In the following, we show that it is very challenging to reduce unnecessary computations by using existing techniques.

**A failed attempt.** Recall that on deterministic graphs, there exist many efficient algorithms to enumerate all maximal cliques based on the classic Bron-Kerbosch algorithm with pivot techniques [7, 8, 52]. Specifically, the key idea of these algorithms is that every maximal clique of a deterministic graph either contains a vertex $v \in V$ or a non-neighbor of $v$ (i.e., if a clique $R$ only contains the neighbor vertices of $v$, the vertex $v$ can be added into $R$ to form a larger clique, thus it is not maximal). Thus, in the maximal clique enumeration procedure, if a vertex $v$ in $C$ or $X$, called a pivot vertex, is selected,

then all neighbors of $v \in C$ can be skipped to expand the current clique $R$, thus significantly reducing the number of recursive calls. Unfortunately, such a pivot technique is failed to enumerate all maximal $(k, \eta)$-cliques; and the detailed reasons are as follows.

Based on existing pivot techniques, all neighbors of a pivot vertex $v$ in the candidate set $C$ will be skipped to expand $R$. However, for uncertain graphs, such neighbors may form a maximal $\eta$-clique with the current clique $R$ (some subsets of a maximal clique may be maximal $\eta$-cliques of $\mathcal{G}$). Therefore, some results can be missed by existing pivot techniques to enumerate all maximal $(k, \eta)$-cliques, which indicates that existing pivot techniques no longer hold for maximal $(k, \eta)$-clique enumeration. For example, reconsider the uncertain graph in Fig. 1. Suppose that $\eta = 0.65$. Then, the set $\{v_4, v_5, v_6, v_7\}$ is a maximal $\eta$-clique in $\mathcal{G}$, but it is not a maximal clique in the corresponding deterministic graph. In the clique enumeration procedure, if we choose $v_8$ as a pivot vertex at the top recursion, then the vertices $\{v_4, v_5, v_6, v_7\}$ will be skipped to expand $R$. Thus, only the cliques containing $R \cup \{v_8\}$ may be found, while the maximal $\eta$-clique $\{v_4, v_5, v_6, v_7\}$ is missed.

Moreover, even if the probabilities of edges are taken into consideration, existing pivot techniques still do not work. The idea is that we first select a pivot vertex $v$ from $C$, and then skip the neighbor set $C' \subseteq C$ of $v$ to expand $R$ such that for each $u \in C'$, $R \cup \{v, u\}$ is an $\eta$-clique. Note that $C'$ can be computed by Algorithm 2. It seems that all maximal $\eta$-cliques that contain the neighbors of $v$ can be obtained by expanding $v$ to $R$. But unfortunately, such a result is incorrect. Suppose that (1) we have three $\eta$-cliques $R \cup \{v, u_1\}$, $R \cup \{v, u_2\}$, and $R \cup \{u_1, u_2\}$ in $\mathcal{G}$; and (2) $R \cup \{v, u_1, u_2\}$ is not a $\eta$-clique, because its clique probability is smaller than the threshold $\eta$. Note that such an assumption can be easily satisfied in many uncertain graphs. Clearly, in this case, the $\eta$-clique $R \cup \{u_1, u_2\}$ will be missed when choosing $v$ as a pivot vertex.

Is there a pivot technique that can be applied to enumerate all maximal $(k, \eta)$-cliques on uncertain graphs? We answer this question affirmatively based on a novel result that we found, which will be introduced in the following section.

## 4 NEW ENUMERATION TECHNIQUES

In this section, we first develop a novel and general *pivot principle* for the problem of enumerating all maximal subgraphs that satisfy the *hereditary property*. Then, based on the general pivot principle and the observation that the maximal $(k, \eta)$-clique satisfies the hereditary property, we propose two new pivot techniques for enumerating all maximal $(k, \eta)$-cliques. Below, we first introduce the general pivot principle for enumerating all maximal hereditary subgraphs, and then present pivot techniques for the problem of enumerating all maximal $(k, \eta)$-cliques.

### 4.1 A New and General Pivot Principle

Here we develop a general pivot principle for the problem of maximal $\mathcal{P}$-subgraph enumeration, where $\mathcal{P}$ denotes a hereditary property. Given a graph $G$, a subgraph $S$ of $G$ is a maximal $\mathcal{P}$-subgraph if (1) any subgraph of $S$ satisfies $\mathcal{P}$ (hereditary property) and (2) there is no other vertex $v$ in $G$ such that $S \cup \{v\}$ satisfies $\mathcal{P}$ (maximal property). Note that we can make use of the classic backtracking set enumeration technique, denoted by Recursion$(R, C, X)$, to enumerate all maximal $\mathcal{P}$-subgraphs. Similar to Algorithm 1, Recursion$(R, C, X)$ admits three input parameters $R$, $C$ and $X$, where $R$ is a $\mathcal{P}$-subgraph, $C$ is the candidate set satisfying that $R \cup \{v\}$ is a $\mathcal{P}$-subgraph for each $v \in C$, and $X$ is the set of vertices already processed in $C$.

**Algorithm 3:** PivotEnum($G$)

**Input:** A graph $G$.
**Output:** All maximal $\mathcal{P}$-subgraphs of $G$.

```
1 PivotRecursion(∅, V, ∅);
2 Procedure PivotRecursion(R, C, X)
3 if C ∪ X = ∅ then Output R as a maximal P-subgraph;
4 Detect a periphery set P in C such that R ∪ P does not contain any
    maximal P-subgraph of G;
5 foreach v ∈ C \ P do
6     Generate C' ⊂ C and X' ⊆ X such that R ∪ {v, u} is a
        P-subgraph for each u in C' ∪ X';
7     PivotRecursion(R ∪ {v}, C', X');
8     Move v from C to X;
```

To reduce the unnecessary recursive calls of Recursion$(R, C, X)$, one may conjecture whether we can skip some vertices in $C$ to expand $R$ without losing any $\mathcal{P}$-subgraph of $G$. If the conjecture is true, the pivot technique is accordingly applicable for the set enumeration algorithm. Next, we first perform some theoretical analysis for this conjecture, and then we show that there indeed exists a novel pivot principle that makes this conjecture true. Without loss of generally, we define a subset of vertices in candidate set $C$ that can be ignored to expand $R$ as a periphery set.

*Definition 4 (Periphery set). Given a graph $G$, a set $P \subseteq C$ of vertices is the periphery set for a recursive call* Recursion$(R, C, X)$ *if such a recursive call only needs to expand $R$ from $C \setminus P$.*

To verify the conjecture, it is sufficient to show whether all maximal $\mathcal{P}$-subgraphs of $G$ containing $R$ can be enumerated by Recursion$(R, C, X)$ with the periphery set $P$. Note that, in this paper, we only consider the nontrivial case in which $P$ is not empty. Below, we give a necessary condition for the periphery set $P$ that must satisfy. All proofs of this paper are omitted due to the space limit.

*Lemma 1 (Necessary condition). For any recursive call* Recursion$(R, C, X)$*, given a periphery set $P \subseteq C$, any maximal $\mathcal{P}$-subgraph $S$ of $G$ that contains $R$ cannot be included in the subgraph of $G$ induced by $R \cup P$, i.e., given any maximal $\mathcal{P}$-subgraph $S \supset R$ in $G$, $S \nsubseteq R \cup P$ if $P \subseteq C$ is a periphery set in* Recursion$(R, C, X)$.

Proof. Assume, on the contrary, that there is a maximal $\mathcal{P}$-subgraph $S' \supset R$ of $G$ such that $S' \subseteq R \cup P$ in a recursive call. Let $S'$ be the set of vertices with $\{v_1, v_2, \ldots, v_h\}$, where $h = |S'|$ and $v_i < v_j$ for each $i, j \leq h$ and $i \neq j$. Following the set enumeration technique, we adopt the lexicographical order of vertices in $C$ to enumerate all maximal $\mathcal{P}$-subgraphs of $G$. It is easy to see that the maximal $\mathcal{P}$-subgraph $S'$ can only be enumerated by the subroutines containing $R = \{v_1\}$. Suppose that Recursion$(R_i, C_i, X_i)$ is the first recursive call including a periphery set $P$ such that $S' \subseteq R_i \cup P$, where $i \geq 1$ is the depth of the current recursive call in the recursion tree, and $R_i$ is the subset of $S'$ containing $v_1$. Based on Definition 4, only the vertices in $C_i \setminus P$ can be used to expand $R_i$. We have $R' = R_i \cup \{u\}$ in next recursive calls where $u \in C_i \setminus P$ and $u \notin S'$. Based on the

result that every maximal $\mathcal{P}$-subgraph $S$ generated by subroutines of Recursion$(R', C', X')$ must contain the set $R'$, we can easily derive that Recursion$(R_i, C_i, X_i)$ cannot generate the maximal $\mathcal{P}$-subgraph $S'$, because there always exists a vertex $u \notin S'$. Moreover, after enumerated all maximal $\mathcal{P}$-subgraphs that contains $R_i$, there is a vertex $u \in R_i$ that needs to be moved into the set $X$, thus any maximal $\mathcal{P}$-subgraphs containing $u$ cannot be generated by the other Recursion$(R'', C'', X'')$ such that $u \in X''$ and $|R''| \leq i$. As a result, the maximal $\mathcal{P}$-subgraph $S'$ cannot be enumerated by the periphery set $P$, and the lemma is established. □

Following Lemma 1, a remaining question is that if we obtain a subset $P$ of $C$ such that $R \cup P$ does not contain any maximal $\mathcal{P}$-subgraph of $G$ in each recursive call, then whether all maximal $\mathcal{P}$-cliques of $G$ can be enumerated based on such a periphery set $P$? We have the following result which answers this question affirmatively.

*Lemma 2 (Sufficient condition). Given a graph G, all maximal $\mathcal{P}$-subgraphs of G can be enumerated by the set enumeration algorithm with a periphery set P such that $R \cup P$ does not contain any maximal $\mathcal{P}$-subgraph of G in each recursive call.*

Proof. To prove this lemma, it is sufficient to verify that (i) all maximal $\mathcal{P}$-subgraphs that contains $R$ can be enumerated by Recursion$(R, C, X)$ with the periphery set $P$; (ii) the obtained maximal $\mathcal{P}$-subgraphs are no redundant; and (iii) all results generated by Recursion$(R, C, X)$ are maximal. We first show the correctness of (i). Let $S$ be an arbitrary maximal $\mathcal{P}$-subgraph containing $R$, we only need to prove that $S$ can be enumerated by Recursion$(R, C, X)$ with a periphery set $P$. According to Definition 4, only the vertices in $C \setminus P$ are considered to expand $R$. It can be seen that there exists a vertex $v$ in $C \setminus P$ such that $v \in S$ based on the principle of $S \nsubseteq R \cup P$. Then, an expanded subset $R' = R \cup \{v\}$ of $S$ can be obtained by selecting $v$ to expand $R$. Moreover, in the next recursive call, a larger $\mathcal{P}$-subgraph $R' \cup \{v'\}$ such that $R' \cup \{v'\} \subseteq S$ can be obtained in a similar way. Thus, any maximal $\mathcal{P}$-subgraph $S$ can be output by Recursion$(R, C, X)$ with periphery set $P$. For case (ii), after each vertex $v \in C \setminus P$ used for expanding, it needs to be moved from $C$ to $X$. Thus, any maximal $\mathcal{P}$-subgraph containing $R$ and $v$ cannot be generated multiple times by Recursion$(R, C, X)$. For case (iii), if there is a non-maximal $\mathcal{P}$-subgraph $S'$ is obtained by Recursion$(R, C, X)$ with a periphery set $P$, there must satisfy that $S'$ is maximal in $R \cup C$, and for every vertex $v \in X$, $S' \cup \{v\}$ is not a $\mathcal{P}$-subgraph. It can be derived that there exists a vertex $v \in V$ such that $v \notin C \cup X$ and $S' \cup \{v\}$ is a $\mathcal{P}$-subgraph, which is a contradiction. Thus, the lemma is established. □

Lemma 1 and Lemma 2 guarantee that all maximal $\mathcal{P}$-subgraphs containing $R$ can be enumerated by Recursion$(R, C, X)$ with a periphery set. Based on this, a general pivot-based framework for enumerating all maximal $\mathcal{P}$-subgraphs is proposed in Algorithm 3. Note that the key point of Algorithm 3 is to find a periphery set in each recursive call (line 4). In the following, we will show how to identify an effective periphery set for maximal $(k, \eta)$-clique enumeration.

## 4.2 The M-pivot Technique

In this subsection, we develop specific pivot techniques for enumerating all maximal $(k, \eta)$-cliques on uncertain graphs based on

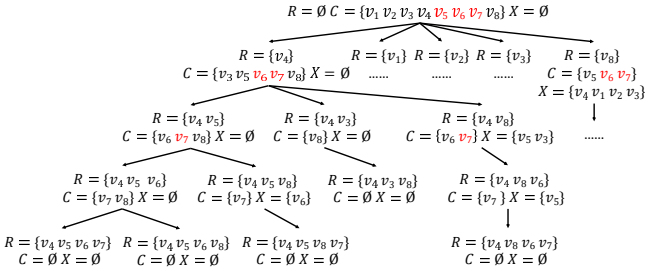



**Figure 2: An enumeration tree of $\mathcal{G}$ in Fig. 1 with the M-pivot technique by setting $k = 1$ and $\eta = 0.65$ (the red vertices in each $C$ form the periphery set; here we omit the probabilities associated with vertices in $C$ and $X$).**

the general pivot principle developed in Section 4.1. It is easy to verify that all $\eta$-cliques in $\mathcal{G}$ meet the hereditary property, i.e., for an $\eta$-clique $R$ of $\mathcal{G}$, any subgraph of $R$ is also an $\eta$-clique. Therefore, by the analysis in Section 4.1, we can reduce the number of recursive calls of MUCE in enumerating maximal $(k, \eta)$-cliques using the pivot technique. The remaining issue is how to determine the periphery set $P$ of $C$ such that $R \cup P$ cannot contain any maximal $(k, \eta)$-clique in each recursion Recursion$(R, C, X)$. Below, we first develop a simple but efficient method, called M-pivot, to construct such a periphery set. The main idea of M-pivot is that if we can detect a maximum $\eta$-clique that contains a subset $R$ in a recursive call Recursion$(R, C, X)$, we can safely prune the vertices in the candidate set $C$ that are also included in the detected maximum $\eta$-clique. Formally, we have the following result.

*Lemma 3. Given an uncertain graph $\mathcal{G}$, all maximal $(k, \eta)$-cliques of $\mathcal{G}$ can be enumerated with the periphery set $P \subseteq C$ if $R \cup \{p_v\} \cup P$ is a maximum $\eta$-clique in $R \cup C$ that contains $R \cup \{p_v\}$, where $p_v \in C$ is referred to as a pivot vertex.*

Proof. Since $R \cup \{p_v\} \cup P$ is a maximum $\eta$-clique, $R \cup P$ cannot be a maximal $\eta$-clique. Thus, the lemma is obviously established by the results shown in Lemma 2. □

Lemma 3 indicates that we can find a periphery set by detecting a maximum $\eta$-clique in each recursive call. Clearly, by Lemma 3, the size of the periphery set is dominated by the maximum $\eta$-clique containing $R \cup \{p_v\}$, where $p_v$ is the pivot vertex selected from the candidate set $C$. Thus, in practice, it is better to select a pivot vertex $p_v$ from $C$ to make the maximum $\eta$-clique containing $R \cup \{p_v\}$ as large as possible (some heuristic pivot selection strategies will be discussed in Section 4.6). An example of the enumeration tree with the M-pivot technique is shown in Fig. 2, and a brief explanation of this example is given as follows.

*Example 1. Given an uncertain graph $\mathcal{G}$ in Fig. 1, assume that $k = 1$ and $\eta = 0.65$. The initial parameters of* MUCE *are $R = \emptyset$, $C = V$, and $X = \emptyset$ respectively. Assume that the pivot vertex of* MUCE$(R, C, X)$ *is $v_4$. Then, the periphery set $P$ can be $\{v_5, v_6, v_7\}$, as $\{v_4, v_5, v_6, v_7\}$ is a maximum $(k, \eta)$-clique. By the* M-pivot *technique, only vertices in $C \setminus P$ are used to expand the current $\eta$-clique $R$. When $R$ is expanded with the vertex $v_4$, the candidate set is updated to $C_1 = \{(v_3, 0.9), (v_5, 0.9), (v_6, 1), (v_7, 1), (v_8, 0.9)\}$ which will be computed in the next recursive call. Furthermore, a new pivot vertex*

*in $C_1$ can also be selected in the second recursion. Suppose that the selected pivot vertex is $v_5$. Then, the periphery set $P_1$ can be $\{v_6, v_7\}$. Thus, the algorithm only picks the vertices in $C_1 \setminus P_1$ to expand $R$. The procedure continues until all vertices in $C$ of the top recursion have been processed. The enumeration tree is shown in Fig. 2.*

To implement the M-pivot technique, a remaining question is how do we obtain the maximum $\eta$-clique that contains $R \cup \{p_v\}$ in advance? Obviously, the traditional set enumeration algorithms cannot meet such a requirement. To overcome this issue, we first introduce a key observation that for any selected pivot vertex $p_v \in C$, the algorithm always needs to enumerate all maximal $(k, \eta)$-cliques containing $p_v$. Therefore, a simple approach to find the maximum $\eta$-clique containing $R \cup \{p_v\}$ is that the algorithm first expands the pivot vertex $p_v$ to $R$ and enumerates all maximal $(k, \eta)$-cliques containing $R \cup \{p_v\}$; and then the algorithm picks the largest one. Note that there may be multiple maximum $\eta$-cliques containing $R \cup \{p_v\}$. In our algorithm, we choose the first found maximum $\eta$-clique. After completing the recursive call Recursion$(R \cup \{p_v\}, C', X')$, a maximum $\eta$-clique $Q$ can be found; and the algorithm backtracks to the recursive call Recursion$(R, C, X)$. Then, $Q \cap C$ can be severed as a periphery set to prune the candidate set $C$ in Recursion$(R, C, X)$.

### 4.3 An Improved M-pivot Technique

Recall that in the M-pivot method, the periphery set $P$ is mainly determined by the set $R \cup \{p_v\}$, where $p_v$ is the selected pivot vertex from a candidate set $C$. Typically, it is not easy to generate a periphery set with the maximum size for a candidate set $C$ (a larger periphery set can prune more vertices in $C$). Here we develop an improved M-pivot technique that can progressively refine the periphery set. The key idea of this technique is that the algorithm first invokes the M-pivot technique to obtain a periphery set $P$. Then, it recursively processes the vertices in $C \setminus P$. In each recursion, if the algorithm finds a maximum $\eta$-clique with size larger than $|R|+|P|+1$ (here the maximum $\eta$-clique is taken over all maximal $\eta$-cliques found in that recursion), the algorithm replaces the periphery set obtained previously with a larger maximum $\eta$-clique, thus improving the pruning performance.

*Lemma 4. For each recursive call, a periphery set $P$ can be replaced with a set $P' \subset C$ if $|P'| > |P|$ and $P' \cup R \cup \{v\}$ is a maximum $\eta$-clique found in the previous recursive call, where $R$ is the $\eta$-clique, $C$ is the candidate set, and $v \in C \setminus P$.*

Proof. Since $P' \cup R \cup \{v\}$ is a maximum $\eta$-clique, $R \cup P'$ does not contain any maximal $\eta$-clique. Thus, by Lemma 2, the enumeration algorithm is still correct with an improved periphery set $P'$. □

Note that such an improved M-pivot technique can be implemented by slightly modifying the *basic* M-pivot method developed in Section 4.2. Specifically, in each recursive call, the algorithm first selects a pivot vertex $p_v \in C$ to generate a periphery set $P$ based on Lemma 3. Then, for each $v \in C \setminus P$, the algorithm computes all maximal $(k, \eta)$-cliques containing $R \cup \{v\}$, and records the detected maximum $\eta$-clique so far. If there is a maximum $\eta$-clique $P' \cup R \cup \{u\}$ such that $|P'| \geq |P|$, the algorithm replaces $P$ with $P'$ and continues to expand $R$ with vertices in $C \setminus P'$. The following example illustrates the idea of the improved M-pivot technique.

*Example 2. Given an uncertain graph $\mathcal{G}$ shown in Fig. 1, let $R = \emptyset$, $C = V$, and $X = \emptyset$ be the initial parameters of the recursive algorithm. Assume that $k = 1$ and $\eta = 0.53 < 0.9^6$. The algorithm first selects a pivot vertex $p_v$ (suppose $p_v = v_1$) in $C$ to enumerate all maximal $(k, \eta)$-cliques containing $p_v$. Then, the algorithm can obtain a periphery set $P = \{v_2, v_3, v_8\}$, since $\{v_1, v_2, v_3, v_8\}$ is the maximum $(k, \eta)$-clique containing $v_1$ in the subgraph induced by $R \cup C$. Clearly, the algorithm only needs to expand $R$ with vertices in $C \setminus P$. For a vertex $v_4 \in C \setminus P$, all maximal $(k, \eta)$-cliques containing $v_4$ need to be computed, and the algorithm can obtain a maximum $(k, \eta)$-clique $\{v_4, v_5, v_6, v_7, v_8\}$. Based on Lemma 4, the periphery set $P$ can be updated to $\{v_5, v_6, v_7, v_8\}$. As a result, only vertices $\{v_2, v_3\}$ are left to expand $R$ in the top recursion. This example illustrates that the improved* M-pivot *technique can significantly boost the pruning performance over the basic* M-pivot *method.*

### 4.4 The Pivot-based Enumeration Algorithm

Note that the traditional enumeration algorithms cannot be directly used with our pivot techniques, because our pivot techniques need to obtain a maximum $\eta$-clique in each recursion. In this subsection, we develop a pivot-based enumeration algorithm which fully integrates the proposed pivot techniques. The detailed implementation of our algorithm is shown in Algorithm 4.

In Algorithm 4, it first sorts all vertices in $V$ based on some ordering methods (line 1). Note that different ordering methods may affect the performance of the algorithm which will be discussed in Section 4.5. Let $\{v_1, \cdots, v_n\}$ be the ordering of vertices in $V$. Then, following the ordering, the algorithm processes vertices in the top recursion to enumerate maximal $(k, \eta)$-cliques by invoking the PMUCE procedure (lines 2-5). PMUCE admits 7 arguments: $R, r, X, C, P, k$, and $\eta$, where the parameter $P$ is used to maintain the maximum $\eta$-clique containing $R$ after the recursive call. The algorithm initializes the set of candidate vertices $C$ as the neighbors of $v_i$ which come after $v_i$, and the set $X$ of vertices as the remaining neighbor vertices of $v_i$ (lines 3-4), where $v_i$ is the $i$-th vertex in the ordered set $V$. Note that each vertex $u$ in $C$ and $X$ is associated with a probability that is the product of probabilities of edges connecting to the vertex $u$ and vertices in $R$. In each recursive call, the algorithm first checks whether $C \cup X$ is empty and the size of $R$ is no less than $k$; and if these conditions hold, the algorithm outputs the set $R$ as a maximal $(k, \eta)$-clique of $\mathcal{G}$ (lines 7-9). Otherwise, the algorithm continues to enumerate all $(k, \eta)$-cliques containing $R$ (lines 10-21).

In particular, the algorithm first initializes a variable $Q$ (line 10) which is used to store the periphery set in the current recursive call. Then, the algorithm selects a pivot vertex $u$ from $C$ (line 11). It starts to expand $R$ with $u$ and sets the larger $\eta$-clique to $P'$ (line 13). Before entering the next recursive call, the algorithm invokes Algorithm 2 to generate sets $C'$ and $X'$ (line 14). If there may be $(k, \eta)$-cliques containing $R'$, i.e., $|R'| + |C'| \geq k$, the algorithm continues the recursive calls (lines 15-16) and returns the maximum $(k, \eta)$-clique $P'$ that contains $R'$. If the size of the current periphery set $Q$ is smaller than $|P'|$, the algorithm revises the periphery set $Q$ by $P'$ based on Lemma 4 (line 17). Also, the set $P$ needs to be replaced by $P'$ if $|P| < |P'|$ (line 18). This is because $P'$ is the maximum $(k, \eta)$-clique containing $R$ found so far. Subsequently, the algorithm removes $u$ from $C$ and adds it into $X$ (line 19). The recursion terminates until all vertices in $C \setminus Q$ have been processed (line 20).

**Algorithm 4:** PMUC($\mathcal{G}, k, \eta$)

**Input:** An uncertain graph $\mathcal{G}$ and the parameters $k$ and $\eta$.
**Output:** All maximal $(k, \eta)$-cliques in $\mathcal{G}$.

1 Sorts the vertices in $V$ which are denoted by $\{v_1, \cdots, v_n\}$;
2 **for** $i = 1$ **to** $n$ **do**
3 $\quad C \leftarrow \{(v_j, p_{(v_i,v_j)}) | v_j \in N_{v_i}(\mathcal{G}), p_{(v_i,v_j)} \geq \eta, j > i\}$;
4 $\quad X \leftarrow \{(v_j, p_{(v_i,v_j)}) | v_j \in N_{v_i}(\mathcal{G}), p_{(v_i,v_j)} \geq \eta, j < i\}$;
5 $\quad$ PMUCE($\{v_i\}, 1, C, X, \{v_i\}, k, \eta$);
6 **Procedure:** PMUCE($R, q, C, X, P, k, \eta$)
7 **if** $C \cup X = \emptyset$ *and* $|R| \geq k$ **then**
8 $\quad$ Output $R$ as an maximal $(k, \eta)$-clique;
9 $\quad$ **return** $P$;
10 $Q \leftarrow \emptyset$;
11 Select a pivot $(u, r)$ in $C$;
12 **do**
13 $\quad R' \leftarrow R \cup \{u\}$; $P' \leftarrow R'$;
14 $\quad$ Generate $X'$ and $C'$ using Algorithm 2;
15 $\quad$ **if** $|R'| + |C'| \geq k$ **then**
16 $\quad\quad P' \leftarrow$ PMUCE($R', q * r, C', X', P', k, \eta$);
17 $\quad$ **if** $|Q| < |P'|$ **then** $Q \leftarrow P'$;
18 $\quad$ **if** $|P| < |P'|$ **then** $P \leftarrow P'$;
19 $\quad C \leftarrow C \setminus \{(u, r)\}$; $X \leftarrow X \cup \{(u, r)\}$;
20 **while** $\exists (u, r) \in C$ *s.t.* $u \notin Q$;
21 **return** $P$;

THEOREM 4.1. *Algorithm 4 correctly computes all maximal $(k, \eta)$-cliques of $\mathcal{G}$.*

PROOF. In each recursion of Algorithm 4, only vertices in $C \cap Q$ can be skipped to expand the current $\eta$-clique $R$ (line 20), where $Q$ is obtained by a maximum $(k, \eta)$-clique $P'$ that contains a pivot vertex $u \in C$ and $R$ (line 17). Note that although the set $Q$ may progressively increase as the pivot vertex $u$ changes (line 17), the skipped vertices are only determined by the final obtained set $Q$. Therefore, only the vertices in $C \cap Q \setminus \{u\}$ are skipped to expand $R$ in each recursion, because the pivot vertex $u \in C$ has been calculated before generating $Q$. Then, the theorem is established based on the results of Lemma 2, since $Q \cup R \setminus \{u\}$ does not contain any maximal $\eta$-clique in each recursive call. □

**Complexity analysis.** In the worst case, there are at most $2^n$ branches in Algorithm 4. Thus, the time complexity of Algorithm 4 is $O(n2^n)$ since each branch takes at most $O(n)$ operations. However, with the graph reduction techniques developed in Section 5.2, the time complexity of our algorithm can be further reduced to $O(n'2^{n'})$, where $n'$ is the number of vertices in the $(Top_k, \eta)$-triangle of $\mathcal{G}$ (detailed definition is given in Section 5.2). Since our graph reduction technique is more powerful than the state-of-the-art technique (see Lemma 10), the proposed algorithm is often much faster than the state-of-the-art algorithm [33] as confirmed in our experiments. Moreover, with our pivot-based pruning techniques, the practical performance of our algorithm can be more efficient, which is further demonstrated in our experiments. Since our algorithm works in a Deep First Search (DFS) manner, the total space overhead of our algorithm is linear to the graph size (i.e., $O(m + n)$).

**Discussions.** The pivot algorithms proposed in this paper are quite different from the classic pivot-based Bron-Kerbosch algorithms [7, 8, 52]. The main differences are twofold. First, the pivot techniques are different. In Section 3, we have analyzed that the pivot technique in classic Bron-Kerbosch algorithms cannot solve the problem of enumerating maximal cliques on uncertain graphs, and it is unclear whether there exists a pivot technique to solve this problem before our work. In this paper, we first discover a novel pivot technique for enumerating all maximal $(k, \eta)$-cliques based on a general pivot principle developed in Section 4.1. More specifically, such a pivot technique corresponds to skipping candidate vertices that are also included in a maximum $(k, \eta)$-clique containing the pivot vertex in each recursion. In addition, we observe that the proposed pivot technique can be progressively refined to further improve efficiency. Second, the DFS enumeration trees are also different. The key step of a clique enumeration algorithm is to determine the periphery set in each recursion. It can be seen that the periphery set can be immediately determined when the pivot vertex is obtained in classic pivot-based Bron-Kerbosch algorithms, which is just a subset of the candidate set that is adjacent to the pivot vertex. However, in our pivot algorithms, such a periphery set cannot be generated in a similar manner. This is because a maximum $(k, \eta)$-clique containing the pivot vertex can not be obtained in advance. To circumvent this issue, we first enumerate all maximal $(k, \eta)$-cliques containing the pivot vertex and make use of an additional argument $P$ to store the maximum $\eta$-clique detected so far. When obtaining all maximal $(k, \eta)$-cliques containing the pivot vertex, the set $P$ can be used as a periphery set.

## 4.5 Ordering Heuristics

Recall that the performance of Algorithm 4 may be related to the ordering of the vertices (see line 1 of Algorithm 4). Here we introduce two orderings to speed up the maximal $(k, \eta)$-clique enumeration.

**Degeneracy ordering.** A simple and efficient ordering is the degeneracy ordering [16] which is widely used in traditional maximal clique enumeration algorithms on deterministic graphs. Such an ordering can be obtained by repeatedly peeling a vertex of minimum degree in the remaining subgraph (the vertex removing ordering is just the degeneracy ordering). Such a peeling procedure can be done in $O(n+m)$ [3]. For an uncertain graph $\mathcal{G}$, we can obtain a degeneracy ordering on the deterministic graph of $\mathcal{G}$. Let $\delta$ be the degeneracy (i.e., the maximum $k$-core number) of the deterministic graph of $\mathcal{G}$. When using the degeneracy ordering, the algorithm can reduce the size of the largest candidate set from $d_{max}$ to $\delta$. As presented in [16], the degeneracy number $\delta$ is often much smaller than the maximum degree $d_{max}$ (also see Table 1). Thus, such an ordering heuristic can reduce the unnecessary computations in enumerating all maximal $(k, \eta)$-cliques.

**$(Top_k, \eta)$-core ordering.** Note that the degeneracy ordering completely ignores the probabilities of edges on $\mathcal{G}$. To further improve the enumeration performance, we adopt a $(Top_k, \eta)$-core concept introduced by Li et al. [33] to sort the vertices in $V$. Specifically, such an ordering can be generated by iteratively peeling the vertex with the minimum $\eta$-topdegree (detailed definition can be found in Section 5.2) in the remaining subgraph of $\mathcal{G}$. The time complexity for computing such an ordering is $O((n + m)\log d_{max})$ [33]. Based

on this ordering, it is easy to notice that the probabilities of edges in the subgraph induced by the candidate set are relatively high. Thus, there is a high opportunity to obtain large maximal $\eta$-cliques in the candidate set, thus improving the effectiveness of our pivot-based pruning techniques. Our experiments indicate that the $(Top_k, \eta)$-core ordering is better than the degeneracy ordering in enumerating all maximal $(k, \eta)$-cliques.

### 4.6 Pivot Selection Strategies

Recall that Algorithm 4 needs to select a pivot vertex $u$ which will be used to compute a maximum $\eta$-clique containing $R \cup \{u\}$. To achieve better pruning performance, we need to find a good pivot vertex based on which a large maximum $\eta$-clique can be found. Below, we propose several heuristic pivot selection strategies which work well in practice as confirmed in our experiments.

**Maximum degree based pivot selection.** Intuitively, the size of the maximum $\eta$-clique containing $R \cup \{v\}$ is related to the maximum degree of $v$. Therefore, a simple pivot selection strategy is to pick a vertex $v$ from $C$ with the maximum degree as the pivot, because the algorithm *is expected to* obtain a large maximum $\eta$-clique with the maximum degree vertex. We refer to such a method as a maximum degree based pivot selection strategy.

**Maximum color number based pivot selection.** The size of the maximum $\eta$-clique containing $R \cup \{v\}$ is also related to the color number of $v$ in $\mathcal{G}$. Here the color number is computed as follows. We first color the graph $\mathcal{G}$ using a classic greedy coloring algorithm [21] which can guarantee that two adjacent vertices are of the different color. Then, the color number of $v$ is the number of different colors that $v$'s neighbors have. Clearly, the color number of $v$ is an upper bound of the size of the maximum $\eta$-clique containing $R \cup \{v\}$, and it is no larger than the maximum degree. Therefore, we can select a pivot vertex from $C$ that has the largest color number. Intuitively, the algorithm with such a pivot can get a large maximum $\eta$-clique.

**Hybrid pivot selection strategy.** Let $\mathsf{PMUCE}(R, C, X)$ be a recursive call of the pivot-based enumeration algorithm. A simple intuition can be derived that for each vertex $v \in C$, the lower bound of the size of the maximum $\eta$-clique containing $v$ is $|R| + 1$. Based on such intuition, we denote by $\mathsf{LB}(v)$ the maximum lower bound size of the maximal $\eta$-clique containing $v$ for each $v$ in $C$. Formally, for each vertex $v \in C$, $\mathsf{LB}(v) = max\{\mathsf{LB}(v), |R| + 1\}$ in each recursive call. Note that $\mathsf{LB}(v)$ is a *global* concept which may be updated across the entire recursive enumeration procedure. Clearly, a vertex $v$ with a large $\mathsf{LB}(v)$ indicates that there is a large maximal $\eta$-clique containing $v$. We can use such a lower bound to obtain a hybrid pivot selection strategy. Specifically, the hybrid strategy first selects a pivot vertex $v$ with the maximum lower bound from the subset of $C$ that has the maximum color number. And then, it selects another pivot vertex $u$ with the maximum color number from the subset of $C$ that has the maximum degree. If $\mathsf{LB}(v)$ is larger than $k$, then we select $v$ as a final pivot, otherwise, we pick $u$ as a final pivot. Our experiments show that such a hybrid pivot selection strategy is better than the maximum degree and maximum color number based strategies.

## 5 FURTHER OPTIMIZATION TECHNIQUES

In this section, we develop two optimization techniques to further improve the efficiency of our enumeration algorithm. The first technique is based on the idea of the size constraint of $(k, \eta)$-clique, which can be used to prune the search branches in the recursive backtracking enumeration procedure. The second technique is a graph reduction technique which can remove the vertices from $\mathcal{G}$ that are definitely not included in any maximal $(k, \eta)$-clique.

### 5.1 Pruning Branches with Size Constraint

**The K-pivot technique.** Note that all maximal $(k, \eta)$-cliques must contain no less than $k$ vertices. We can use such a size constraint to construct a periphery set based on the result in Lemma 2. Specifically, in each recursive call $\mathsf{Recursion}(R, C, X)$, we can select a subset of $C$ as a periphery set $P$ that satisfies $|R| + |P| < k$. Note that such a periphery set $P$ only relies on the size constraint with a parameter $k$, and it does not need to select a pivot vertex to construct. For convenience, we refer to such a pivot technique as the K-pivot technique, and the periphery set $P$ as the K-pivot set. The following lemma formally describes the K-pivot technique.

*Lemma 5. Given an uncertain graph $\mathcal{G}$, all maximal $(k, \eta)$-cliques of $\mathcal{G}$ can be enumerated with the periphery set $P$ such that $|R| + |P| < k$ in each recursive call* $\mathsf{Recursion}(R, C, X)$*, where $P$ is an arbitrary subset of the candidate set $C$ with size less than $k - |R|$.*

Proof. Since $|R| + |P| < k$ in each $\mathsf{Recursion}(R, C, X)$, $R \cup P$ cannot contain any maximal $(k, \eta)$-clique. Thus, by Lemma 2, the algorithm can correctly enumerate all maximal $(k, \eta)$-cliques. □

**An improved K-pivot technique.** The K-pivot technique can be further improved using a color-based strategy. Specifically, we first color the uncertain graph $\mathcal{G}$ using a greedy coloring algorithm [21] such that two adjacent vertices are of the different color. The vertices with the same color form a color class. Instead of picking $k - |R| - 1$ vertices from $C$ to construct a K-pivot set, the improved K-pivot method selects $k - |R| - 1$ different color classes from $C$. All vertices in the selected color classes form the periphery set $P$. Let $cl(P)$ be the number of colors in $P$. Then, we have the following result.

*Lemma 6. Given an uncertain graph $\mathcal{G}$, all maximal $(k, \eta)$-cliques of $\mathcal{G}$ can be enumerated with the periphery set $P \subseteq C$ satisfying $cl(P) < k - |R|$ in each recursive call.*

Proof. Since the number of colors in $P$ is $cl(P)$, we can derive that the size of the maximum $\eta$-clique in the subgraph induced by $P$ can not exceed $cl(P)$. Then based on the fact that $cl(P) < k - |R|$ in each recursive call, we have that $R \cup P$ cannot contain any maximal $(k, \eta)$-clique. Thus, by Lemma 2, the algorithm can correctly enumerate all maximal $(k, \eta)$-cliques. □

Note that the improved K-pivot technique is clearly more effective than the K-pivot technique to reduce the search branches in the enumeration procedure, because the set of vertices with $k - |R| - 1$ colors must contain at least $k - |R| - 1$ vertices. Thus, in this paper, we only use the improved K-pivot technique in the proposed enumeration algorithm.

## 5.2 Graph Reduction Techniques

**The $(Top_k, \eta)$-triangle based reduction technique.** Here we develop a new graph reduction technique, called $(Top_k, \eta)$-triangle, to dramatically reduce the size of the input graph $\mathcal{G}$. This technique is mainly based on the fact that every edge in a maximal $(k, \eta)$-clique of $\mathcal{G}$ is associated with at least $k-2$ triangles that have the product of probabilities no less than $\eta$.

Let $\triangle_{uvw}$ be a triangle with three vertices $u, v, w \in V$, where $u, v, w$ connect with each other in $\mathcal{G}$. For an edge $e = (u, v) \in E$, we denote by $E_e(\mathcal{G})$ the set of triangles in $\mathcal{G}$ containing edge $e$, i.e., $E_e(\mathcal{G}) = \{\triangle_{uvw} | (u, w) \in E, (v, w) \in E\}$, where $e = (u, v) \in E$. For an edge $(u, v)$ and a triangle $\triangle_{uvw}$, the *open triangle probability* denoted by $Pr_{uv}(\triangle_{uvw}, \mathcal{G})$ is defined as the product of $p_{(u,w)}$ and $p_{(v,w)}$, i.e. $Pr_{uv}(\triangle_{uvw}, \mathcal{G}) = p_{(u,w)}p_{(v,w)}$. If the context is clear, we will omit the subscript $uv$ in $Pr_{uv}(\triangle_{uvw}, \mathcal{G})$. Suppose without loss generality that for any $e = (u, v) \in E$, the triangles in $E_e(\mathcal{G}) = \{\triangle_{uvw_1}, \triangle_{uvw_2}, \cdots, \triangle_{uvw_h}\}$ are sorted in a non-increasing order of their open triangle probabilities $Pr_{uv}(\triangle_{uvw}, \mathcal{G})$. The following definition gives the *top triangle degree* of an edge $e = (u, v)$ in $\mathcal{G}$.

*Definition 5 (Top triangle degree). Given an uncertain graph $\mathcal{G}$ and a probability threshold $\eta \in [0, 1]$, the top triangle degree of $e = (u, v) \in E$ in $\mathcal{G}$, denoted by $t_\eta(e, \mathcal{G})$, is the maximum number $k$ such that the product of $p_e$ and the top-$k$ open triangle probability of $e$ in $\mathcal{G}$ is no less than $\eta$.*

Denote by $E_e^k(\mathcal{G})$ the triangle set of $e \in E$ in $E_e(\mathcal{G})$ with the top-$k$ highest open triangle probabilities. Then, the following definition introduces a concept of top-$k$ open triangle probability of $e$ in $\mathcal{G}$.

*Definition 6 (Top-k open triangle probability). Given an uncertain graph $\mathcal{G}$ and an integer $k$, the top-$k$ open triangle probability of $e = (u, v)$ is defined as $Pr(E_e^k(\mathcal{G})) = \prod_{\triangle_{uvw} \in E_e^k(\mathcal{G})} Pr(\triangle_{uvw}, \mathcal{G})$.*

By Definition 5 and Definition 6, for an $e \in E$, the top triangle degree of $e$ in $\mathcal{G}$ can be computed by

$$t_\eta(e, \mathcal{G}) = \max\{k \in [0, h] | p_e Pr(E_e^k(\mathcal{G})) \geq \eta\}, \quad (3)$$

where $h$ is the cardinality of $E_e(\mathcal{G})$, i.e., $h = |E_e(\mathcal{G})|$.

We can easily derive that the top triangle degree of each $e$ in $\mathcal{G}$ follows a monotonic property.

*Lemma 7. Given two subgraphs $C_1$ and $C_2$ of $\mathcal{G}$ that contain an edge $e$, we always have $t_\eta(e, C_1) \leq t_\eta(e, C_2)$ if $C_1 \subseteq C_2$.*

PROOF. Let $E'$ be the set of edges excluded from $C_2$ by $C_1$. If we add the edges in $E'$ to $C_1$, we can notice that in the worst case, $E_e^k(C_1)$ is still the triangle set of $e$ with the top-$k$ highest open triangle probabilities, which indicates that the top-$k$ triangle probability of $e$ in the new subgraph (corresponding to $C_2$) will not decrease. Thus, we have $t_\eta(e, C_1) \leq t_\eta(e, C_2)$ if $C_1 \subseteq C_2$. □

Based on Definition 5 and Definition 6, we introduce the concept of $(Top_k, \eta)$-triangle as follows.

*Definition 7 ($(Top_k, \eta)$-triangle). Given an uncertain graph $\mathcal{G}$, a probability threshold $\eta \in [0, 1]$, and a positive integer $k \in \mathbb{N}^+$, a subgraph $C = (V_C, E_C, p)$ of $\mathcal{G}$ is a $(Top_k, \eta)$-triangle if every edge $e$ in $E_C$ satisfies $t_\eta(e, C) \geq k$.*

**Algorithm 5:** TopKTrianglePruning($\mathcal{G}, k, \eta$)

**Input:** An uncertain graph $\mathcal{G}$ and the parameters $k$ and $\eta$.
**Output:** The maximal $(Top_k, \eta)$-triangle.

```
1  Q ← ∅;
2  foreach (u, v) ∈ E do
3      Compute t_η(u, v, G); ▷ Eq. (3)
4      if t_k(u, v, G) < k then
5          Q ← Q ∪ {(u, v)};
6  foreach (u, v) ∈ Q do
7      Remove (u, v) from Q and G;
8      foreach w ∈ N_v(G) ∩ N_u(G) do
9          if t_η(w, v, G) ≥ k then
10             Update t_η(w, v, G);
11             if t_η(w, v, G) < k then Q ← Q ∪ {(w, v)} ;
12         if t_η(w, u, G) ≥ k then
13             Update t_η(w, u, G);
14             if t_η(w, u, G) < k then Q ← Q ∪ {(w, u)} ;
15 C ← the subgraph induced by remaining edges of G;
16 return C;
```

A $(Top_k, \eta)$-triangle $C$ is maximal if there is no other subgraph $C' \supset C$ of $\mathcal{G}$ such that $C'$ is a $(Top_k, \eta)$-triangle. The possible triangle number of an edge $e \in E$, called $s_\eta(e)$, is the maximum integer $k$ such that there is a $(Top_k, \eta)$-triangle of $\mathcal{G}$ containing $e$.

*Lemma 8. Given an uncertain graph $\mathcal{G}$, all maximal $(k+2, \eta)$-cliques of $\mathcal{G}$ are contained in the maximal $(Top_k, \eta)$-triangle of $\mathcal{G}$ where $\eta \in [0, 1]$ and $k \in \mathbb{N}^+$.*

PROOF. The lemma clearly holds, because any maximal $(k+2, \eta)$-clique of $\mathcal{G}$ must be a $(Top_k, \eta)$-triangle of $\mathcal{G}$ (by Definition 7). □

**Computing the $(Top_k, \eta)$-triangle.** Based on the monotonic property of the top triangle degree (Lemma 7), the maximal $(Top_k, \eta)$-triangle can be computed by iteratively removing edges of $\mathcal{G}$ with the top triangle degree less than $k$. Algorithm 5 shows the detailed procedure for computing the maximal $(Top_k, \eta)$-triangle in $\mathcal{G}$. In particular, Algorithm 5 first computes the top triangle degree for each edge in $E$ using Eq. (3) (lines 2-3). Then, the algorithm pushes the edges whose top triangle degrees are less than $k$ into a queue $Q$ (lines 2-5). Next, the algorithm iteratively removes the edges in $Q$ (lines 6-14). When removing an edge $e$, the algorithm updates the top triangle degree for each remaining edge in $\mathcal{G}$ associated with $e$ by a triangle (lines 10 and 13); and it pushes the edges whose updated top triangle degrees are less than $k$ into $Q$ (lines 11 and 14). Note that, in lines 10 and 13, we can update the top triangle degree of $e \in E$ by examining whether the removed edge is included in $E_e^k(\mathcal{G})$ with an additional index array in constant time. The algorithm terminates until the queue $Q$ is empty.

*Lemma 9. The time and space complexity of Algorithm 5 are $O(m^{1.5} \log(d_{\max}))$ and $O(m^{1.5})$ respectively.*

PROOF. For the time complexity, Algorithm 5 first computes the top triangle degree for each edge in $E$, which takes $O(\sum_{(u,v)\in E}(d_v + d_u)\log(d_{\max})) \leq O(m^{1.5} \log(d_{\max}))$ time in total (lines 2-5). Moreover, only $O(m^{1.5})$ time is spent to update the top triangle degrees of

the edges in $\mathcal{G}$ (lines 6-14), since the time cost for updating the top triangle degree for each edge can be achieved in constant time with an additional index array. Thus, the time complexity of Algorithm 5 is $O(m^{1.5}\log(d_{\max}))$. For the space complexity, Algorithm 5 uses $O(m^{1.5})$ space to store all triangles of the input graph; and it needs several linear arrays to maintain the top triangle degree for each edge. As a result, the space complexity of Algorithm 5 is $O(m^{1.5})$. □

**The $(Top_k, \eta)$-core based reduction technique.** In [33], Li et al. proposed a $(Top_k, \eta)$-core concept to prune the vertices of $\mathcal{G}$ with the time complexity $O((m+n)\log(d_{\max}))$. Below, we first briefly introduce the concept of $(Top_k, \eta)$-core, since it can be used as a preprocessing method for the proposed $(Top_k, \eta)$-triangle based reduction technique.

Denote by $E_v^k(\mathcal{G})$ the set of neighbors of $v$ with the top-$k$ largest edge probabilities. The $\eta$-topdegree of a vertex $v$ is defined as

$$\eta\text{-topdegree}_v(\mathcal{G}) = max\{k| \prod_{e\in E_v^k(\mathcal{G})} p_e \geq \eta\}. \quad (4)$$

Based on the concept of $\eta$-topdegree, the $(Top_k, \eta)$-core [33] is defined as follows.

*Definition 8. Given an uncertain graph $\mathcal{G}$, and two parameters $k$ and $\eta$, a subgraph $C = (V_C, E_C, p)$ of $\mathcal{G}$ is a $(Top_k, \eta)$-core if every vertex $v \in V_C$ has a $\eta$-topdegree no less than $k$ in $C$.*

Below, we show a connection between the $(Top_k, \eta)$-core and the proposed $(Top_k, \eta)$-triangle.

*Lemma 10. Given an uncertain graph $\mathcal{G}$, any $(Top_k, \eta)$-triangle $C$ of $\mathcal{G}$ must be a $(Top_{k+1}, \eta)$-core of $\mathcal{G}$.*

Proof. It is easy to see that for each $e = (u, v)$ in $C$, the $\eta$-topdegrees of $u$ and $v$ on $e$ are no less than $k+1$ based on Eq. (3-4). Thus, the subgraph $C$ is also a $(Top_{k+1}, \eta)$-core of $\mathcal{G}$. □

By Lemma 10, we can see that the pruning performance of the proposed $(Top_k, \eta)$-triangle technique is better than that of the $(Top_k, \eta)$-core technique. However, the worst case time complexity for computing the $(Top_k, \eta)$-triangle is higher than computing the $(Top_k, \eta)$-core. Thus, to achieve a good pruning performance, we can first apply the $(Top_k, \eta)$-core based reduction technique to prune vertices in $\mathcal{G}$, and then we apply the proposed $(Top_k, \eta)$-triangle technique to further remove unpromising vertices in the $(Top_k, \eta)$-core. Our experimental results suggest that such a method can achieve a very good performance in practice.

## 6 EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the proposed algorithms. Below, we first introduce the experimental setup and then report our results.

### 6.1 Experimental Setup

We implement three algorithms MUC, PMUC, and PMUC+ to enumerate all maximal $(k, \eta)$-cliques. MUC is the state-of-the-art algorithm proposed in [33] which invokes a $(Top_k, \eta)$-core pruning technique to remove unpromising vertices before the recursive backtracking enumeration procedure. PMUC is our pivot-based algorithm which integrates all pruning techniques developed in Section 4. Note that PMUC is implemented based on the improved M-pivot technique, because we find that the M-pivot method is typically worse than the improved M-pivot technique. In addition, we also integrate the $(Top_k, \eta)$-core pruning technique in PMUC, as used in MUC, for a fair comparison. PMUC+ is an improved version of PMUC, which includes all the pruning techniques proposed in both Section 4 and Section 5. Note that both PMUC and PMUC+ are equipped with the $(Top_k, \eta)$-core ordering and a lower-bound based hybrid pivot selection strategy, which typically shows a better performance compared to the other strategies. All algorithms are implemented in C++. All experiments are conducted on a PC with one 2.2 GHz AMD CPU and 128GB memory running CentOS 7.6.

**Table 1: Datasets**

| Type | Dataset | $\|V\|$ | $\|E\|$ | $d_{\max}$ | $\delta$ |
|---|---|---|---|---|---|
| Semi-real uncertain graphs | Enron | 87,273 | 1,148,072 | 38,785 | 53 |
| | SuperUser | 194,085 | 1,443,339 | 27,637 | 61 |
| | CaHepPh | 28,093 | 4,596,803 | 11,134 | 410 |
| | Wiki-fr | 1,420,367 | 4,641,928 | 1,096,752 | 120 |
| | Soflow | 2,601,977 | 63,497,050 | 194,806 | 198 |
| Real uncertain graphs | CORE | 2,708 | 7,123 | 141 | 15 |
| | NL27K | 27,221 | 175,412 | 4157 | 20 |
| | CN15K | 15,000 | 241,158 | 1252 | 33 |
| | DBLP | 474,131 | 2,715,562 | 279 | 93 |

**Table 2: Running time of various algorithms on real uncertain graphs with default parameter settings (in milliseconds)**

| Dataset | PMUC+ | PMUCE | MUCE |
|---|---|---|---|
| CORE | 0.225 | 0.319 | 0.612 |
| NL27K | 2.353 | 9.279 | 36.343 |
| CN15K | 0.074 | 2.457 | 6.836 |
| DBLP | 30.062 | 38.939 | 61.581 |

**Datasets.** We make use of five *semi-real* uncertain graphs and four real uncertain graphs to evaluate the performance of different algorithms. Table 1 provides the detailed statistical information of each dataset, where the last two columns show the maximum degree and the degeneracy of each dataset respectively. Enron is a weighted email interaction network, where the weight for each edge denotes the number of emails sent by two users. SuperUser, Stackoverflow (Soflow for short), and Wiki-fr are weighted communication networks between online users, in which the weight of each edge denotes the number of interactions between two users. CaHepPh is a weighted scientific collaboration network where the weight on an edge represents the number of papers coauthored by two scholars. CORE is a real protein-protein interaction (PPI) network provided by Krogan et al. [31], where the probability of each edge represents the confidence of the interaction between proteins. CN15K and NL27K are the real-world uncertain knowledge graphs [10, 11] extracted from ConceptNet [51] and NELL [41] respectively, where the probability of each relation fact in these graphs represents the confidence score in the interval [0.1, 1.0]. DBLP is an uncertain graph in which the edges are inferred by the standard LDA model [4] on DBLP collaboration network as used in [6, 22]. The probability on each edge in DBLP represents the likelihood between two vertices related to a specific topic task. The first five datasets are downloaded from (http://konect.cc/), and the remaining datasets are provided by their

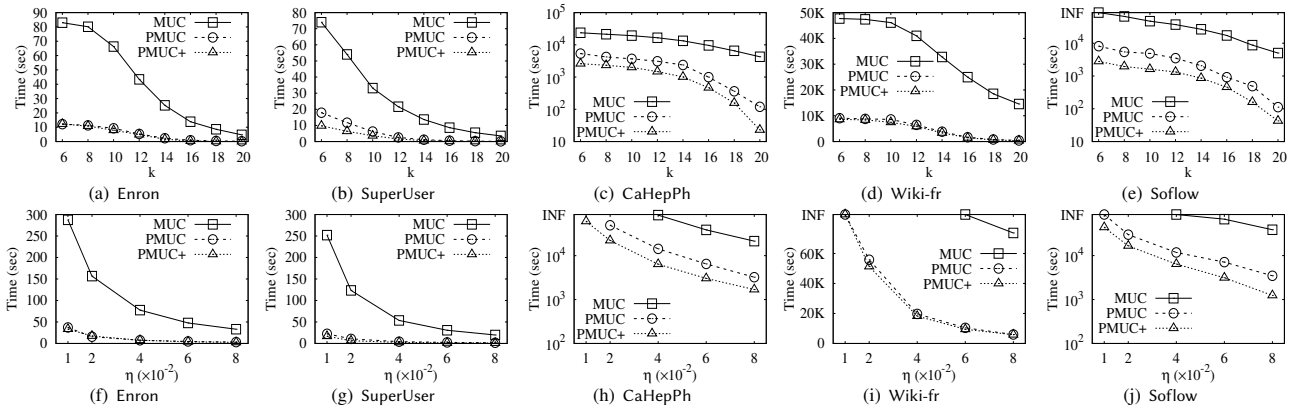



**Figure 3: Runtime of different algorithms with varying $k$ and $\eta$**

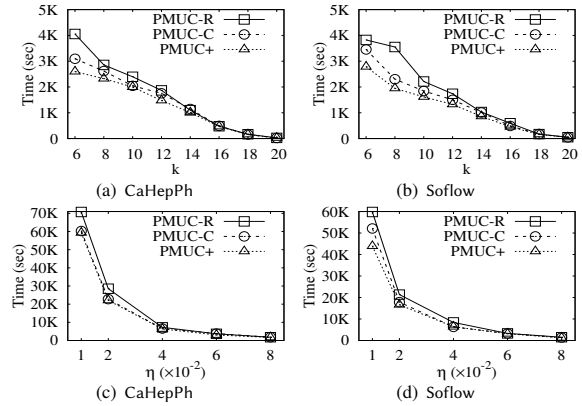



**Figure 4: Runtime of pivot-based algorithms with different ordering techniques**

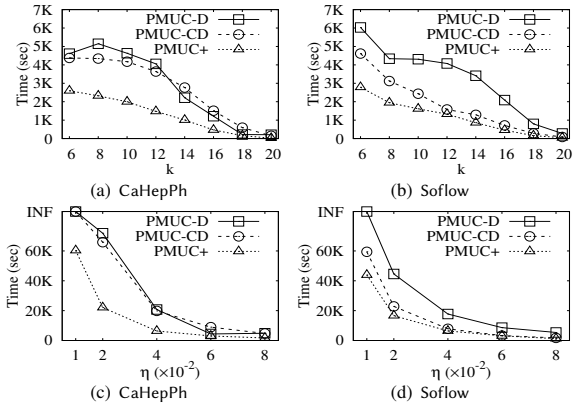



**Figure 5: Runtime of pivot-based algorithms with different pivot selection strategies**

original authors. For the first five datasets, we adopt a widely-used method in the uncertain graph mining literature [33, 34, 46, 47] to generate semi-real uncertain graphs. Specifically, for each dataset, we make use of an exponential cumulative distribution function ($f(w) = 1 - e^{-w/2}$) to generate the probability of each edge, where $w$ denotes the weight of the edge. We will also study the effect of different probability distributions in Exp-5.

**Parameters.** In all algorithms, there are two parameters: $k$ and $\eta$. We select $k$ from the interval $[6, 20]$ with a default value of 14. The parameter $\eta$ is chosen from the interval $[0.01, 0.1]$ with a default value of 0.1. Unless otherwise specified, the value of the other parameter is set to its default value when varying a parameter.

## 6.2 Efficiency Results

**Exp-1: Runtime of different algorithms.** In this experiment, we evaluate the running time of MUC, PMUC, and PMUC+ to enumerate all maximal $(k, \eta)$-cliques. Fig. 3 shows the runtime of each algorithm on all semi-real datasets with varying $k$ and $\eta$, respectively. Note that if an algorithm cannot terminate within 24 hours, its runtime is set to "INF". As can be seen, PMUC+ consistently outperforms PMUC and MUC under all parameter settings. More specifically, PMUC+ can achieve 1× faster than PMUC on large datasets with most parameter settings; and it can be up to two orders of magnitude faster than MUC on the large datasets. For instance, on Soflow (Fig. 3(e)), when $k = 20$ and $\eta = 0.1$, PMUC+ only takes 42.68 seconds to compute all maximal $(k, \eta)$-cliques, while PMUC and MUC take 2.74× and 117.31× more time than PMUC+, respectively. These results demonstrate the powerful performance of our pivot-based algorithms for enumerating all maximal $(k, \eta)$-cliques. Moreover, on the large datasets, PMUC+ (or PMUC) can achieve a better speedup ratio w.r.t. MUC for a larger $k$. This is because the pruning performance of our pivot techniques becomes more effective as $k$ grows. As expected, the runtime of all algorithms decreases when $k$ or $\eta$ increases, because the number of maximal $(k, \eta)$-cliques decreases with an increasing $k$ or $\eta$. In addition, it is worth noting that the runtime of PMUC and PMUC+ is very close on some datasets as shown in Fig. 3, although PMUC is not equipped with the optimization techniques developed in Section 5. These results indicate that our improved M-pivot technique is very effective in pruning unnecessary vertices.

We also evaluate the performance of various algorithms on real uncertain graphs with default parameter settings. As shown in Table 2, all algorithms run very fast on all real datasets, because all these real uncertain graphs are small and often do not contain large uncertain cliques. Also, we can see that the proposed algorithm PMUC+ consistently achieves the best performance, which further confirms the high efficiency of the proposed techniques. Below, we will mainly use the real uncertain graphs to evaluate the effectiveness

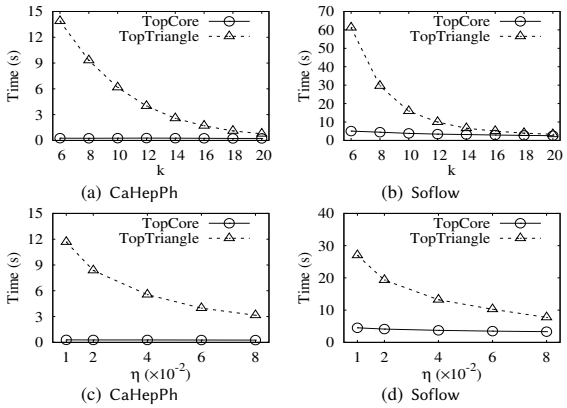



**Figure 6: Runtime of different graph reduction techniques**

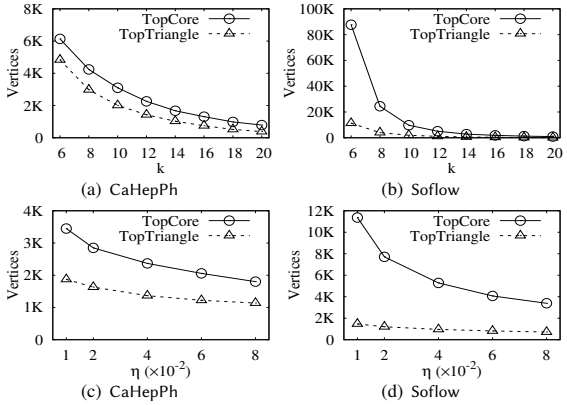



**Figure 7: The number of remaining vertices obtained by different graph reduction techniques**

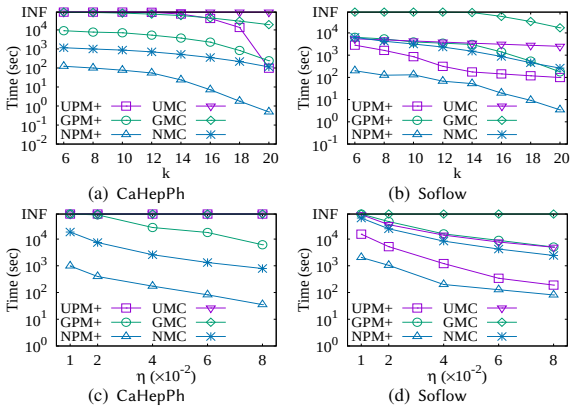



**Figure 8: Efficiency of various algorithms on uncertain graphs with different probability distributions**

of the proposed algorithms and make use of the semi-real uncertain graphs to test the efficiency.

**Exp-2: Effect of different ordering techniques.** In this experiment, we test the efficiency of our pivot-based algorithms with different ordering techniques proposed in section 4.5. Let PMUC-R, PMUC-C, and PMUC+ be pivot-based algorithms based on vertices as-is ordering, degeneracy ordering, and $(Top_k, \eta)$-core ordering, respectively, where the time complexities of the last two orderings are $O(m + n)$

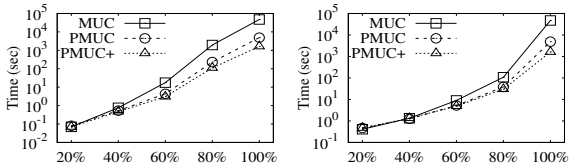



**Figure 9: Scalability of different algorithms on** Soflow ($k = 10$ **and** $\eta = 0.1$)

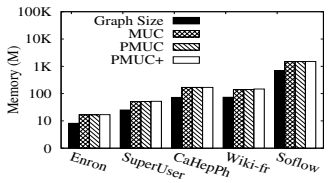



**Figure 10: Memory usages of different algorithms**

and $O((n + m)\log d_{\max})$, respectively. Note that except for different ordering techniques, all these algorithms are equipped with the same enumeration technique proposed in Section 4 and Section 5. Fig. 4 shows the runtime of different algorithms on CaHepPh and Soflow with varying $k$ and $\eta$, respectively. The results on the other datasets are consistent. From Fig. 4, we observe that the time overhead of PMUC-C and PMUC+ are consistently lower than those of PMUC-R with different parameter settings. For instance, on Soflow, when setting $k = 12$ and $\eta = 0.1$, PMUC-C and PMUC+ consume 1501.79 seconds and 1324.96 seconds to compute all maximal $(k, \eta)$-cliques respectively while PMUC-R takes 1737.29 seconds to complete the computation. Moreover, we can see that PMUC+ is typically faster than PMUC-C with varying $k$ and $\eta$, indicating that the $(Top_k, \eta)$-core ordering is better than the degeneracy ordering in enumerating all maximal $(k, \eta)$-cliques.

**Exp-3: Effect of different pivot selection strategies.** In this experiment, we evaluate the performance of three pivot selection strategies: PMUC-D, PMUC-CD, and PMUC+, where PMUC-D is the maximum degree based strategy, PMUC-CD is a maximum color number based strategy, and PMUC+ is the hybrid pivot selection strategy proposed in Section 4.6, respectively. Fig 5 shows the runtime of three different algorithms on CaHepPh and Soflow. The results on the other datasets are consistent. As can be seen, PMUC+ achieves the best performance among these three algorithms with varying $k$ and $\eta$, while PMUC-D exhibits the worst performance in enumerating all maximal $(k, \eta)$-cliques. A notable example is when setting $k = 12$ and $\eta = 0.1$, PMUC+ and PMUC-CD take 1324.96 seconds and 1579.58 seconds to enumerate all maximal $(k, \eta)$-cliques on Soflow respectively. However, PMUC-D consumes 4074.9 seconds to complete the computations. We also notice that the performance of PMUC+ is more robust compared to the other two pivot selection strategies. These results suggest that the proposed hybrid pivot selection strategy is indeed very efficient in enumerating all maximal $(k, \eta)$-cliques.

**Exp-4: Comparison of graph reduction techniques.** Denote by TopTriangle the $(Top_k, \eta)$-triangle graph reduction technique proposed in Section 5.2, and TopCore be the $(Top_k, \eta)$-core graph reduction technique proposed in [33]. Fig. 6 shows the runtime of

**Table 3: Precision of various algorithms on CORE.**

| Algorithm | #Results | TP | FP | PR |
|---|---|---|---|---|
| USCAN [47] | 456 | 1086 | 2037 | 0.348 |
| PCluser [30] | 475 | 1027 | 3021 | 0.266 |
| UKCore [6] | 3 | 418 | 151 | 0.318 |
| UKTruss [22] | 3 | 266 | 134 | 0.374 |
| PMUCE | 1505 | 61971 | 16355 | **0.791** |

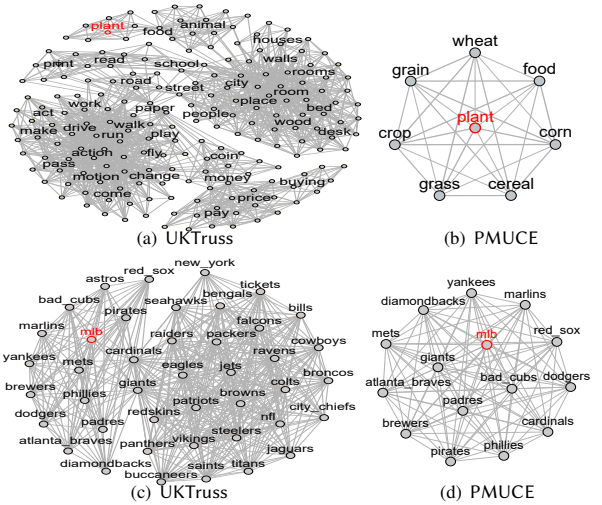
TopTriangle and TopCore on CaHepPh and Soflow; and Fig. 7 reports the number of remaining vertices obtained by these two graph reduction techniques on CaHepPh and Soflow. Similar results can be observed on the other datasets.

From Fig. 6, we can see that the time overhead of TopCore is very stable with varying $k$ and $\eta$ on different datasets. The runtime of TopTriangle, however, increases as $k$ and $\eta$ decrease. The reasons are as follows. The runtime of TopCore is dominated by the initialization phase for computing the $\eta$-topdegrees of all vertices which is insensitive w.r.t. $k$ and $\eta$. TopTriangle works on the $(Top_k, \eta)$-core subgraph, whose size increases as $k$ decreases (or $\eta$). Thus, the time overhead of TopTriangle increases as $k$ and $\eta$ decrease. However, when comparing the total time overhead for enumerating all maximal $(k, \eta)$-cliques using different graph reduction techniques (see Exp-1), we can observe that the pivot-based algorithm equipped with TopTriangle is much faster than that of the algorithm with TopCore.

Fig. 7 shows the graph reduction performance of TopCore and TopTriangle. As can be seen, TopTriangle consistently outperforms TopCore for pruning vertices of $\mathcal{G}$ with varying parameters of $k$ and $\eta$, because the structure of $(Top_k, \eta)$-triangle is more cohesive than $(Top_k, \eta)$-core (Lemma 10). This result confirms that the proposed $(Top_k, \eta)$-triangle technique can achieve better pruning performance than the state-of-the-art TopCore technique.

**Exp-5: The effect of different probability distributions.** Here we evaluate the performance of different algorithms on semi-real uncertain graphs generated from uniform, geometry, and normal distributions, where the probability of each edge is generated by uniformly selecting a value from [0.5,1], by a function $f(w) = \sum_{i=1}^{w}(1-p)^w p$ with $p = 0.2$, and by $f(w) = \frac{1}{2}(1 + erf(\frac{w-\mu}{\sigma}))$ with $\mu = 5$ and $\sigma = 8$, respectively. The results are shown in Fig. 8. Note that in Fig. 8, *UMC*, *GMC*, and *NMC* denote MUCE on the datasets with edge probabilities generated by uniform, geometry, or normal distributions, respectively. Similarly, *UPM+*, *GPM+*, and *NPM+* represent PMUC+ on uncertain graphs with edge probabilities generated by uniform, geometry, or normal probability distributions, respectively. As can be seen, the time overheads of PMUC+ are consistently lower than that of MUCE on all uncertain graphs with different probability distributions. Moreover, PMUC+ is one order of magnitude faster than MUCE within most parameter settings. These results are consistent with our previous results, which further suggest that the performance of our algorithm is insensitive to different types of uncertain graphs.

**Exp-6: Scalability testing.** In this experiment, we make use of the largest dataset Soflow to test the scalability of the proposed pivot-based algorithms. In particular, we generate four subgraphs by randomly sampling 20-80% vertices (edges) from Soflow; and evaluate the runtime of each algorithm on these subgraphs. Fig. 9 shows the scalability results of PMUC+, PMUC, and MUC on Soflow with




(a) UKTruss (b) PMUCE (c) UKTruss (d) PMUCE

**Figure 11: Community search on real-world uncertain knowledge graphs: (a) and (b) are the results on CN15K with a query {"*plant*"}; (c) and (d) are the results on NL27K with a query {"*mlb*"}.**

$k = 10$ and $\eta = 0.1$. We can see that the time overheads of all algorithms increase smoothly with the increase of $|V|$ and $|E|$. Once again, we can observe that the runtime of our pivot-based algorithms (PMUC+ and PMUC) are significantly lower than those of the state-of-the-art algorithm (MUC). These results indicate that our algorithms exhibit better scalability performance than MUC in enumerating all maximal $(k, \eta)$-cliques.

**Exp-7: Memory overhead.** Fig. 10 shows the memory overheads of PMUC+, PMUC, and MUC. From Fig. 10, we can see that the memory usages of all algorithms are slightly larger than the graph size (most of them are twice the graph size). Moreover, even for PMUC+, which uses the $(Top_k, \eta)$-triangle graph reduction technique, it has roughly the same memory usage as PMUC. This is because the $(Top_k, \eta)$-triangle graph reduction technique is computed in the subgraph induced by $(Top_k, \eta)$-core, which is often much smaller than the original graph. These results indicate that the proposed pivot-based algorithms are highly space-efficient.

## 6.3 Case Studies

**Exp-8: Clustering quality on PPI networks.** Here we evaluate the clustering quality of maximal $(k, \eta)$-cliques on protein-protein interaction (PPI) network CORE. According to [47], the ground truth results of protein interactions can be determined by the MIPS protein database [30]. Then, we can evaluate the clustering quality of various algorithms by computing the number of true positive protein interactions (TP), the number of false positive protein interactions (FP), and the precision result (PR=TP/(FP+TP)) obtained by the algorithms. We make use of four state-of-the-art algorithms, USCAN [47], PCluser [30], UKCore [6], and UKTruss [22], as baselines to compare the effectiveness with our algorithm. For USCAN and PCluser, we run them with the default parameter settings as used in their experiments. For the other algorithms, we set $\eta = 0.1$, and then make use of PMUCE, UKCore, and UKTruss to enumerate all maximal $(10, 0.1)$-cliques, $(9, 0.1)$-cores, and local $(10, 0.1)$-trusses,

**Table 4: Task-driven team formation queries and results.**

| $Q$ = {"*Jiawei Han*"} | |
|---|---|
| $T$ = {"*Databases*"} | $T$ = {"*Information Networks*"} |
| Qiming Chen, **Jiawei Han**, Jianyong Wang, Meichun Hsu, Jian Pei, Umeshwar Dayal, Behzad Mortazavi-Asl | Heng Ji, **Jiawei Han**, Yizhou Sun, Chi Wang, Jialu Liu, Xiang Ren, Brandon Norick |

respectively. Table 3 shows the results of various algorithms. As can be seen, PMUCE achieves the best clustering quality among all algorithms. In particular, the precisions of all baselines are lower than 0.375, while PMUCE can achieve a precision of 0.791. The reason for this is that proteins usually interact strongly in a small area which cannot be well characterized by the graph clustering methods (USCAN [47] and PCluser [30]), and by $k$-core (UKCore) or $k$-truss (UKTruss) based cohesive subgraph models, because all of them often obtain relatively large clusters. These results indicate that our algorithms are very effective in applications of detecting protein complexes in PPI networks.

**Exp-9: Community search on uncertain knowledge graphs.** We evaluate the community search performance of our algorithm on two real-world uncertain knowledge graphs CN15K and NL27K [10, 11]. For comparison, we apply UKCore and UKTruss to detect communities, as these two algorithms are the state-of-the-art for community search applications on uncertain graphs [6, 22]. Fig. 11 shows the community search results of different algorithms on CN15K and NL27K. In particular, Fig. 11(a-b) are the results on CN15K with $\eta = 0.001$ and a query $Q$ = {"*plant*"}; Fig. 11(c-d) are the results on NL27K with $\eta = 0.1$ and a query $Q$ = {"*mlb*"}. Note that the communities obtained by UKCore on these two uncertain graphs are very large, thus we did not visualize them. On CN15K, we can see that the community obtained by UKTruss has 200 vertices and 1541 edges with a diameter of 7. A large number of words in this community are not related to the word "*plant*", which is clearly not what we expected. While in Fig. 11(b), the community obtained by our algorithm is the plant-based foods, which is more likely to be a community. In addition, on NL27K, we can see that the community found by UKTruss not only contains the Major League Baseball (MLB) sports teams but also the National Football League (NFL) sports teams, while the result obtained by our solution only contains the MLB sports teams. These results demonstrate the superiority of our algorithm in community search on uncertain graphs.

**Exp-10: Task-driven team formation.** In this experiment, we apply our maximal $(k, \eta)$-clique algorithm to solve the task-driven team formation problem on DBLP [6, 22]. Let $\mathcal{G}^T$ be the uncertain graph obtained by the specific topic task $T$ [6]. Given a query pair $\langle T, Q\rangle$ with a task $T$ and a vertex set $Q \subset V$, and a probability threshold $\eta$, the problem of task-driven team formation is to find a good team $A$ from $\mathcal{G}^T$, with $Q \subset A$, which maximizes a notion of density. For comparison, we also use UKCore and UKTruss as baselines; and set the parameter $\eta = 10^{-10}$ (as the probabilities of the edges in DBLP are small). We take two queries $\langle$"*Databases*", "*Jiawei Han*"$\rangle$ and $\langle$"*Information Networks*", "*Jiawei Han*"$\rangle$ as examples. For the first query, UKCore obtains a team that has 13,473 nodes and 120,049 edges, and UKTruss gets a team with 10,546 nodes and 74,875 edges. Since the results are too large, we do not visualize the teams founded by these methods. Compared to UKCore and UKTruss, which often contain many irrelevant vertices in the team, our solution can obtain a more compact team which only contains 7 vertices (highly relevant to the query) as shown in Table 4. The results for the second query are consistent. Moreover, we can see that the obtained teams by our solution are quite different when we use the same query vertex "*Jiawei Han*" to detect teams with respect to two topic tasks (Table 4). This result indicates that our algorithm performs well in the application of task-driven team formation.

## 7 RELATED WORKS

**Maximal clique enumeration.** Our work is closely related to the traditional maximal clique enumeration problem on deterministic graphs. The well-known solutions for enumerating all maximal cliques are the classic Bron-Kerbosch algorithm [7] and its pivot-based variants [17, 44, 52]. Tomita et al. [52] shown that the time overhead to list all maximal cliques is optimal in the worst case using their pivot technique. Eppstein et al. [17] further derived a tighter worst-case time complexity for enumerating all maximal cliques based on a degeneracy ordering technique [35]. Naudé et al. [44] revised the pivot algorithm by refined the pivot selection process to speedup the computations. Additionally, a class of output-sensitive algorithms has also been developed for the problem of maximal clique enumeration [9, 13, 37], which are bounded with a time delay between reported cliques. Among those algorithms, the state-of-the-art was obtained by Conte et al. [13], which can achieve the sub-linear space complexity with bounded time delay. Although those output-sensitive algorithms are theoretically efficient, their practical performance is often worse than the pivot-based algorithms as reported in [13]. More recently, the maximal clique enumeration problem was also studied on some particular types of graphs. For example, Viard et al. [54] studied the problem of enumerating all maximal cliques in temporal graphs, in which each edge is associated with a timestamp. Li et al. [32] proposed a concept of signed clique in signed graphs; and they also developed an efficient branch and bound algorithm to enumerate all maximal signed cliques. Chen et al. [12] further studied the problem of enumerating all balanced cliques in signed graphs. Unfortunately, all aforementioned techniques cannot be used to enumerate maximal $(k, \eta)$-cliques on uncertain graphs.

**Uncertain graph mining.** Mining uncertain graphs is a fundamental research topic in graph analysis. Except for maximal $(k, \eta)$-clique enumeration, there are many uncertain graph mining tasks that are related to our work. Notable examples for uncertain graph mining include identifying reliable connected subgraphs [26, 27], computing shortest paths or nearest neighbors [18, 46, 49], performing influence analysis [39], and clustering or finding cohesive subgraphs in uncertain graphs [6, 22, 47, 55]. More specifically, Khan et al. [27] studied the problem of detecting reliable vertices in uncertain graphs. Ke et al. [26] provided guidelines for researches and applications on the problem of $s$-$t$ reliability estimation by extensive experimental evaluations. Saha et al. [49] studied the problem of computing shortest paths in uncertain networks. Potamias et al. [46] and Gao et al. [18] investigated the problem of $k$-nearest neighbors over uncertain graphs. Mehmood et al. [39] studied an influence cascade problem which aims to find a set of vertices that have the minimum excepted Jaccard distance from a source $s$. In addition, many graph clustering models on uncertain graphs have also been proposed. For example,

Bonchi et al. [6] proposed a $k$-core concept on uncertain graphs; and Yang et al. [55] further studied the problem of index-based $k$-core search in large uncertain graphs. Huang et al. [22] studied the problem of computing local and global truss decomposition on uncertain graphs. Qiu et al. [47] developed a graph structural clustering technique on uncertain graphs.

# 8 CONCLUSION

In this paper, we study the problem of enumerating all maximal $(k, \eta)$-cliques on an uncertain graph. We show that the state-of-the-art enumeration algorithms include many unnecessary computations, thus resulting in poor practical performance. To reduce such unnecessary computations, we first develop a novel and general pivot principle for the problem of enumerating all maximal $\mathcal{P}$-subgraphs that satisfy a hereditary property. Based on this principle, we propose two new pivot techniques which are tailored for maximal $(k, \eta)$-clique enumeration. We also propose a size-constraint based pruning technique and a new graph reduction technique to further improve the efficiency of the enumeration algorithm. We conduct extensive experiments on nine real-world graphs to evaluate the proposed algorithms. The results show that our best algorithm can be up to two orders of magnitude faster than the state-of-the-art algorithm.

# REFERENCES


[1] Eytan Adar and Christopher Re. 2007. Managing uncertainty in social networks. *IEEE Data Eng. Bull.* 30, 2 (2007), 15–22.

[2] Balabhaskar Balasundaram and Sergiy Butenko. 2006. Graph Domination, Coloring and Cliques in Telecommunications. In *Handbook of Optimization in Telecommunications*. Springer, 865–890.

[3] Vladimir Batagelj and Matjaz Zaversnik. 2003. An O(m) Algorithm for Cores Decomposition of Networks. *CoRR* cs.DS/0310049 (2003).

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3 (2003), 993–1022.

[5] Vladimir Boginski, Sergiy Butenko, and Panos M. Pardalos. 2005. Statistical analysis of financial networks. *Comput. Stat. Data Anal.* 48, 2 (2005), 431–443.

[6] Francesco Bonchi, Francesco Gullo, Andreas Kaltenbrunner, and Yana Volkovich. 2014. Core decomposition of uncertain graphs. In *KDD*. 1316–1325.

[7] Coenraad Bron and Joep Kerbosch. 1973. Finding All Cliques of an Undirected Graph (Algorithm 457). *Commun. ACM* 16, 9 (1973), 575–576.

[8] Frédéric Cazals and Chinmay Karande. 2006. *Reporting maximal cliques: new insights into an old problem*. Ph.D. Dissertation. INRIA.

[9] Lijun Chang, Jeffrey Xu Yu, and Lu Qin. 2013. Fast Maximal Cliques Enumeration in Sparse Graphs. *Algorithmica* 66, 1 (2013), 173–186.

[10] Xuelu Chen, Muhao Chen, Weijia Shi, Yizhou Sun, and Carlo Zaniolo. 2019. Embedding Uncertain Knowledge Graphs. In *AAAI*. AAAI Press, 3363–3370.

[11] Zhu-Mu Chen, Mi-Yen Yeh, and Tei-Wei Kuo. 2021. PASSLEAF: A Pool-bAsed Semi-Supervised LEArning Framework for Uncertain Knowledge Graph Embedding. In *AAAI*. AAAI Press, 4019–4026.

[12] Zi Chen, Long Yuan, Xuemin Lin, Lu Qin, and Jianye Yang. 2020. Efficient Maximal Balanced Clique Enumeration in Signed Networks. In *WWW*. 339–349.

[13] Alessio Conte, Roberto Grossi, Andrea Marino, and Luca Versari. 2016. Sublinear-Space Bounded-Delay Enumeration for Massive Network Analytics: Maximal Cliques. In *ICALP (LIPIcs, Vol. 55)*. 148:1–148:15.

[14] Marcus T Dittrich, Gunnar W Klau, Andreas Rosenwald, Thomas Dandekar, and Tobias Müller. 2008. Identifying functional modules in protein–protein interaction networks: an integrated exact approach. *Bioinformatics* 24, 13 (2008), i223–i231.

[15] Jun Dong and Steve Horvath. 2007. Understanding network concepts in modules. *BMC systems biology* 1, 1 (2007), 1–20.

[16] David Eppstein, Maarten Löffler, and Darren Strash. 2010. Listing All Maximal Cliques in Sparse Graphs in Near-Optimal Time. In *ISAAC*, Vol. 6506. 403–414.

[17] David Eppstein, Maarten Löffler, and Darren Strash. 2013. Listing All Maximal Cliques in Large Sparse Real-World Graphs. *ACM J. Exp. Algorithmics* 18 (2013).

[18] Yunjun Gao, Xiaoye Miao, Gang Chen, Baihua Zheng, Deng Cai, and Huiyong Cui. 2017. On efficiently finding reverse k-nearest neighbors over uncertain graphs. *VLDB J.* 26, 4 (2017), 467–492.

[19] Anne-Claude Gavin, Markus Bösche, Roland Krause, Paola Grandi, Martina Marzioch, Andreas Bauer, Jörg Schultz, Jens M Rick, Anne-Marie Michon, Cristina-Maria Cruciat, et al. 2002. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature* 415, 6868 (2002), 141–147.

[20] Eric Harley, Anthony Bonner, and Nathan Goodman. 2001. Uniform integration of genome mapping data using intersection graphs. *Bioinformatics* 17, 6 (2001), 487–494.

[21] William Hasenplaugh, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. 2014. Ordering heuristics for parallel graph coloring. In *SPAA*. 166–177.

[22] Xin Huang, Wei Lu, and Laks V. S. Lakshmanan. 2016. Truss Decomposition of Probabilistic Graphs: Semantics and Algorithms. In *SIGMOD*. 77–90.

[23] Rui Jiang, Zhidong Tu, Ting Chen, and Fengzhu Sun. 2006. Network motif identification in stochastic networks. *Pro. Nat. Acad. Sci.* 103, 25 (2006), 9404–9409.

[24] Ruoming Jin, Lin Liu, and Charu C. Aggarwal. 2011. Discovering highly reliable subgraphs in uncertain graphs. In *SIGKDD*. 992–1000.

[25] Ruoming Jin, Lin Liu, Bolin Ding, and Haixun Wang. 2011. Distance-Constraint Reachability Computation in Uncertain Graphs. *Proc. VLDB Endow.* 4, 9 (2011), 551–562.

[26] Xiangyu Ke, Arijit Khan, and Leroy Lim Hong Quan. 2019. An In-Depth Comparison of s-t Reliability Algorithms over Uncertain Graphs. *Proc. VLDB Endow.* 12, 8 (2019), 864–876.

[27] Arijit Khan, Francesco Bonchi, Aristides Gionis, and Francesco Gullo. 2014. Fast Reliability Search in Uncertain Graphs. In *EDBT*.

[28] Ina Koch. 2001. Enumerating all connected maximal common subgraphs in two graphs. *Theor. Comput. Sci.* 250, 1-2 (2001), 1–30.

[29] George Kollios, Michalis Potamias, and Evimaria Terzi. 2011. Clustering large probabilistic graphs. *IEEE Trans. Knowl. Data Eng.* 25, 2 (2011), 325–336.

[30] George Kollios, Michalis Potamias, and Evimaria Terzi. 2013. Clustering Large Probabilistic Graphs. *IEEE Trans. Knowl. Data Eng.* 25, 2 (2013), 325–336.

[31] Nevan J. Krogan, Gerard Cagney, Haiyuan Yu, Gouqing Zhong, Xinghua Guo, Alexandr Ignatchenko, Joyce Li, Shuye Pu, Nira Datta, Aaron P. Tikuisis, et al. 2006. Global landscape of protein complexes in the yeast Saccharomyces cerevisiae. *Nature* 440, 7084 (2006), 637–643.

[32] Rong-Hua Li, Qiangqiang Dai, Lu Qin, Guoren Wang, Xiaokui Xiao, Jeffrey Xu Yu, and Shaojie Qiao. 2018. Efficient Signed Clique Search in Signed Networks. In *ICDE*. 245–256.

[33] Rong-Hua Li, Qiangqiang Dai, Guoren Wang, Zhong Ming, Lu Qin, and Jeffrey Xu Yu. 2019. Improved Algorithms for Maximal Clique Search in Uncertain Networks. In *ICDE*. 1178–1189.

[34] Rong-Hua Li, Jeffrey Xu Yu, Rui Mao, and Tan Jin. 2016. Recursive Stratified Sampling: A New Framework for Query Evaluation on Uncertain Graphs. *IEEE Trans. Knowl. Data Eng.* 28, 2 (2016), 468–482.

[35] Don R. Lick and Arthur T. White. 1970. k-Degenerate graphs. *Canadian Journal of Mathematics* 22, 5 (1970), 1082–1096.

[36] Michael Luby. 1986. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing* 15, 4 (1986), 1036–1053.

[37] Kazuhisa Makino and Takeaki Uno. 2004. New Algorithms for Enumerating All Maximal Cliques. In *SWAT*, Vol. 3111. 260–272.

[38] Julian Mcauley and Jure Leskovec. 2014. Discovering social circles in ego networks. *ACM Trans. on Knowl. Discovery from Data* 8, 1 (2014), 1–28.

[39] Yasir Mehmood, Francesco Bonchi, and David García-Soriano. 2016. Spheres of Influence for More Effective Viral Marketing. In *SIGMOD*. 711–726.

[40] Zhuqi Miao, Balabhaskar Balasundaram, and Eduardo L. Pasiliao. 2014. An exact algorithm for the maximum probabilistic clique problem. *J. Comb. Optim.* 28, 1 (2014), 105–120.

[41] Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha P. Talukdar, Bo Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matt Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil A. Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. 2018. Never-ending learning. *Commun. ACM* 61, 5 (2018), 103–115.

[42] Arko Mukherjee, Pan Xu, and Srikanta Tirthapura. 2017. Enumeration of Maximal Cliques from an Uncertain Graph. *IEEE Trans. Knowl. Data Eng.* 29, 3 (2017), 543–555.

[43] Arko Provo Mukherjee, Pan Xu, and Srikanta Tirthapura. 2015. Mining maximal cliques from an uncertain graph. In *ICDE*. 243–254.

[44] Kevin A. Naudé. 2016. Refined pivot selection for maximal clique enumeration in graphs. *Theor. Comput. Sci.* 613 (2016), 28–37.

[45] Nishith Pathak, Sandeep Mane, and Jaideep Srivastava. 2006. Who Thinks Who Knows Who? Socio-cognitive Analysis of Email Networks. In *ICDM*. 466–477.

[46] Michalis Potamias, Francesco Bonchi, Aristides Gionis, and George Kollios. 2010. k-Nearest Neighbors in Uncertain Graphs. *PVLDB* 3, 1 (2010), 997–1008.

[47] Yu-Xuan Qiu, Rong-Hua Li, Jianxin Li, Shaojie Qiao, Guoren Wang, Jeffrey Xu Yu, and Rui Mao. 2019. Efficient Structural Clustering on Probabilistic Graphs. *IEEE Trans. Knowl. Data Eng.* 31, 10 (2019), 1954–1968.

[48] Ron Rymon. 1992. Search through Systematic Set Enumeration. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*. 539–550.

[49] Arkaprava Saha, Ruben Brokkelkamp, Yllka Velaj, Arijit Khan, and Francesco Bonchi. 2021. Shortest Paths and Centrality in Uncertain Networks. *Proc. VLDB*

*Endow.* 14, 7 (2021), 1188–1201.

[50] Roded Sharan, Igor Ulitsky, and Ron Shamir. 2007. Network-based prediction of protein function. *Molecular systems biology* 3, 1 (2007), 88.

[51] Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *AAAI*. AAAI Press, 4444–4451.

[52] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. 2006. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theor. Comput. Sci.* 363, 1 (2006), 28–42.

[53] Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. 1977. A New Algorithm for Generating All the Maximal Independent Sets. *SIAM J. Comput.* 6, 3 (1977), 505–517.

[54] Tiphaine Viard, Matthieu Latapy, and Clémence Magnien. 2016. Computing maximal cliques in link streams. *Theor. Comput. Sci.* 609 (2016), 245–252.

[55] Bohua Yang, Dong Wen, Lu Qin, Ying Zhang, Lijun Chang, and Rong-Hua Li. 2019. Index-Based Optimal Algorithm for Computing K-Cores in Large Uncertain Graphs. In *ICDE*. 64–75.

[56] Long Yuan, Lu Qin, Xuemin Lin, Lijun Chang, and Wenjie Zhang. 2017. Effective and Efficient Dynamic Graph Coloring. *Proc. VLDB Endow.* 11, 3 (2017), 338–351.

[57] Long Yuan, Lu Qin, Wenjie Zhang, Lijun Chang, and Jianye Yang. 2018. Index-Based Densest Clique Percolation Community Search in Networks. *IEEE Trans. Knowl. Data Eng.* 30, 5 (2018), 922–935.