# Efficient Core Maintenance in Dynamic Uncertain Graphs


Yu Chen
Zhejiang University
cy1098@zju.edu.cn

Qing Liu
Zhejiang University
qingliucs@zju.edu.cn

Yunjun Gao
Zhejiang University
gaoyj@zju.edu.cn



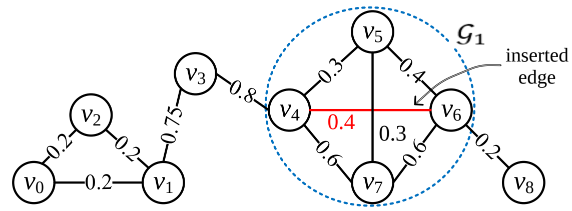
## ABSTRACT

In an uncertain graph $\mathcal{G}$, a $(k, \eta)$-core is a maximal subgraph of $\mathcal{G}$ such that the probability that each vertex's degree in the subgraph is no less than $k$ equals or exceeds $\eta$. Core decomposition in uncertain graphs, which is to calculate the $\eta$-threshold of each vertex for all possible $k$, is a fundamental problem for graphs analysis. While existing studies on core decomposition primarily address static uncertain graphs in the literature, many real-world scenarios involve highly dynamic uncertain graphs. It is costly to recompute all $\eta$-thresholds from the scratch whenever the uncertain graphs update, e.g., edge insertion and deletion, and edge probability increase and decrease. Hence, in this paper, we introduce efficient core maintenance algorithms tailored for dynamic uncertain graphs. Firstly, we investigate the impact of edge insertion and deletion on $\eta$-thresholds. Building upon this analysis, we introduce core maintenance algorithms designed to adjust the $\eta$-thresholds for edge insertions or deletions within the uncertain graphs. Our approaches involve identifying a compact subgraph encompassing all vertices necessitating $\eta$-threshold updates, followed by an iterative process of vertex deletion to complete the $\eta$-threshold updates. To improve the algorithms' efficiency, we devise three optimizations to further reduce the number of candidate $\eta$-thresholds requiring adjustment. Moreover, we extend proposed algorithms to handle the core maintenance for edge probability change. Finally, we conduct extensive experiments on both real and synthetic datasets to demonstrate the efficiency of the proposed algorithms. The results reveal that our proposed algorithms consistently outperform the baseline, exhibiting improvements ranging from at least three orders of magnitude to as high as six orders of magnitude.



**PVLDB Reference Format:**
Yu Chen, Qing Liu, and Yunjun Gao. Efficient Core Maintenance in Dynamic Uncertain Graphs. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

**PVLDB Artifact Availability:**
The source code, data, and/or other artifacts have been made available at https://github.com/ZJU-DAILY/TASK/UnMaintenance.


(a) An example uncertain graph $\mathcal{G}$

| $k$ | $v_0$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $v_7$ | $v_8$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.36 | 0.75 | 0.36 | 0.8 | 0.8 | 0.706 | 0.706 | 0.706 | 0.2 |
| 2 | 0.04 | 0.04 | 0.04 | 0.04 | 0.18 | 0.18 | 0.18 | 0.18 | - |

(b) Core decomposition result of $\mathcal{G}$

| $k$ | $v_0$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $v_7$ | $v_8$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.36 | 0.75 | 0.36 | 0.8 | 0.8 | 0.706 | 0.76 | 0.76 | 0.2 |
| 2 | 0.04 | 0.04 | 0.04 | 0.04 | 0.258 | 0.258 | 0.258 | 0.258 | - |
| 3 | - | - | - | - | 0.036 | 0.036 | 0.036 | 0.036 | - |

(c) Core decomposition after the insertion of $(v_4, v_6, 0.4)$

**Figure 1: A running example**

## 1 INTRODUCTION

Uncertainty is ubiquitous in graph data due to a variety of reasons, such as noisy measurements [2], inference and prediction models [28], and privacy concerns [8]. In these cases, the graphs are modeled as an uncertain graphs [24], whose edges are assigned with a probability of existence. Some uncertain graph examples include protein-protein interaction (PPI) networks with experimentally inferred links [4], social networks with inferred influence [36], and sensor networks with uncertain connectivity links [23]. In the literature, many cohesive subgraph models have been studied for uncertain graphs. In this paper, we focus on $(k, \eta)$-core [9].

Specifically, a $(k, \eta)$-core is a maximal subgraph of an uncertain graph $\mathcal{G}$ such that the probability that each vertex's degree in the subgraph is no less than $k$ equals or exceeds $\eta$. Given an integer $k$, the $\eta$-threshold of a vertex $u$ is the maximum $\eta$ among all $(k, \eta)$-cores containing $u$. Core decomposition in an uncertain graph $\mathcal{G}$ involves calculating the $\eta$-threshold for each vertex w.r.t. all possible $k$ values. We use Figure 1 as an example. Figure 1(a) depicts an uncertain graph $\mathcal{G}$. The subgraph $\mathcal{G}_1$ induced by vertices $\{v_4, v_5, v_6, v_7\}$ is a $(2, 0.18)$-core. Figure 1(b) shows the core decomposition results of $\mathcal{G}$. According to Figure 1(b), when $k = 1$, the $\eta$-threshold of $v_4$ is 0.8, meaning $v_4$ is in $(1, 0.8)$-core, but not in any $(1, \eta)$-cores with $\eta > 0.8$.

Core decomposition of uncertain graphs has a wide range of applications, such as describing biological functions of proteins in PPI networks [25], identifying influential spreaders in social networks [46], analyzing the topological structure of sensor networks [14], and community search [35, 39]. For example, in a collaboration uncertain graph $\mathcal{G}$, vertices represent individuals and edges associated with a probability denote the likelihood of collaboration between two individuals. The $(k, \eta)$-core decomposition of $\mathcal{G}$ can facilitate the identification of a cohesive team, wherein individuals are highly probable to collaborate [9].



Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

**Motivations.** In real-world applications, the uncertain graphs are usually dynamic. For example, in PPI networks, we may find new protein interactions or revise some existing protein interactions, and the protein interaction likelihood changes under different conditions. In social networks, the friendship and influence between users are constantly changing. In sensor networks, nodes may be added or removed, and the connectivity probability between two nodes may change due to transmission issues. These changes involve the vertices/edges insertion and deletion, as well as adjustments to edge probability, in the uncertain graph, and may further affect some downstream applications, which require the latest $\eta$-thresholds. For instance, in PPI networks, the latest $\eta$-thresholds can help to discover new protein complexes [32, 37, 38]. In social networks, the $\eta$-threshold reflects the current probability of action or information propagation between users in a community. Timely $\eta$-thresholds updating enables accurate prediction and control of information flow [1, 28, 54]. In sensor networks, maintaining $\eta$-thresholds allows for real-time tracking of changes in network topology to ensure network security [7, 17]. In these applications, a key issue is how to efficiently maintain the $\eta$-thresholds for dynamic uncertain networks.

A naive approach to maintain the $\eta$-thresholds is to perform the core decomposition from the scratch by employing the peeling algorithm without dynamic updating whenever the uncertain graph changes.[1] The corresponding time complexity is $O(k_{max} \cdot n \cdot \log n + k_{max}^2 \cdot m \cdot d_{max})$, where $m$, $n$, $k_{max}$, and $d_{max}$ denote the number of edges, the number of vertices, the maximum core number, and the maximum degree of the uncertain graph, respectively. It is time consuming to perform core decomposition from the scratch whenever the uncertain graph changes. For example, in the case of the uncertain graph *Orkut* with 117 million edges, the peeling algorithm fails to complete within a month. Hence, it is imperative to develop efficient algorithms for core maintenance in uncertain graphs. Motivated by the above reasons, for the first time, we study the problem of core maintenance in dynamic uncertain graphs, i.e., to update $\eta$-thresholds when the uncertain graph changes.

**Challenges and Solutions.** By now, many techniques have been proposed for core maintenance in different types of graphs [27, 30, 33, 45, 60]. However, these existing techniques can not be applied to the core maintenance in uncertain graphs due to different maintenance objects. Overall, the core maintenance in uncertain graphs faces two challenges. (1) The update of uncertain graphs influences not only the core number but also the $\eta$-thresholds, making core maintenance in uncertain graphs more complex compared to other types of graphs. For instance, in Figure 1(c), after inserting the edge $(v_4, v_6, 0.4)$ into $\mathcal{G}$, the core numbers of vertices $v_4, v_5, v_6$, and $v_7$ increase from 2 to 3, leading to the introduction of new $\eta$-thresholds for $k = 3$. (2) Updates in uncertain graphs can induce substantial alterations in $\eta$-thresholds, presenting challenges in pinpointing the affected $\eta$-thresholds. Notably, different $k$ values necessitate updates to different vertices' $\eta$-thresholds. Hence, distinct candidate vertex sets should be identified for different $k$ values to facilitate $\eta$-threshold updates. For instance, in Figure 1(c), for $k = 1$ and $k = 2$, the $\eta$-thresholds for updating, i.e., the red numbers, are different.

[1] As analyzed in [12], the peeling algorithm with dynamic updating returns incorrect $(k, \eta)$-core due to recursive floating-point number division operations.

Faced with these challenges, we propose efficient algorithms for core maintenance in dynamic uncertain graphs. Our approaches encompasses handling edge insertions and deletions, as well as variations in edge probabilities. Note that this paper does not delve into vertex insertions and deletions, as they can be treated as sequences of edge insertions and deletions. To efficiently maintain the cores in uncertain graphs, it is crucial to identify the candidate $\eta$-thresholds that may require updates. To this end, we first determine the range $\mathcal{R}^+$ and $\mathcal{R}^-$ of $k$ for edge insertions and deletions, respectively. Subsequently, for each $k$ in $\mathcal{R}^+/\mathcal{R}^-$, we further determine the range of $\eta$-threshold. If a vertex's $\eta$-threshold falls in this range, its $\eta$-threshold should be examined for update. Based on this, we traverse the uncertain graph from $u/v$ in a BFS manner to identify the candidate vertex set, whose $\eta$-threshold may change. Then, we iteratively delete the candidate vertex with the minimum probability that the vertex's degree is no less than $k$ to update its $\eta$-threshold. As mentioned in the first challenge, the edge insertion (resp. deletion) may introduce a new coreness (delete an old coreness) for some vertices. Finally, we update the $\eta$-thresholds for these special cases. Specifically, for the newly added coreness, we should add new $\eta$-thresholds; for the deleted coreness, we should delete the corresponding $\eta$-thresholds as well.

To improve the efficiency of above algorithms, we develop three optimizations. The first optimization introduces the concepts of restricted vertices and potential set for edge insertions and deletions, respectively, to narrow down the range of $\eta$-threshold for examination. The second optimization utilizes the vertices visited in the BFS traversal to prune the vertices whose $\eta$-threshold does not need update. The third optimization utilizes lazy computing to minimize redundant calculations. Moreover, for edge probability change, we extend the above proposed algorithms and optimizations to handle the core maintenance.

**Contributions.** Our principal contributions are as follows.

- We formally define the core maintenance problem in dynamic uncertain graphs. To the best of our knowledge, it is the first time to explore this problem.
- We propose basic core maintenance algorithms for edge insertions and deletions. Furthermore, we devise three optimizations to further improve the efficiency of algorithms.
- We extend the proposed algorithms to handle the core maintenance for edge probability change.
- We conduct extensive experiments on both real and synthetic uncertain graphs to demonstrate the efficiency and effectiveness of our proposed algorithms and optimizations.

**Outline.** Section 2 reviews the related works. Section 3 formally defines the problem. Sections 4 and 5 introduce core maintenance algorithms for edge insertion and deletion, respectively. Section 6 presents the core maintenance algorithms for the edge probability change. Section 7 reports the experimental results for out proposed algorithms. Finally, Section 8 concludes the paper.

## 2 RELATED WORK

**Cohesive subgraphs in uncertain graphs**. Many cohesive subgraph models have been proposed for uncertain graphs, including core-based models (e.g., $(k, \eta)$-core [9], $(\alpha, \beta, \eta)$-core [58], and $(k, \theta)$-core [39, 42]), truss-based models (e.g., $k$-Bitruss [59] and

$(k, \gamma)$-truss [21, 62]), and clique-based models (e.g., $(k, \tau)$-clique [11, 26], $\alpha$-clique [41, 61], and $(\alpha, \gamma)$-quasi-clique [40]).

In this work, we primarily focus on the $(k, \eta)$-core model. To be specific, Bonchi et al. [9] firstly extended the $k$-core model to uncertain graphs and proposed a peeling algorithm with dynamical $\eta$-degree updating technique for core decomposition. Here, $\eta$-degree of a vertex $u$ is the maximum $k$ such that the probability that $u$'s degree is no less than $k$ equals or exceeds $\eta$. Then, Yang et al. [53] proposed a UCF-Index, which store the core decomposition results, to facilitate a single $(k, \eta)$-core search. In some cases, the graph may be too large to be loaded into memory, and Wen et al. [50] extended the UCF-Index to UCEF-Index, which is constructed in external memory. Moreover, Dai et al. [12] discovered that employing the peeling algorithm with the dynamical $\eta$-degree updating technique to compute the $(k, \eta)$-core is incorrect, due to recursive floating-point number division operations. To compute the $(k, \eta)$-core correctly, Dai et al. [12] proposed bottom-up and top-down algorithms. It is worth mentioning that existing works of $(k, \eta)$-core consider static uncertain graphs, which can not be applied for efficient $(k, \eta)$-core maintenance.

**Core maintenance**. In the literature, many efforts have been devoted to the study of core maintenance, which is to maintain the core number of each vertex for dynamic graphs. [27, 45] primarily focused on single edge operations to maintain vertex core number. Zhang et al. [57] proposed an order-based approach to reduce the time by maintaining a $k$-order. Then, [5, 6, 18, 19, 22, 47] have investigated core maintenance with multiple edge operations. Since the connections among different $k$-cores in the hierarchy are not considered in above works, [29] studied the hierarchical core maintenance which is to compute the $k$-core hierarchy incrementally. Moreover, [3, 51] explored the core maintenance in distributed environment. In addition, the core maintenance problem has also been studied for other core models, including colorful $h$-star core [16] and distance-generalized core [31], and various types of dynamic graphs, including weighted graphs [48, 52, 55, 60], bipartite graphs [30, 34, 49], and hypergraphs [15, 20, 33]. Nevertheless, due to the different definitions and semantics of core models, the existing core maintenance works cannot be extended to handle the core maintenance in dynamic uncertain graphs.

## 3 PRELIMINARIES

We consider an undirected uncertain graph $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}}, p)$, where $V_{\mathcal{G}}$ is the set of vertices, $E_{\mathcal{G}}$ is the set of edges, and $p$: $E_{\mathcal{G}} \rightarrow (0, 1]$ is a function assigning an existence probability to each edge. Given a vertex $u \in V_{\mathcal{G}}$, the degree of $u$ is $deg(u, \mathcal{G}) = |\{v \in V_{\mathcal{G}} | (u, v) \in E_{\mathcal{G}}\}|$. For a vertex set $V_C \subseteq V_{\mathcal{G}}$, the induced subgraph of $V_C$ is denoted by $\mathcal{G}[V_C] = (V_C, \{(u, v) | \forall u, v \in V_C \wedge (u, v) \in E_{\mathcal{G}}\}, p)$. We denote the probability of an edge $e \in E_{\mathcal{G}}$ by $p_e$, and assume that the existence probability of each edge is independent of each other [9, 39, 50, 53]. In this paper, we adopt the well-known possible-world semantics [44] for uncertain graph analysis. Specifically, let $G = (V_G, E_G) \sqsubseteq \mathcal{G}$ be a possible world of $\mathcal{G}$, where $V_G = V_{\mathcal{G}}$ and $E_G \subseteq E_{\mathcal{G}}$. The probability of $G$, denoted by $Pr(G)$, is:

$$Pr(G) = \prod_{e \in E_G} p_e \cdot \prod_{e \in E_{\mathcal{G}} \setminus E_G} (1 - p_e) \tag{1}$$

Note that each possible world is a deterministic graph, and, totally, there are $2^{|E_{\mathcal{G}}|}$ possible worlds for an uncertain graph $\mathcal{G}$. Let $\mathcal{G}^u_{\geq k}$ be the set of all possible worlds of $\mathcal{G}$, where $u$'s degree is not less than $k$, i.e., $\mathcal{G}^u_{\geq k} = \{G \sqsubseteq \mathcal{G} | deg(u, G) \geq k\}$. Based on it, we give the definition of $k$-probability as follows.

Definition 3.1. **($k$-probability)** *[53] Given an uncertain graph $\mathcal{G}$ and an integer $k$, the $k$-probability of a vertex $u \in V_{\mathcal{G}}$, denoted by $k$-$prob(u, \mathcal{G})$, is the probability that $u$'s degree is not less than $k$, i.e., $k$-$prob(u, \mathcal{G}) = Pr[deg(u, \mathcal{G}) \geq k] = \sum_{G \subseteq \mathcal{G}^u_{\geq k}} Pr(G)$.*

Take the vertex $v_3$ in Figure 1(a) as an example. 1-$prob(v_3, \mathcal{G})$ = $1 - 0.25 * 0.2 = 0.95$ and 2-$prob(v_3, \mathcal{G}) = 0.75 * 0.8 = 0.6$. We use $k$-$prob(u)$ instead of $k$-prob$(u, \mathcal{G})$ when the context is clear. Next, we give the core definition for uncertain graphs.

Definition 3.2. **($(k, \eta)$-core)** *[9] Given an uncertain graph $\mathcal{G}$, an integer $k$, and a probabilistic threshold $\eta \in [0, 1]$, a $(k, \eta)$-core is a maximal subgraph $\mathcal{H} = (V_{\mathcal{H}}, E_{\mathcal{H}}, p) \subseteq \mathcal{G}$ satisfying: for each vertex $u \in V_{\mathcal{H}}$, the probability that $u$'s degree is no less than $k$ in $\mathcal{H}$ equals or exceeds $\eta$, i.e., $\forall u \in V_{\mathcal{H}}$, $k$-$prob(u, \mathcal{H}) \geq \eta$.*

For example, in Figure 1(a), let $k = 2$ and $\eta = 0.1$. The subgraph $\mathcal{G}_1$ is a (2, 0.1)-core since (i) 2-prob$(v_4, \mathcal{G}_1) = 0.18$, 2-prob$(v_5, \mathcal{G}_1)$ = 0.258, 2-prob$(v_6, \mathcal{G}_1) = 0.24$, and 2-prob$(v_7, \mathcal{G}_1) = 0.504$, and (ii) $\forall v \in \mathcal{G} - \mathcal{G}_1, \forall \mathcal{G}' \subseteq \mathcal{G}$, 2-prob$(v, \mathcal{G}') < 0.1$. We would like to highlight three points for the $(k, \eta)$-core. (1) When $\eta = 0$, the $(k, \eta)$-core of an uncertain graph $\mathcal{G}$ corresponds to the $k$-core of the corresponding deterministic graph by ignoring the edge probabilities of $\mathcal{G}$. Hence, we assume $\eta > 0$ in this paper. (2) For a deterministic graph, the core number of a vertex $u$, denoted by $c(u)$, is the maximum $k$ such that there exists a non-empty $k$-core containing $u$. For a vertex $u \in V_{\mathcal{G}}$, if $k > c(u)$, there does not exist a $(k, \eta)$-core with $\eta > 0$ containing $u$. (3) Like $k$-core, the $(k, \eta)$-core also has the nesting property. Specifically, given an integer $k$ and two probabilistic thresholds $\eta_1$ and $\eta_2$ with $0 \leq \eta_1 \leq \eta_2 \leq 1$, it holds that $(k, \eta_1)$-core contains $(k, \eta_2)$-core. Based on the nesting property of the $(k, \eta)$-core, $\eta$-threshold is defined as follows.

Definition 3.3. **($\eta$-threshold)** *[53] Given an uncertain graph $\mathcal{G}$, an integer $k$, and a vertex $u \in V_{\mathcal{G}}$, the $\eta$-threshold of $u$ w.r.t. $k$, denoted by $\eta(k, u)$, is the maximum $\eta$ such that there is a non-empty $(k, \eta)$-core containing $u$.*

In a word, the $\eta$-threshold of $u$ w.r.t. $k$ represents the maximum probability of $u$ in the $k$-core. For example, in Figure 1(a), let $k = 2$, we have $\eta(2, v_4) = 0.18$ since there does not exist a $(2, \eta)$-core with $\eta > 0.18$ containing $v_4$. By employing the $\eta$-threshold, the $(k, \eta)$-core also can be defined as follows: the $(k, \eta)$-core consists of all vertices $u$ with $\eta(k, u) \geq \eta$. Based on the $\eta$-threshold, the core decomposition and maintenance in uncertain graphs are defined as follows.

Definition 3.4. **(Core decomposition in uncertain graphs)** *[53] Given an uncertain graph $\mathcal{G}$, the core decomposition of $\mathcal{G}$ is to calculate $\eta(k, u)$ for each vertex $u \in V_{\mathcal{G}}$ w.r.t. each value of $k \in [1, c(u)]$.*

Problem 1. **(Core maintenance in uncertain graphs)** *Given an uncertain graph $\mathcal{G}$ and its core decomposition result, when $\mathcal{G}$ updates, the core maintenance of $\mathcal{G}$ is to update $\eta(k, u)$ for $\forall u \in V_{\mathcal{G}}$ w.r.t. $\forall k \in [1, c(u)]$.*

For example, in Figure 1(b), the core decomposition results of $\mathcal{G}$ are depicted, while Figure 1(c) displays the core decomposition results after the insertion of the edge $(v_4, v_6, 0.4)$ into $\mathcal{G}$. As analyzed in Introduction, when the uncertain graph updates, it is costly to perform core decomposition from scratch. Therefore, we propose efficient maintenance algorithms in the following three sections.

# 4 THE BASIC ALGORITHMS

In this section, we present the basic core maintenance algorithms for the edge insertion and deletion, respectively. Note that, due to space limitation, some proofs are removed to appendix [10].

## 4.1 Edge Insertion

According to Problem 1, we should update the $\eta$-threshold $\eta(k, u)$ for all vertices and $k$'s. To avoid recomputing all $\eta(k, u)$'s, we should identify the candidate $\eta$-thresholds that need to be updated.

Li et al. [27] has demonstrated that, for deterministic graphs, when an edge $(u, v)$ is inserted or deleted with $c(u) \le c(v)$, it holds that (1) only vertices with the same core number as $u$ require to update their core numbers, and (2) core numbers increase or decrease by at most 1. Based on it, we have the following theorem.

Theorem 4.1. *Assume that an edge $(u, v, p_{(u,v)})$ is inserted into an uncertain graph $\mathcal{G}$. Let $k_0 = min\{c(u), c(v)\}$. For a vertex $w \in V_{\mathcal{G}}$, if $k \in \mathcal{R}^+ = [1, k_0 + 1]$, $\eta(k, w)$ needs to be updated. Otherwise, $\eta(k, w)$ does not change.*

Proof. We prove Theorem 4.1 from three cases. (1) If $k \in [1, k_0]$, an edge is added to the initial $(k, \eta(k, w))$-core, causing a change in $\eta(k, w)$. (2) If $k = k_0 + 1$, the core number of certain vertices may increase by 1 to $k_0 + 1$. These vertices are added to the initial $(k_0 + 1, \eta(k_0 + 1, w))$-core, causing a change in $\eta(k_0 + 1, w)$. (3) If $k \notin [1, k_0 + 1]$, no new vertices or edges are added to the original $(k, \eta(k, w))$-core. Thus, $\eta(k, w)$ remains unchanged. □

Theorem 4.1 indicates the range of $k$ values to determine which $\eta(k, w)$ should be updated. Based on this, for each $k \in \mathcal{R}^+$, we first propose a connectivity theorem.

Theorem 4.2. *Assume that an edge $(u, v, p_{(u,v)})$ is inserted into an uncertain graph $\mathcal{G}$. Given an integer $k \in \mathcal{R}^+$, all the vertices whose $\eta(k, w)$ values have changed can form a connected subgraph $\mathcal{G}' \subseteq \mathcal{G} \cup (u, v, p_{(u,v)})$.*

Since $\eta(k, w)$ has another parameter, i.e., the vertex, in what follows, we introduce how to identify the candidate vertex, whose $\eta(k, w)$ should be updated for a fixed $k$. To facilitate clarity, we introduce the concept of induced $\eta$-threshold subgraph.

Definition 4.1. **(Induced $\eta$-threshold subgraph)** *Given an uncertain graph $\mathcal{G}$ and an integer $k$, the induced $\eta$-threshold subgraph for a vertex $w \in V_{\mathcal{G}}$, denoted by $\mathcal{G}_k^w$, is the $(k, \eta(k, w))$-core containing $w$.*

According to the definition of $(k, \eta)$-core, the probability that $w$ has a degree of at least $k$ in $\mathcal{G}_k^w$ is not less than $\eta(k, w)$, i.e., $k\text{-prob}(w, \mathcal{G}_k^w) \ge \eta(k, w)$. We use $N(w, \mathcal{G}_k^w)$, simplified to $N_k^w$, to represent the neighbor set of $w$ in $\mathcal{G}_k^w$.

To further identify the impact of different $k$ values on vertices' $\eta$-thresholds, we initially focus on a fixed $k \in \mathcal{R}^+$. When an edge $(u, v, p_{(u,v)})$ is inserted, there are three cases: (1) $\eta(k, u) < \eta(k, v)$, (2) $\eta(k, u) = \eta(k, v)$, (3) $\eta(k, u) > \eta(k, v)$. Below, we provide a detailed analysis of case-1. After the insertion, $u$ acquires a new neighbor $v$, which has a higher $\eta(k, v)$. Thus, the edge $(u, v)$ becomes part of the updated $\eta$-threshold subgraph of $u$, denoted as $\mathcal{G}_k^{u+}$. The updated neighbor set of $u$, denoted as $N_k^{u+} = N_k^u \cup \{v\}$, leads to an increase in the $k$-probability of $u$. In this case, the inequality $k\text{-prob}(u, \mathcal{G}_k^{u+}) > k\text{-prob}(u, \mathcal{G}_k^u)$ must hold. For convenience, $k\text{-}prob(u, \mathcal{G}_k^u)$ and $k\text{-}prob(u, \mathcal{G}_k^{u+})$ are simplified as $k\text{-}prob(u)$ and $k\text{-}prob^+(u)$, respectively. Based on the above definition, we propose the following theorem.

Theorem 4.3. *Assume that an edge $(u, v, p_{(u,v)})$ with $\eta(k, u) < \eta(k, v)$ is inserted into an uncertain graph $\mathcal{G}$. Given an integer $k \in \mathcal{R}^+$ and a vertex $w \in V_{\mathcal{G}}$, we have (1) if $\eta(k, w) < \eta(k, u)$, $\eta(k, w)$ does not change; and (2) if $\eta(k, w)$ changes, it increases at most to $k\text{-}prob^+(u)$.*

Proof. First, we prove conclusion (1). If $\eta(k, w) < \eta(k, u) < \eta(k, v)$, the initial $(k, \eta(k, w))$-core has contained both $u$ and $v$. The inserted edge will not cause an increase in the $k$-probability of $w$. Thus, $\eta(k, w)$ remains unchanged. Second, we prove conclusion (2) by contradiction. Assume that the updated $\eta$-threshold of $u$, denoted as $\eta'(k, u)$, is greater than $k\text{-}prob^+(u)$. It follows that $k\text{-}prob(u, \mathcal{G}_k^{u'}) > k\text{-}prob(u, \mathcal{G}_k^{u+})$, where $\mathcal{G}_k^{u'}$ represents the induced $\eta$-threshold subgraph of $u$ after the insertion. That is, $\mathcal{G}_k^{u+}$ is a proper subgraph of $\mathcal{G}_k^{u'}$ ($\mathcal{G}_k^{u+} \subset \mathcal{G}_k^{u'}$). However, $\mathcal{G}_k^{u+}$ has contained $\mathcal{G}_k^u$ and the inserted edge $(u, v)$. Based on conclusion (1), $\eta(k, x)$ remains unchanged if $\eta(k, x) < \eta(k, u)$, so that $x$ cannot be part of $\mathcal{G}_k^{u+}$. As a result, $\mathcal{G}_k^{u'}$ is at most $\mathcal{G}_k^{u+}$, which is a contradiction. We conclude that $\eta(k, u)$ increases to at most $k\text{-}prob^+(u)$. Similarly, assume that $\eta(k, w)$ is increased to $\eta'(k, w)$ and $\eta'(k, w) > k\text{-}prob^+(u)$. It must be true that $(u, v) \in \mathcal{G}_k^{w'}$, as otherwise $\mathcal{G}_k^{w'}$ forms a seed $(k, \eta_k'(w))$-core before the insertion as well, which is a contradiction. However, we have proved that $\eta(k, u)$ increases to at most $k\text{-}prob^+(u)$. Thus, edge $(u, v)$ cannot be part of $\mathcal{G}_k^{w'}$, which is again contradiction. □

Based on Theorem 4.3, we formally present the core update theorem for the insertion of an edge an uncertain graphs, which contains three different cases.

Theorem 4.4. *Assume that an edge $(u, v, p_{(u,v)})$ is inserted into an uncertain graph $\mathcal{G}$. Given an integer $k \in \mathcal{R}^+$ and a vertex $w \in V_{\mathcal{G}}$, we have the following results.*

- *if $\eta(k, u) < \eta(k, v)$ and $\eta(k, u) \le \eta(k, w) < k\text{-}prob^+(u)$, $\eta(k, w)$ may require updating and $u$ is assigned as the root.*
- *if $\eta(k, u) = \eta(k, v)$ and $\eta(k, u) \le \eta(k, w) < min\{k\text{-}prob^+(u), k\text{-}prob^+(v)\}$, $\eta(k, w)$ may require updating and both $u$ and $v$ can be assigned as the root.*
- *if $\eta(k, u) > \eta(k, v)$ and $\eta(k, v) \le \eta(k, w) < k\text{-}prob^+(v)$, $\eta(k, w)$ may require updating and $v$ is assigned as the root.*

*According to Theorem 4.2, $w$ must be connected to the root through a path containing only vertices within the $\eta$-threshold range.*

proof. We prove three cases respectively. (1) if $\eta(k, u) < \eta(k, v)$, Theorem 4.3 demonstrates that only $\eta(k, w) \in [\eta(k, u), k\text{-}prob^+(u))$ may need to be updated. Furthermore, if $\eta(k, w)$ increases due to

**Algorithm 1: Edge Insertion (EI)**

**Input:** $\mathcal{G}$, an edge $(u, v, p_{(u,v)})$ to be inserted
**Output:** the updated $\eta(k, w)$ for $w \in V_{\mathcal{G}}$ at each possible $k$

1 $\mathcal{G}' \leftarrow \mathcal{G} \cup (u, v, p_{(u,v)})$;
2 $\mathcal{R}' \leftarrow [1, min\{c(u), c(v)\}]$;
3 **foreach** $k \in \mathcal{R}'$ **do**
4     determine lb, ub and the root using Theorem 4.4;
5     $C \leftarrow$ *FindInsert* $(\mathcal{G}', root, lb, ub, k)$;
6     *UpdateInsert* $(k, C)$;
7 $k$++;    *// critical $\mathcal{R}^+$ processing*
8 $V_c \leftarrow$ *Insertion* $(G, u, v)$;    //Maintaining the core number of each vertex using Algorithm proposed in [27].
9 **if** $V_c = \varnothing$ **then**
10     **return** *;*
11 **if** $k > max_{i \in V_{\mathcal{G}} \setminus V_c}\{c(i)\}$ **then**
12     *UpdateInsert* $(k, V_c)$;
13 **else**
14     determine the root using Theorem 4.4;
15     $lb = 0$;    $ub = k\text{-}prob(root, \mathcal{G}_k^{root+} \cup V_c)$;
16     $C \leftarrow$ *FindInsert* $(\mathcal{G}', root, lb, ub, k)$;
17     *UpdateInsert* $(k, C)$;

the insertion, then $\eta(k, u)$ must also increase. This arises from the recursive argument in the proof of Theorem 4.2. Otherwise, no vertices would have their $\eta$-thresholds changed. We know that all updated vertices form a connected subgraph $\mathcal{G}'$ (Theorem 4.2). As $u$ also belongs to $\mathcal{G}'$, the proof completes. (2) For $\eta(k, u) > \eta(k, v)$, it can be proved in the same way as the case $\eta(k, u) < \eta(k, v)$, and thus is omitted here. (3) If $\eta(k, u) = \eta(k, v)$, we perform the above calculations for both $u$ and $v$. The updated $\eta$-thresholds of $u$ and $v$, denoted by $\eta'(k, u)$ and $\eta'(k, v)$, are less than or equal to $k\text{-}prob^+(u)$ and $k\text{-}prob^+(v)$, respectively. According to Definition 3.2, $u$ is unable to join the $(k, k\text{-}prob^+(v))$-core if $k\text{-}prob^+(u) < k\text{-}prob^+(v)$. The case is same for $v$. However, $u$ and $v$ may join the same $(k, \eta')$-core, where $\eta'$ is the minimum between $k\text{-}prob^+(u)$ and $k\text{-}prob^+(v)$. Thus, $min\{k\text{-}prob^+(u), k\text{-}prob^+(v)\}$ serves as upper-bound for both $u$ and $v$. This completely proves Theorem 4.4. ◻

Next, we formally propose the Edge Insertion (EI, Algorithm 1) algorithm for core maintenance in uncertain graphs. Firstly, Theorem 4.1 determines the range $\mathcal{R}^+$ of $k$ for edge insertions. We divide $\mathcal{R}^+$ into two parts: $\mathcal{R}^+ = \mathcal{R}' \cup \{k_0 + 1\}$, with $\mathcal{R}' = [1, k_0]$. For each $k \in \mathcal{R}'$, lines 3-6 correctly maintain the $\eta$-threshold updates via the following steps. Theorem 4.4 determines the $\eta$-threshold range $[lb, ub)$ and the root at $k$ (line 4). Then, Algorithm 2 identifies the candidate set $C$ (line 5) and Algorithm 3 updates the $\eta(k, w)$ value for each vertex $w \in C$ (line 6). Since the edge insertion may introduce a new coreness for some vertices, we should add new $\eta$-thresholds for them. Finally, critical $\mathcal{R}^+$ processing (lines 7-17) is used to handle this special case. Detailed descriptions for the related algorithms are provided below.

Algorithm 2, named **FindInsert**, performs a breadth-first search (BFS) traversal to identify all candidates reachable from the root through vertices within the $\eta$-threshold range. Initially, the candidate set $C$ is empty and the queue $Q$ stores the vertices to be processed, starting with the addition of the root(lines 1-2). Each

**Algorithm 2: FindInsert**

**Input:** updated graph: $\mathcal{G}$, root: $r$, $lb$, $ub$, $k$
**Output:** candidate set $C$

1 $C \leftarrow$ empty set; $visited[] \leftarrow$ **false**;
2 $Q \leftarrow$ empty queue; $Q$.push($r$); $visited[r] \leftarrow$ **true**;
3 **while** $Q \neq \varnothing$ **do**
4     $u \leftarrow Q$.pop(); $C$.push($u$)
5     **foreach** $(u, v) \in E$ **do**
6         **if** $lb \leq \eta(k, v) < ub \wedge \neg visited[v]$ **then**
7             $Q$.push($v$); $visited[v] \leftarrow$ **true**;
8 **return** $C$ *;*

**Algorithm 3: UpdateInsert**

**Input:** $k$-value: $k$, candidate set: $C$
**Output:** the updated $\eta(k, w)$ for each vertex $w \in C$

1 $curThres \leftarrow 0$;
2 compute $k\text{-}prob(v)$ for each $v \in C$;    // DP Algorithm [9]
3 **while** $C \neq \varnothing$ **do**
4     $u \leftarrow arg\ min_{v \in C} k\text{-}prob(v)$;
5     $curThres \leftarrow max\{k\text{-}prob(u), curThres\}$;
6     $\eta(k, u) = curThres$;
7     $C \leftarrow C \setminus \{u\}$
8     **foreach** $(u, v) \in E_{\mathcal{G}} \wedge v \in C$ **do**
9         update $k\text{-}prob(v)$;

vertex in $Q$ is processed individually, and all its neighbors within the $\eta$-threshold range are identified and added to $Q$. This iterative process continues until $Q$ is empty (lines 3-7). Finally, the candidate set $C$ containing all candidates is returned.

Algorithm 3, named **UpdateInsert**, updates the $\eta(k, w)$ value of each vertex $w \in C$ by iteratively removing the vertices with the minimum $k$-probability. Initially, the DP algorithm [9] initializes the $k$-probability of each candidate (line 2). Then, a vertex $u$ with the minimum $k$-probability is selected, and curThres is updated accordingly (lines 4-5). The updated curThres is saved as $\eta(k, u)$, and $u$ is removed from the candidate set $C$ (lines 6-7). Finally, DP algorithm recalculates the $k$-probability for each neighbor of $u$ (lines 8-10). Once the algorithm terminates, all vertices in $C$ have been updated at the current $k$.

The detailed pseudocode for critical $\mathcal{R}^+$ processing can be found in lines 7-17 of Algorithm 1. Initially, we use the core maintenance algorithm proposed for deterministic graph edge insertion [27] to identify and update the vertices that require an increment of 1 in their core numbers. The set $V_c$ represents these vertices. Let $u$ be one such vertex, and $c'(u)$ denote its updated core number. In the following discussion, we will address three distinct cases.

**Case-1:** $V_c$ is empty (lines 9-10). This indicates that the core numbers of all vertices remain unchanged. Consequently, no new vertices or edges are added to the original $(k_0 + 1, \eta(k_0 + 1, w))$-core, ensuring that the $\eta(k_0 + 1, w)$ value remains unaffected.

**Case-2:** $V_c$ is not empty and $c'(u)$ exceeds the core numbers of all other vertices, which remain unchanged (lines 11-12). Prior to the edge insertion, there is no $(c'(u), \eta)$-core ($\eta > 0$). After the edge insertion, such a core will emerge. The candidate set at this stage is $V_c$. Next, UpdateInsert is employed to update the $\eta(c'(u), w)$ of each candidate $w \in V_c$.

**Case-3:** $V_c$ is not empty and $c'(u)$ is less than or equal to the core numbers of other vertices (lines 15-18). After the insertion, the new $\eta(c'(u), u)$ is added, so that the lower-bound ($lb$) is set to 0 (line 15). Since a vertex with an increased core number must be connected to $u$, the updated neighbor set of $u$ may satisfy $N_u^+ \subseteq (N_u \cup V_c)$. Thus, we can find the upper-bound ($ub$) and the root within the subgraph ($\mathcal{G}_k^{u+} \cup V_c$) by Theorem 4.4 (lines 14-15). Then, FindInsert and UpdateInsert update the $\eta(c'(u), w)$ for each vertex $w \in$ C.

Example 4.1. *In Figure 1(b), let $(v_4, v_6, 0.4)$ be the edge to be inserted. Since $c(v_4) = c(v_6) = 2$, we can determine $\mathcal{R}^+ = [1, 3]$. Each $k \in \mathcal{R}^+$ will be processed sequentially. When $k = 1$, the original $\eta(1, v_4)$ and $\eta(1, v_6)$ are 0.8 and 0.706, respectively. According to Theorem 4.4, $lb = 0.706$, root $= v_6$, and $ub = k\text{-}prob(v_6, \mathcal{G}_1^{v_6+}) = 0.856$. Next, a BFS traversal is performed starting from $v_6$. The candidate set $C = \{v_6, v_4, v_5, v_7, v_3, v_1\}$ can be identified and updated. When $k = 2$, the execution steps are the same as above. When $k = 3$, the Insertion algorithm returns $V_c = \{v_4, \ldots, v_7\}$. This corresponds to case-2. We directly determine and update the candidate set $C = V_c$.*

## 4.2 Edge Deletion

Firstly, Theorem 4.5 analyzes the range of $k$ values to determine which $\eta(k, w)$ may be affected by edge deletion. Then, for each possible $k$, we propose a connectivity theorem.

Theorem 4.5. *Assume that an edge $(u, v, p_{(u,v)})$ is deleted from an uncertain graph $\mathcal{G}$. Let $k_0 = min\{c(u), c(v)\}$. For a vertex $w \in V_{\mathcal{G}}$, if $k \in \mathcal{R}^- = [1, k_0]$, $\eta(k, w)$ needs to be updated. Otherwise, $\eta(k, w)$ does not change.*

Theorem 4.6. *Assume that an edge $(u, v, p_{(u,v)})$ is deleted from an uncertain graph $\mathcal{G}$. Given an integer $k \in \mathcal{R}^-$, all the vertices whose $\eta(k, w)$ values have changed should form a connected subgraph $\mathcal{G}'' \subseteq \mathcal{G}$.*

Similarly, $\eta(k, w)$ has another parameter related to $k$, i.e., the vertex. We introduce how to identify the candidate vertex, whose $\eta(k, w)$ may be updated. Given an integer $k \in \mathcal{R}^-$, if an edge $(u, v, p_{(u,v)})$ is deleted, there are also three distinct cases. We primarily consider the $\eta(k, u) < \eta(k, v)$ case. After the deletion, $u$ loses one neighbor $v$, which has a higher $\eta(k, v)$. The updated $\eta$-threshold subgraph of $u$, denoted as $\mathcal{G}_k^{u-}$, leads to a decrease in the $k$-probability of $u$. Thus, the inequality $k\text{-}prob(u, \mathcal{G}_k^{u-}) < k\text{-}prob(u, \mathcal{G}_k^{u})$ holds in this case. For convenience, we simplify $k$-prob$(u, \mathcal{G}_k^{u-})$ to $k$-prob$^-(u)$. Based on this, we propose the following theorem.

Theorem 4.7. *Assume that an edge $(u, v, p_{(u,v)})$ with $\eta(k, u) < \eta(k, v)$ is deleted from an uncertain graph $\mathcal{G}$. Given an integer $k \in \mathcal{R}^-$ and a vertex $w \in V_{\mathcal{G}}$, we have (1) if $\eta(k, w) > \eta(k, u)$, $\eta(k, w)$ does not change; and (2) if $\eta(k, w)$ changes, it decreases at least to $k\text{-}prob^-(u)$.*

Proof. First, we prove conclusion (1). If $\eta(k, u) < \eta(k, w)$, the initial $(k, \eta(k, w))$-core does not include $u$. Thus, the deletion does not affect the $(k, \eta(k, w))$-core, and $\eta(k, w)$ remains unchanged. Second, we prove conclusion (2). We denote the updated $\eta$-threshold of vertex $w$ at integer $k$ as $\eta'(k, w)$. Let $x$ be a neighbor of $u$ and $\eta(k, x) = \eta(k, u)$. There are two different cases: (i) if $k\text{-}prob(u) \leq k\text{-}prob(x)$, the equation $\eta(k, x) = \eta(k, u) = k\text{-}prob(u)$ must exist. After the deletion, there is $k\text{-}prob^-(u) < k\text{-}prob(x)$. This corresponds to case-i, and we easily get $\eta'(k, u) \leq \eta'(k, x)$. (ii) if $k\text{-}prob(x) < k\text{-}prob(u)$, the equation $\eta(k, x) = \eta(k, u) = k\text{-}prob(x)$ must exist. If $k\text{-}prob^-(u) < k\text{-}prob(x)$, this is the same as described above. If $k\text{-}prob(x) \leq k\text{-}prob^-(u)$, both $(k, \eta(k, u))$-core and $(k, \eta(k, x))$-core remain unchanged. Thus, $\eta'(k, x) = \eta'(k, u) = k\text{-}prob(x)$. In both cases, the inequality $\eta'(k, x) \geq \eta'(k, u)$ holds.

Next, we prove $\eta'(k, u)$ decreases at least to $k\text{-}prob^-(u)$ by contradiction. Assume that $\eta'(k, u)$ is less than $k\text{-}prob^-(u)$. However, based on the above proof, if $\eta(k, y) \geq \eta(k, u)$, we have $\eta'(k, y) \geq \eta'(k, u)$. These vertices and $u$ can form a $(k, \eta)$-core, where $u$ haing the smallest $\eta$-threshold. That is, $\eta = prob^-(u)$. According to Definition 3.3, $k\text{-}prob^-(u)$ will be the updated $\eta$-threshold of $u$, which is a contradiction. Furthermore, we consider the one-hop neighbor $w$ of $u$. If $\mathcal{G}_k^w$ contains $u$ before the deletion, then $\eta'(k, w)$ decreases to at least $k\text{-}prob^-(u)$ due to $u$. By recursively considering the one-hop neighbors of $w$, we can conclude that all updated $\eta'(k, w)$ decreases to at least $k\text{-}prob^-(u)$. This completes the proof. □

Based on Theorem 4.7, we formally present the core update theorem for the deletion of an edge in uncertain graphs.

Theorem 4.8. *Assume that an edge $(u, v, p_{(u,v)})$ is deleted from an uncertain graph $\mathcal{G}$. Given an integer $k \in \mathcal{R}^-$ and a vertex $w \in V_{\mathcal{G}}$, we have the following results.*

- *if $\eta(k, u) < \eta(k, v)$ and $k\text{-}prob^-(u) < \eta(k, w) \leq \eta(k, u)$, $\eta(k, w)$ may require updating and $u$ is assigned as the root.*
- *if $\eta(k, u) = \eta(k, v)$ and $min\{k\text{-}prob^-(u), k\text{-}prob^-(v)\} < \eta(k, w) \leq \eta(k, u)$, $\eta(k, w)$ may require updating and both $u$ and $v$ are assigned as the root.*
- *if $\eta(k, u) > \eta(k, v)$ and $k\text{-}prob^-(v) < \eta(k, w) \leq \eta(k, v)$, $\eta(k, w)$ may require updating and $v$ is assigned as the root.*

*According to Theorem 4.6, $w$ must be connected to the root through a path containing only the vertices within the $\eta$-threshold range.*

Proof. We prove three cases respectively. (1) if $\eta(k, u) < \eta(k, v)$, Theorem 4.7 has proved that only $\eta(k, w) \in (k\text{-}prob^-(u), \eta(k, u)]$ may need to be updated. Similarly, if the $\eta(k, w)$ value of any vertex $w \in \mathcal{G}$ decreases, then $\eta(k, u)$ must have decreased as well. Theorem 4.6 shows that all those vertices form a connected subgraph $\mathcal{G}''$. As $u$ belongs to $\mathcal{G}''$, the proof is complete. (2) if $\eta(k, u) > \eta(k, v)$, similar proofs can be used in this case, but for brevity, they are omitted here. (3) if $\eta(k, u) = \eta(k, v)$, we perform the above calculations for both $u$ and $v$. The updated $\eta$-thresholds of $u$ and $v$ are denoted as $\eta'(k, u)$ and $\eta'(k, v)$, respectively. Let $w$ be the common neighbor of $u$ and $v$, and $\eta(k, w) = \eta(k, u) = \eta(k, v)$. After the deletion, if $k\text{-}prob^-(u) < k\text{-}prob^-(v)$, $\eta'(k, u)$ is initially determined as $k\text{-}prob^-(u)$. If $k$ neighbors of vertex $w$ are dependent on $u$, $\eta'(k, w)$ is updated to $k\text{-}prob^-(u)$. Finally, $\eta'(k, v)$ can be identified. It is necessary that $\eta'(k, v) \geq \eta'(k, w)$ exists. However, according to the proof of Theorem 4.7, we have established that $\eta'(k, w) \geq \eta'(k, v)$, which leads to a contradiction. In this case, the equation $\eta'(k, v) = \eta'(k, w) = k\text{-}prob^-(u)$ must hold. That is, vertices $u$, $v$ and $w$ can potentially form a $(k, \eta')$-core, where $\eta'$ is the minimum between $k\text{-}prob^-(u)$ and $k\text{-}prob^-(v)$. Thus,

**Algorithm 4: Edge Deletion (ED)**

**Input:** $\mathcal{G}$, an edge $(u, v, p_{(u,v)})$ to be deleted
**Output:** the updated $\eta(k, w)$ for $w \in V_{\mathcal{G}}$ at each possible $k$

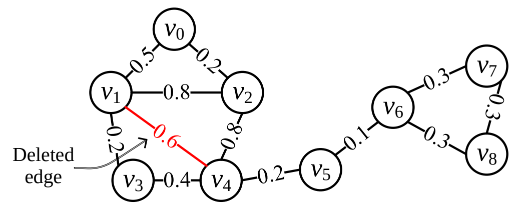
1 $\mathcal{G}' \leftarrow \mathcal{G} \setminus (u, v, p_{(u,v)})$;
2 $\mathcal{R}' \leftarrow [1, min\{c(u), c(v)\} - 1]$;
3 **foreach** $k \in \mathcal{R}'$ **do**
4     determine $lb$, $ub$ and the roots ($r[]$) using Theorem 4.8;
5     $C \leftarrow$ *FindDelete* $(\mathcal{G}', \varnothing, r[], lb, ub, k)$;
6     *UpdateInsert* $(k, C)$;
7 $k$++;     // critical $\mathcal{R}^-$ processing
8 $V_c \leftarrow$ *Deletion* $(G, u, v)$;     // [27]
9 **if** $V_c = \varnothing$ **then**
10     run lines 4-6;
11 **else**
12     $lb = 0$; $ub \leftarrow \max_{i \in V_c}\{\eta(k, i)\}$; $r[] \leftarrow \emptyset$;
13     **if** $c'(u) < k \wedge c'(v) \geq k$ **then**
14         $r[] \leftarrow v$ **if** $\eta(k, v) \leq ub$;
15     **else if** $c'(u) \geq k \wedge c'(v) < k$ **then**
16         $r[] \leftarrow u$ **if** $\eta(k, u) \leq ub$;
17     **else**
18         **if** *$c(i) < k$ for each vertex $i \in V_{\mathcal{G}}$* **then**
19             **return** *;*
20     $C \leftarrow$ *FindDelete* $(\mathcal{G}', V_c, r[], lb, ub, k)$;
21     *UpdateInsert* $(k, C)$;

$min\{k\text{-}prob^-(u), k\text{-}prob^-(v)\}$ serves as low-bound for both $u$ and $v$. This completely proves Theorem 4.8. □

Next, we formally propose the Edge Deletion (ED, Algorithm 4) algorithm for core maintenance in uncertain graphs. Firstly, Theorem 4.8 determines the range $\mathcal{R}^-$ of $k$ for edge deletions. Similarly, $\mathcal{R}^-$ is divided into two parts: $\mathcal{R}^- = \mathcal{R}' \cup \{k_0\}$, where $\mathcal{R}' = [1, k_0 - 1]$. For each $k \in \mathcal{R}'$, all vertices requiring updates can be correctly maintained through lines 4-6. Theorem 4.8 determines the $\eta$-threshold range and the root at an integer $k$. Especially, if $c(u) = c(v)$, ED handles two different cases: (1) if $u$ can reach $v$, all candidates can be found using a single root (either $u$ or $v$). (2) if $u$ cannot reach $v$, neither $u$ nor $v$ can find each other. Thus, both $u$ and $v$ are used as roots and added to the array $r$. Then, the algorithm **FindDelete** identifies the candidate set $C$ (line 5) and the same algorithm **UpdateInsert** updates the $\eta(k, w)$ for each vertex $w \in C$ (line 6). Since the edge deletion may decrease an old coreness for some vertices, we should delete the corresponding $\eta$-thresholds. Finally, critical $\mathcal{R}^-$ processing (lines 8-21) is used to handle this special case. We begin by providing a detailed description of critical $\mathcal{R}^-$ processing and then highlight the differences of FindDelete.

In the critical $\mathcal{R}^-$ processing, we initially utilize the core maintenance algorithm in [27] to update the vertices whose core numbers need to be decremented by 1. Let $V_c$ denote the set of these vertices (line 8). Next, we will discuss two distinct cases.

**Case-1:** $V_c$ is empty (lines 9-10). It indicates that the core numbers of all vertices remain unchanged. Thus, the initial subgraph used to calculate the $\eta(k_0, w)$ for each $w \in V_{\mathcal{G}}$ after the edge deletion only loses edge $(u, v, p_{(u,v)})$, compared with the initial subgraph before the edge deletion. This is similar to the case when $k \in \mathcal{R}'$, so that all vertices will be updated through lines 4-6.




(a) An example uncertain graph $\mathcal{G}'$

| $k$ | $v_0$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $v_7$ | $v_8$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.6 | 0.92 | 0.92 | 0.52 | 0.92 | 0.28 | 0.51 | 0.51 | 0.51 |
| 2 | 0.1 | 0.48 | 0.48 | 0.08 | 0.48 | 0.02 | 0.09 | 0.09 | 0.09 |

(b) Core decomposition result of $\mathcal{G}'$

**Figure 2: An example for edge deletion.**

**Case-2:** $V_c$ is not empty (lines 11-21). There are vertices whose core numbers decrease by 1. Let $u$ be one such vertex, and $c'(u)$ denotes its updated core number. Since $\eta(k_0, u)$ no longer exists, zero is considered as the low-bound ($lb$). Furthermore, the initial subgraph $\mathcal{G}_1$, which is used to calculate $\eta(k_0, w)$ for each vertex $w \in V_{\mathcal{G}}$, excludes $V_c$ and its adjacent edges after the edge deletion. However, the initial subgraph $\mathcal{G}_1$ must include the subgraph $\mathcal{G}_2$, which appears after determining the $\eta(k_0, w)$ value of each vertex $w \in V_c$. As the vertices with the smallest $k$-probability are successively removed, $\mathcal{G}_1$ inevitably evolves into $\mathcal{G}_2$. In other words, the $(k_0, \eta')$-core is determined, where $\eta'$ is defined as the maximum among $\eta(k, i)$ for $i \in V_c$, and it is set as the upper-bound ($ub$) (line 12). Furthermore, each vertex $w \in V_c$ is excluded from the $(k_0, \eta)$-core, which directly affects the neighboring vertices of $w$. Thus, all vertices in $V_c$ become new roots. We then consider three cases. (i) if $u \in V_c$ and $v \notin V_c$ (lines 13-14), we examine whether $\eta(k_0, v)$ is less than or equal to $ub$. If this condition holds, $v$ is recorded as a separate root. (ii) if $u \notin V_c$ and $v \in V_c$ (lines 15-16), the situation is similar to case (i), and the same process is followed. (iii) if $u \in V_c$ and $v \in V_c$ (lines 17-19), $V_c$ can be directly used to identify all candidates. Furthermore, if the core number of each vertex is below $k_0$, there is no $(k_0, \eta)$-core. Hence, this step can be skipped.

Finally, the **FindDelete** algorithm adopts a similar approach as Algorithm 2. The detailed pseudocode is removed to the appendix due to space constraints. After the deletion, there may be multiple roots, represented by $V_c$ and $r$. The neighbor $x$, which is affected by $w \in V_c$ and has a $\eta(k, x)$ value within the $\eta$-threshold range, is added to the queue $Q$. If a root in the array $r$ has not been added to $Q$, it is included directly. The initialization of $Q$ is now complete. The subsequent steps follow the same approach as Algorithm 2.

Example 4.2. *In Figure 2, let $(v_1, v_4, 0.6)$ be the edge to be deleted. Since $c(v_1) = c(v_4) = 2$, $\mathcal{R}^- = [1, 2]$ can be determined. Next, each $k \in \mathcal{R}^-$ will be processed sequentially. For $k = 1$, the original $\eta(1, v_1)$ and $\eta(1, v_4)$ are both 0.92. Using Theorem 4.8, we find $ub = 0.92$, $r = \{v_1, v_4\}$, and $lb = k\text{-}prob(v_1, \mathcal{G}_1^{v_6 -}) = k\text{-}prob(v_4, \mathcal{G}_1^{v_4 -}) = 0.8$. A BFS traversal is performed starting from $v_1$ and $v_4$ and the candidate set $C = \{v_1, v_2, v_4\}$ can be identified and updated. When $k = 2$, the Deletion algorithm returns $V_c = \emptyset$, which corresponds to case-1. Similarly, We can identify $ub = 0.48$, $r = \{v_1, v_4\}$, and $lb = 0$. Consequently, candidate set $C = \{v_0, \ldots, v_8\}$ is discovered and subsequently updated.*

## 4.3 Theoretical Analysis

**Complexity Analysis.** For each $k \in \mathcal{R}^+/\mathcal{R}^-$, if $\eta(k,u) < \eta(k,v)$, the total time complexity of Algorithm 1 or Algorithm 4 is $O(k_0 \cdot (|C|\log|C| + |V_c|\sum_{u \in V_c} deg(u)) + k_0^2 \cdot \sum_{u \in C} deg^2(u))$. Due to space constraints, a detailed complexity analysis will be provided in the appendix. Here, $V_c$ is bounded by $C$ and $k_0$ is bounded by the minimum core number of $u$ and $v$. Additionally, the size of $C$ is bounded by the maximum subgraph. Each vertex $w$ in the subgraph is connected to the root, and has a $\eta(k,w)$ value within the $\eta$-threshold range. Therefore, in practice, the maintenance algorithm is highly efficient since the visited subgraph during the search process is usually much smaller than $\mathcal{G}$, and $k_0$ is typically smaller than the largest core number in the deterministic graph.

**Boundedness Analysis.** For the proposed $(k,\eta)$-core maintenance problem, the set of vertices whose $\eta$-thresholds change is represented as CNG. If an incremental algorithm $\mathcal{A}$ is *bounded* [43], its cost should be a polynomial function of the size of CNG's $c$-hop neighbors, denoted as $||\text{CNG}||_c$, where $c$ is a positive integer. Based on this, we introduce the following Theorem 4.9.

Theorem 4.9. *The $(k,\eta)$-core maintenance algorithm is unbounded for both edge insertion and edge deletion in uncertain graphs.*

Next, we conduct a separate analysis of the two cases. Firstly, we focus on edge deletions. The removal algorithm for core maintenance described in [45] is a bounded algorithm. Its effectiveness stems from the fact that, in an undirected and unweighted graph, we can efficiently prune the BFS process by directly computing the maximum-core degree and purecore degree of $u$'s neighbors through a simple decrement operation. However, in uncertain graphs, the DP algorithm needs to be employed to update the $k$-probability values for both the root $u$ and its neighbors. The frequent utilization of DP during the BFS process may not yield significant pruning benefits compared to the additional computational overhead. Hence, our algorithm considers all vertices connected to the root and falling within the $\eta$-threshold range, without relying on a polynomial function of $||\text{CNG}||_c$. Secondly, we focus on edge insertions. Previous studies have demonstrated that both $k$-truss maintenance and $k$-core maintenance exhibit asymmetry concerning edge insertions and deletions [56]. While no maintenance algorithm with complexity dependent on CNG is currently known for edge insertions, the complexity of the $(k,\eta)$-core definition poses a challenge in devising a linear time maintenance algorithm.

# 5 OPTIMIZATIONS

This section presents three optimizations aimed at enhancing the efficiency of our basic core maintenance algorithm. Firstly, we narrow down the $\eta$-threshold range at an integer $k$. Secondly, we utilize the vertices visited during the BFS traversal to enable further pruning. Thirdly, we adopt a lazy computing approach to minimize redundant calculations.

## 5.1 Bounds Contraction

**Upper-bound Contraction in Edge Insertion**. Let's revisit the core update Theorem 4.4, which applies to the insertion of an edge $(u, v, p_{(u,v)})$ in uncertain graphs. Given an integer $k$, if $\eta(k,u)$ is less than $\eta(k,v)$, the upper-bound $k\text{-}prob^+(u) = k\text{-}prob(u, \mathcal{G}_k^{u+})$ can be determined. Here, $\mathcal{G}_k^{u+}$ simultaneously considers the initial induced $\eta$-threshold subgraph of vertex $u$ and the inserted edge. By reducing the size of $\mathcal{G}_k^{u+}$, the upper-bound diminishes, resulting in a smaller $\eta$-threshold range. Following the iterative principle of the peeling algorithm, we prioritize the vertex with the minimum $k$-probability. Based on this, we provide the subsequent definition.

Definition 5.1. **(Restricted Vertex)** *Assume that an edge $(u, v, p_{(u,v)})$ is inserted into an uncertain graph $\mathcal{G}$, where $\eta(k,u) < \eta(k,v)$. Given an integer $k$, a restricted vertex refers to a neighbor $w$ of vertex $u$, satisfying $\eta(k,w) = \eta(k,u)$ and $k\text{-}prob(w, \mathcal{G}_k^w) = \eta(k,w)$.*

Given an integer $k$, the insertion of an edge $(u, v, p_{(u,v)})$ does not introduce a new neighbor to $w$. Thus, $k\text{-}prob(w, \mathcal{G}_k^{w+})$ remains the same as $k\text{-}prob(w, \mathcal{G}_k^w)$, where $k\text{-}prob(w, \mathcal{G}_k^{w+})$ serves as the upper-bound for the updated $\eta(k,w)$. If $k\text{-}prob(w, \mathcal{G}_k^w)$ equals $\eta(k,w)$, then $\eta(k,w)$ remains unchanged. However, if $\eta(k,u)$ increases, $w$ cannot be included in the updated $\eta$-threshold subgraph of $u$, denoted as $\mathcal{G}_k^{u+}$. We then present the following Lemma.

Lemma 5.1. *Assume that an edge $(u, v, p_{(u,v)})$ is inserted into an uncertain graph $\mathcal{G}$. Given an integer $k$, the $\eta(k,w)$ value of the restricted vertex $w$ remains unchanged.*

Therefore, it is crucial to identify and exclude such restricted vertices, as this reduces the size of $\mathcal{G}_k^{u+}$ and diminishes the upper-bound. This reduction leads to a smaller $\eta$-threshold range, thereby enhancing the efficiency of Algorithm 1. Let us reconsider Example 4.1. When $k = 1$, $v_6$ becomes the root. $v_5$ and $v_7$ are neighbors of $v_6$ with $\eta(1,v_5) = \eta(1,v_7) = \eta(1,v_6)$. By computing $k\text{-}prob(v5, \mathcal{G}_1^{v_5+})$ = 0.706 = $\eta(1,v_5)$ and $k\text{-}prob(v7, \mathcal{G}_1^{v_7+}) = 0.888 > \eta(1,v_7)$, we identify $v_5$ as a restricted vertex. Then, $ub = k\text{-}prob(v_6, \mathcal{G}_1^{v_6+}\backslash\{v_5\}) =$ 0.76 can be computed, which determines the $\eta$-threshold range $[0.706, 0.76)$. This range is a subset of the initial range $[0.706, 0.856)$. By conducting a BFS traversal starting from $v_6$, we identify the candidate set $C = \{v_6, v_5, v_7\}$. Finally, $\eta(1,v_6)$ and $\eta(1,v_7)$ are also updated to 0.76. This example demonstrates the case when $k = 1$.

Furthermore, it is essential for the lower-bound ($lb$) and upper-bound ($ub$) of the $\eta$-threshold range to satisfy $lb < ub$. If $lb \geq ub$, it implies that $u$ cannot join a $(k,\eta')$-core ($\eta' > \eta(k,u)$) without the assistance of restricted vertices. Thus, it is possible to identify opportunities for early termination of the current $k$ processing.

**Lower-bound Contraction in Edge Deletion**. Let us now revisit the core update Theorem 4.8, which applies to the deletion of an edge $(u, v, p_{(u,v)})$ in uncertain graphs. Given an integer $k$, if $\eta(k,u) < \eta(k,v)$, we can identify the lower-bound $k\text{-}prob^-(u) = k\text{-}prob(u, \mathcal{G}_k^{u-})$. Here, $\mathcal{G}_k^{u-}$ represents the updated $\eta$-threshold subgraph of $u$, and $N_k^{u-}$ denotes the updated neighbor set of $u$ in $\mathcal{G}_k^{u-}$. We have two situations to consider: (i) $|N_k^{u-}| \geq k$. This implies that $k\text{-}prob^-(u)$ is non-zero and can be served as the lower-bound ($lb$). (ii) $|N_k^{u-}| < k$. This means that $k\text{-}prob^-(u)$ is zero. Using zero as the $lb$ would result in a large number of candidates. By expanding the size of $\mathcal{G}_k^{u-}$, it is possible to increase the lower-bound, thereby leading to a smaller $\eta$-threshold range. Hence, our objective is to introduce additional neighbors $w$ to $\mathcal{G}_k^{u-}$ with the highest $\eta(k,w)$. Before delving into the optimization of lower-bound contraction, we first provide the following definition.

Definition 5.2. **(Potential Set)** *Assume that an edge $(u, v, p_{(u,v)})$ is deleted from an uncertain graph $\mathcal{G}$, where $\eta(k, u) < \eta(k, v)$. Given an integer $k$, the potential set of vertex $u$, denoted as $P_k^u$, contains neighbors $w$ of $u$, satisfying the conditions $w \notin N_k^{u-}$ and $c(w) \geq k$.*

If $|N_k^{u-}| < k$, We can expand $\mathcal{G}_k^{u-}$ through a greedy algorithm. The vertices in $P_k^u$ are sorted in a non-increasing order based on their $\eta$-thresholds. Each vertex $w_i \in P_k^u$ is sequentially added to $\mathcal{G}_k^{u-}$, and $k\text{-}prob(u, \mathcal{G}_k^{u-} \cup w_{1,\ldots,i})$ is updated accordingly. Next, Lemma 5.2 gives the stopping condition for the algorithm.

Lemma 5.2. *Given an integer $k$, when we add the vertices $w_i$ in the non-increasing set $P_k^u$ to $\mathcal{G}_k^{u-}$, the $\eta(k, w_i)$ value of the vertex $w_i$ remains unchanged if $k\text{-}prob(u, \mathcal{G}_k^{u-} \cup w_{1,\ldots,i}) \geq \eta(k, w_i)$.*

Obviously, some vertices in $P_k^u$ might belong to the same $(k, \eta)$-core as $u$. Once $k\text{-}prob(u, \mathcal{G}_k^{u-} \cup w_{1,\ldots,i})$ becomes greater than or equal to $\eta(k, w_i)$, it signifies that $N_k^{u-} \cup w_{1,\ldots,i}$ is at least a $(k, \eta(k, w_i))$-core. Consequently, $\eta(k, w_i)$ remains unchanged and serves as the lower-bound ($lb$). If all vertices in $P_k^u$ have been processed without satisfying Lemma 5.2, then $lb$ remains zero. Let's reconsider Example 4.2. When $k = 2$, $v_1$ and $v_4$ become the roots. The updated neighbor sets $N_2^{v_1-}$ and $N_2^{v_4-}$ both exclusively contain the vertex $v_2$ and have sizes smaller than 2. The potential set of $v_1$ is $P_2^{v_1} = \{v_0, v_3\}$. We add $v_0$ to $\mathcal{G}_2^{v_1-}$ and compute $k\text{-}prob(v_1, \mathcal{G}_2^{v_1-} \cup v_0) = 0.4 > \eta(2, v_0)$. Similarly, the potential set of $v_4$ is $P_2^{v_4} = \{v_3, v_5\}$. We add $v_3$ to $\mathcal{G}_2^{v_4-}$ and compute $k\text{-}prob(v_4, \mathcal{G}_2^{v_4-} \cup v_3) = 0.32 > \eta(2, v_3)$. According to Theorem 4.8, the smaller value between $k\text{-}prob^-(v_4)$ and $k\text{-}prob^-(v_3)$ is taken as $lb$. Consequently, the $\eta$-threshold range $(0.08, 0.48]$ can be determined, which is a a subset of the initial range $(0, 0.48]$. The candidate set $C = \{v_0, v_1, v_2, v_4\}$ is found and updated.

In addition, if $P_k^u$ contains a large number of vertices, performing multiple updates of $k\text{-}prob^-(u)$ can introduce additional computational overhead. To mitigate this, we can utilize Equation 16 outlined in [13] for incremental updates. When $\eta(k, w_i)$ becomes the $lb$, it implies that $w_i$ will not be included in $C$. The initial calculation of $k\text{-}prob(u)$ in line 2 of Algorithm 3 may yield a value smaller than $\eta(k, w_i)$. However, in such case, $w_i$ and $u$ must both belong to at least the $(k, \eta(k, w_i))$-core, which is contradictory. Hence, it is crucial to set the initial value of *low* in Algorithm 3 to $\eta(k, w_i)$.

## 5.2 Dynamic Update

**Incremental Update in Edge Insertion**. Algorithm 2 conducts a BFS traversal to identify all candidates reachable from the root through vertices within the $\eta$-threshold range. It returns the candidate set $C$. For an given integer $k$, if $c$ is a vertex in the candidate set $C$, the insertion of an edge has the potential to increase the $\eta(k, c)$ value. Now, assuming $w$ is a neighbor of $c$ and $\eta(k, w) < \eta(k, c)$. Since $w$ does not add a new neighbor, the equality $k\text{-}prob(w, \mathcal{G}_k^{w+}) = k\text{-}prob(w, \mathcal{G}_k^{w})$ still holds. Based on the basic iterative steps of the peeling algorithm, vertex $w$ must be processed prior to $c$, ensuring that $\eta(k, w)$ remains unchanged. Consequently, we formally propose Lemma 5.3 as follows.

Lemma 5.3. *Assume that an edge is inserted into an uncertain graph $\mathcal{G}$. Given an integer $k$, a pre-determined $\eta$-threshold range $[lb, ub)$ and the root, a candidate must be connected to the root through a non-increasing path containing only vertices within $[lb, ub)$.*

According to Lemma 5.3, when processing a candidate vertex $c$ at an integer $k$, it is only necessary to consider its neighbors $w$ with $\eta(k, w) \geq \eta(k, c)$. To reduce the size of candidate set $C$, we dynamically update the lower-bound (lb) to the $\eta(k, c)$ value of the currently processed vertex $c$. Let's reconsider Example 4.1, where the update range $[0.706, 0.856)$ is determined. The BFS traversal starts from $v_6$ with $lb = 0.706$. Then, $v_4$, $v_5$ and $v_7$ are identified and added to the candidate set $C$. Upon processing $v_4$, $lb$ is updated to 0.8, and $v_3$ is identified. When $v_3$ is processed, $lb = 0.8$ prevents $v_1$ from being discovered and added to $C$. Finally, $C = \{v_6, v_4, v_5, v_7, v_3\}$ is determined and $C$ is a subset of the initial candidate set.

**Decremental Update in Edge Deletion**. For an given integer $k$, let $c$ be a vertex in the candidate set $C$, indicating that the deletion of an edge has the potential to decrease the $\eta(k, c)$ value. If vertex $w$ is a neighbor of $c$ and $\eta(k, w) > \eta(k, c)$, the initial $\eta$-threshold subgraph of $w$, denoted as $\mathcal{G}_k^{w}$, does not include $c$. Furthermore, the edge deletion cannot decrease the $\eta(k, w)$ value. Consequently, $c$ cannot be part of the updated $\eta$-threshold subgraph of $w$, denoted as $\mathcal{G}_k^{w'}$. In other words, the $\eta(k, w)$ value will not be affected by $c$. Consequently, we formally propose Lemma 5.4 as follows.

Lemma 5.4. *Assume that an edge is deleted from an uncertain graph $\mathcal{G}$. Given an integer $k$, a pre-determined $\eta$-threshold range $(lb, ub]$ and the root, a candidate must be connected to the root through a non-decreasing path containing only vertices within $(lb, ub]$.*

Based on Lemma 5.4, when processing a candidate vertex $c$ at an integer $k$, we only need to consider its neighbors $w$ with $\eta(k, w) \leq \eta(k, c)$. It is crucial to dynamically update the upper-bound ($ub$) to the $\eta(k, c)$ value of the vertex $c$ being processed. This decremental update helps obtain a smaller candidate set $C$, thereby enhancing the efficiency of Algorithm 4. Let's reconsider Example 4.2. When $k = 2$, the $\eta$-threshold range $(0, 0.48]$ has been determined. Starting with $v_1$ and $ub = 0.48$, a BFS traversal identifies $v_0$, $v_2$ and $v_3$, which are then added to the candidate set $C$. Processing $v_2$ leads to the inclusion of $v_4$ in the set $C$. Similarly, processing $v_4$ adds $v_5$ to $C$. However, $v_6$ is not discovered and added to $C$ due to the updated $ub$ = 0.02 when processing $v_5$. Furthermore, another root $v_4$ has been processed. Finally, $C = \{v_0, \ldots, v_5\}$ is determined, which is a subset of the initial candidate set.

## 5.3 Lazy Computation

Algorithm 3 sequentially processes the vertex with the minimum $k$-probability. However, if two vertices $v_1$ and $v_2$ are processed successively, their common neighbor needs to be recalculated twice, which results in additional computational overhead. To address this issue, we propose a lazy computation optimization in Algorithm 5.

The main objective of Algorithm 5 is to identify each vertex $v$ with determinable $\eta(k, v)$ and add $v$ to queue $D$ initially (line 5). If $D$ is empty, only the vertex with the smallest $k$-probability is found (lines 6-7). During the removal phase, the two mentioned cases are handled separately. When $D$ is non-empty, all vertices in $D$ are updated in batches, and their neighbors are pushed into the set $S$ in which each vertex appears only once (lines 9-12). If $D$ is empty, the vertex with the smallest $k$-probability is processed, similar to

**Algorithm 5: LazyUpdate**

**Input:** $k$-value: $k$, candidate set: $C$
**Output:** the updated $\eta(k, w)$ for each vertex $w \in C$

1 $D \leftarrow$ empty queue; $S \leftarrow$ empty set; $curThres \leftarrow 0$;
2 compute $k\text{-}prob(v)$ for each $v \in C$;
3 **while** $C \neq \varnothing$ **do**
4   **if** $D = \varnothing$ **then**
5     $D \leftarrow \{v \in C | k\text{-}prob(v) \leq curThres\}$;
6     **if** $D = \varnothing$ **then**
7       $u \leftarrow arg\ min_{v \in C} k\text{-}prob(v)$;
8   **if** $D \neq \varnothing$ **then**
9     $S \leftarrow \bigcup_{v \in D} \{u \in (N_v \cap C) | k\text{-}prob(u) > curThres\}$;
10     $\eta(k, v) = curThres$ for all $v \in D$;
11     $C \leftarrow C \setminus D$; $\quad D \leftarrow \varnothing$;
12     **foreach** $v \in S$ **do**
13       update $k\text{-}prob(v)$;
14       **if** $k\text{-}prob(v) \leq curThres$ **then**
15         $D$.push($v$);
16   **else**
17     lines 5-9 of Algorithm 3, and add above judgments (lines 14-15);

Algorithm 3. Obviously, Algorithm 5 recalculates the $k$-probability of a common neighbor vertex at most once, significantly improving the overall execution efficiency of EI (Algorithm 1) and DI (Algorithm 4). Furthermore, an additional condition is employed (lines 15-16, 18), where any neighbor vertex with a $k$-probability smaller than the current $curThres$ is added to $D$. Then, all vertices $w \in D$ have a determinable $\eta(k, w)$ and can be processed in batch, thereby further enhancing the execution efficiency. It should be emphasized that if EI utilizes the optimization of lower-bound contraction, the initial value of the $lb$ must be set accordingly.

## 6 PROBABILITY CHANGE

In this section, we provide a brief description of the core maintenance algorithms for increasing and decreasing probability values in uncertain graphs, respectively.

### 6.1 Probability Increase

Assuming the probability of edge $(u, v)$ increases to $p^+_{(u,v)}$ (where $p^+_{(u,v)} \leq 1$), then the core number of all vertex remains unaffected. The influence range of $k$, denoted as $\mathcal{R}' = [1, k_0]$, can be determined, where $k_0 = min\{c(u), c(v)\}$. A similar proof can be found in Theorem 4.1 and is omitted here. Given an integer $k \in \mathcal{R}'$ and $\eta(k, u) < \eta(k, v)$, $\mathcal{G}^u_k$ represent the induced $\eta$-threshold subgraph of vertex $u$. After increasing the probability, the updated subgraph of $u$ can still be represented by $\mathcal{G}^{u+}_k$. It is evident that $k\text{-}prob^+(u) > k\text{-}prob(u)$ also holds. Consequently, in this case, Theorem 4.4 also applies to the core maintenance of uncertain graphs. The only difference lies in modifying $\mathcal{R}^+$ to $\mathcal{R}'$.

Recall that algorithm EI divided $\mathcal{R}^+$ into two parts, denoted as $\mathcal{R}^+ = \mathcal{R}' \cup \{k_0+1\}$. When $k = k_0+1$, the core number of each vertex remains unchanged, eliminating the need for critical $\mathcal{R}^+$ processing. For each $k$ belonging to $\mathcal{R}'$, the $\eta(k, w)$ of each vertex $w \in V_{\mathcal{G}}$ can be correctly maintained through lines 4-6 in Algorithm 1. Thus, after increasing the edge probability, a simplified maintenance algorithm **PI** can be used for uncertain graphs. In the worst case, the total time complexity of PI is $O(k_0 \cdot |C| \log |C| + k_0^2 \cdot \sum_{u \in C} deg^2(u))$. In addition, the three optimizations described in Section 5 can also be applied to speed up this basic algorithm.

### 6.2 Probability Decrease

Assume that the probability of edge $(u, v)$ decreases to $p^-_{(u,v)}$ (where $p^-_{(u,v)} \geq 0$) and $\eta(k, u) < \eta(k, v)$. The influence range of $k$ can be determined as $\mathcal{R}^- = [1, k_0]$, where $k_0 = \{c(u), c(v)\}$. A similar proof can be found in Theorem 4.5 and is omitted here. Given an integer $k \in \mathcal{R}^-$, $\mathcal{G}^{u-}_k$ can still represent the updated $\eta$-threshold subgraph of $u$. It is evident that $k\text{-}prob^-(u) < k\text{-}prob(u)$ holds as well. Consequently, Theorem 4.8 also applies to the core maintenance of uncertain graphs when decreasing the edge probability.

Recall that algorithm DI divided $\mathcal{R}^-$ into two parts: $\mathcal{R}^- = \mathcal{R}' \cup \{k_0\}$, where $\mathcal{R}' = [1, k_0 - 1]$. However, the core number of the vertex remains unchanged. When $k = k_0$, the only requirement is to maintain the $\eta(k, w)$ of each vertex $w \in V_{\mathcal{G}}$. In summary, for each $k$ belonging to $\mathcal{R}^-$, the core maintenance of the uncertain graph can be accomplished through lines 1-6 in Algorithm 4. The only difference is to replace $\mathcal{R}'$ in Algorithm 4 with $\mathcal{R}^-$. This basic maintenance algorithm is referred to as **PD**. In the worst case, the total time complexity of PD is $O(k_0 \cdot |C| \log |C| + k_0^2 \cdot \sum_{u \in C} deg^2(u))$. We can also use the three optimizations in Section 5 to speed up this basic algorithm.

## 7 EXPERIMENTS

This section evaluates the performance of our proposed algorithms. All algorithms are implemented in C++, and compiled by G++ with -O3 optimization. The experiments are conducted on CentOS Linux 7 Core with Intel Xeon 2.40GHz and 125G main memory.

### 7.1 Experimental Settings

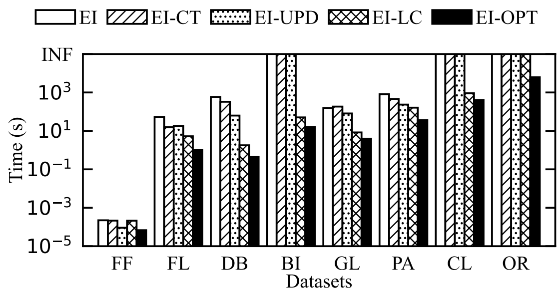
**Datasets:** We employ eight real-world graphs in experiments, which are summarized in Table 1. Specifically, Fruit-Fly[2] and Biomine[3] are protein-protein interaction networks, where the probability of an edge represents the possibility that the interaction of two proteins exists. Flickr[4] is an online community for sharing photos, where the probability of an edge is set to the Jaccard coefficient of the interest communities shared by the two users. DBLP[5] is a co-authorship network, where the probability of an edge is determined by an exponential function to the number of collaborations. Google is a web graph that was released in 2002 by Google. Patents is a citation network spanning 37 years (1963 to 1999), consisting of 3,923,922 utility patents granted during that period. Comlj and Orkut are online communities that provide integrated functionalities such as forums and blogs. Google, Patents, Comlj, and Orkut are from SNAP[6], and their edge probabilities are generated uniformly at the range [0, 1] as with [12, 35].

**Algorithms:** We evaluate a set of algorithms in experiments.

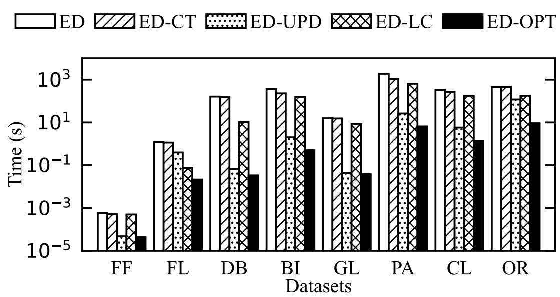
[2] http://thebiogrid.org
[3] https://biomine.ijs.si/downloads/
[4] https://www.flickr.com
[5] https://dblp.uni-trier.de
[6] https://snap.stanford.edu/data/index.html

**Table 1: Uncertain graphs ($d_{max}$: maximum degree; $k_{max}$: maximum core number)**

| Dataset (Abbr.) | $\lvert V_{\mathcal{G}} \rvert$ | $\lvert E_{\mathcal{G}} \rvert$ | $d_{max}$ | $k_{max}$ |
|---|---|---|---|---|
| Fruit-fly (*FF*) | 3,752 | 3,692 | 27 | 4 |
| Flickr (*FL*) | 24,125 | 300,836 | 546 | 225 |
| DBLP (*DB*) | 684,911 | 2,284,991 | 611 | 114 |
| Biomine (*BI*) | 1,008,201 | 6,722,513 | 139624 | 448 |
| Google (*GL*) | 875,713 | 5,105,039 | 6332 | 44 |
| Patents (*PA*) | 3,774,768 | 16,518,948 | 793 | 64 |
| Comlj (*CL*) | 3,997,962 | 34,681,189 | 14815 | 360 |
| Orkut (*OR*) | 3,072,441 | 117,185,083 | 33313 | 253 |




(a) Edge insertion




(b) Edge deletion

**Figure 3: Evaluation of optimizations**

- **BASE**: The core decomposition algorithms, i.e., the peeling algorithm without dynamic update technology [12, 53].
- **EI** and **ED**: Our proposed basic algorithms for edge insertion and deletion, respectively.
- **PI** and **PD**: Our proposed basic algorithms for edge probability increase and decrease, respectively.
- **Three optimizations**. **-CT**: contraction optimization. **-UPD**: update optimization. **-LC**: lazy computation optimization. **-OPT**: all three optimizations.

**Parameters and Metrics:** The evaluated parameters include the number of updated edges, the change of probability values , and the uncertain graph size. In experiments, we randomly select 500 edges to simulate uncertain graphs update and report the average running time. The algorithm terminates if it does not complete within $10^4$ seconds. Due to space limitations and similar trends over different uncertain graphs, we report experimental results of partial datasets in this paper.

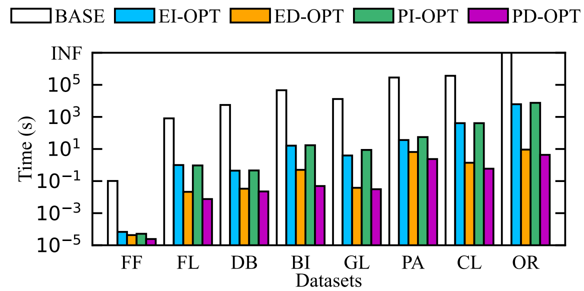
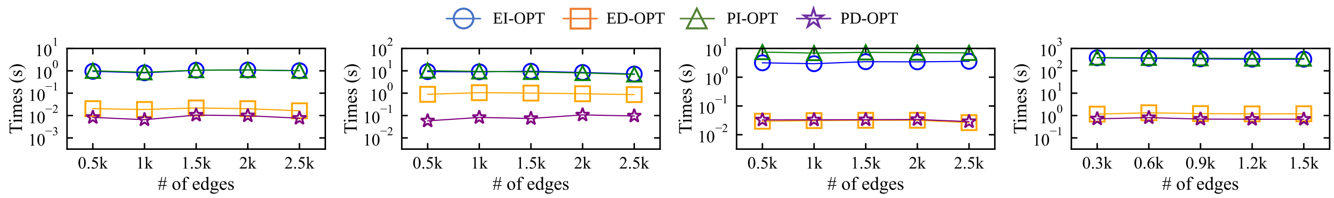



**Figure 4: Evaluation of optimal algorithms.**

## 7.2 Experimental Results

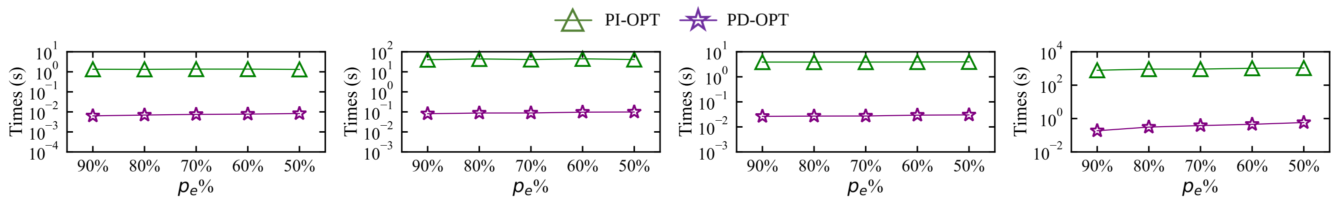
**EXP-1: Evaluation of Optimizations.** Firstly, we evaluate the efficiency of the three optimizations, i.e., -CT, -UPD, and -LC. To this end, we integrate the optimizations into four basic core maintenance algorithms, i.e., EI, ED, PI, and PD. Since PI and PD have similar results with EI and ED, they are removed to our the appendix [10] due to space limitation. Figure 3 shows the results. We can observe that the three optimizations have different improvements over basic algorithms. Specifically, Figure 3(a) depicts the time of edge insertion on different datasets. EI-LC demonstrates the best performance on all datasets except FF. For instance, for DB dataset, EI-LC, EI-UPD, and EI-CT are 329.3, 9.4, and 1.8 times faster than EI, respectively, showing that the lazy computation optimization significantly improves the efficiency of EI. Figure 3(b) reports the time for edge deletion, where ED-UPD has a greater performance improvement for ED than the other two optimizations. For example, for DB dataset, ED-UPD, ED-LC, and ED-CT are 2295.6, 15.7, and 1.1 times faster than ED, respectively. In BFS traversal, the $\eta$-threshold of the currently visited vertex can be used to further prune the vertices whose $\eta$-threshold do not need updates, improving the efficiency of ED. Moreover, the three optimizations together also make basic core maintenance algorithms EI and ED more efficient. For instance, for DB dataset, EI-OPT and ED-OPT exhibit significantly improved efficiency compared to EI and ED, with speedups of 1317.4 and 4829.8 times, respectively. Note that, in the remainder of the experiments, we only report our proposed algorithms with all optimizations.

**EXP-2: Base VS. Our Proposed Optimized Algorithms.** In this experiment, we compare our proposed optimized algorithms with baseline, which is to perform core decomposition from the scratch when the uncertain graphs update. The results are shown in Figure 4. It is evident that all algorithms we proposed exhibit a significant improvement across all datasets in terms of running time compared to BASE. Specially, EI-OPT and PI-OPT are at least three orders of magnitude faster than BASE. ED-OPT and PD-OPT outperform BASE by at least four orders of magnitude. In the best case, our proposed optimized algorithms are up to six orders of magnitude faster than BASE. For example, for GL dataset, ED-OPT can process an edge deletion in 0.04 second, while BASE takes 13,080 second. The reason for this significant speed improvement is that BASE should handle the whole uncertain graph size for each update while our proposed algorithms only need to process the induced $\eta$-threshold subgraph, which is small.



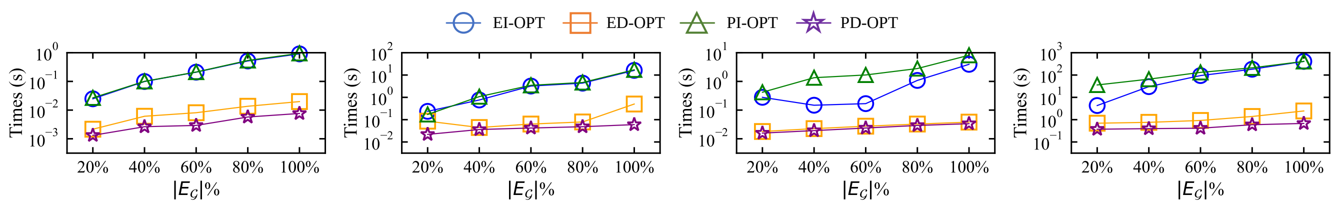

(a) FL (b) BI (c) GL (d) CL

**Figure 5: Effect of the number of updated edges (average time)**




(a) FL (b) BI (c) GL (d) CL

**Figure 6: Effect of $p_e$**




(a) FL (b) BI (c) GL (d) CL

**Figure 7: Scalability evaluation**

**EXP-3: Effect of the Number of Updated Edges.** To test the effect of the number of updated edges on average processing time, a sequence of $r$ updates is performed. For most graphs, $r$ is selected from the set $\{500, 1000, 1500, 2000, 2500\}$, while for CL and OR, the set $\{300, 600, 900, 1200, 1500\}$ is used. The experimental results, shown in Figure 5, indicate that as the number of updated edges increases, the average update time almost keep the same. It is because the proposed maintenance algorithms incorporate two distinct parameters, namely, the vertex and $k$, where the number of vertices and $k$ exhibit opposite trends. For example, if $k$ is relatively large, only a few vertices may be included in a $(k, \eta)$-core, which results in a smaller subgraph needing updates. Thus, in a broader range of $\mathcal{R}^+/\mathcal{R}^-$, processing larger $k$ values incurs minimal update costs compared to processing smaller $k$ values, which ensures the stability of our update algorithm.

**EXP-4: Effect of $p_e$.** We study the effect of $p_e$ on PI-OPT and PD-OPT by varying $p_e$ from $p_e{\cdot}50\%$ to $p_e{\cdot}90\%$. The experimental results, shown in Figure 6, indicate that when $\Delta p_e$ increases, the average update time grows by a small amount and remains almost stable. The main reason is as follows. Larger $\Delta p_e$ results in larger upper-bounds and smaller lower-bounds for PI and PD, respectively. Consequently, the $\eta$-threshold update range becomes larger, requiring updates larger subgraph.

**EXP-5: Scalability Evaluation.** Finally, we evaluate the scalability of our proposed algorithms. To this end, we randomly select 20%, 40%, 60%, 80%, and 100% of edges from original uncertain graphs, and then get the corresponding subgraphs induced by these edges. The experimental results are shown in Figure 7. As expected, the running time of all algorithms grows when the graphs become larger. It is because the increase in graph size leads to higher vertex degrees, resulting in increased computational cost for $k$-probability and BFS traversal cost. Additionally, vertices in dense graphs may have larger core numbers, leading to a larger $\mathcal{R}^+/\mathcal{R}^-$ and more running time. Moreover, we can observe that the performance of ED-OPT and PD-OPT degrades more slightly than that of EI-OPT and PI-OPT. The reason behind is that ED-OPT and PD-OPT can locate smaller subgraphs, thereby demonstrating superior performance (as shown in Figure 3). As the graph size increases, smaller subgraphs require less expansion compared to larger subgraphs. Thus, ED and PD exhibit excellent scalability in larger graphs.

## 8 CONCLUSIONS

In this paper, we study the problem of core maintenance in dynamic uncertain graphs. We propose core maintenance algorithms with three optimizations for four update operations, including inserting/deleting edge and increasing/decreasing edge probability. Extensive experimental evaluations demonstrate the efficiency of our proposed algorithms. In the future, we would like to investigate parallel maintenance algorithms in large uncertain graphs and study the maintenance of other cohesive subgraphs (e.g., $k$-truss) in uncertain graphs.

# REFERENCES


[1] Eytan Adar and Christopher Re. 2007. Managing uncertainty in social networks. *IEEE Data Eng. Bull.* 30, 2 (2007), 15–22.
[2] Charu C Aggarwal. 2013. *Managing and mining sensor data.* Springer.
[3] Sabeur Aridhi, Martin Brugnara, Alberto Montresor, and Yannis Velegrakis. 2016. Distributed k-core decomposition and maintenance in large dynamic graphs. In *DEBS*. 161–168.
[4] Joel S Bader, Amitabha Chaudhuri, Jonathan M Rothberg, and John Chant. 2004. Gaining confidence in high-throughput protein interaction networks. *Nat. Biotechnol.* 22, 1 (2004), 78–85.
[5] Wen Bai, Yuncheng Jiang, Yong Tang, and Yayang Li. 2023. Parallel core maintenance of dynamic graphs. *IEEE Trans. Knowl. Data Eng.* 35, 9 (2023), 8919–8933.
[6] Wen Bai, Yuxiao Zhang, Xuezheng Liu, Min Chen, and Di Wu. 2020. Efficient core maintenance of dynamic graphs. In *DASFAA*. 658–665.
[7] Sanjit Biswas and Robert Morris. 2005. ExOR: Opportunistic multi-hop routing for wireless networks. In *SIGCOMM*. 133–144.
[8] Paolo Boldi, Francesco Bonchi, Aristides Gionis, and Tamir Tassa. 2012. Injecting uncertainty in graphs for identity obfuscation. *Proc. VLDB Endow.* 5, 11 (2012), 1376–1387.
[9] Francesco Bonchi, Francesco Gullo, Andreas Kaltenbrunner, and Yana Volkovich. 2014. Core decomposition of uncertain graphs. In *SIGKDD*. 1316–1325.
[10] Yu Chen, Qing Liu, and Yunjun Gao. 2025. Efficient core maintenance in dynamic uncertain graphs. https://github.com/ayrnb/ZJU/tree/main/paper/VLDB2025.pdf
[11] Qiangqiang Dai, Rong-Hua Li, Meihao Liao, Hongzhi Chen, and Guoren Wang. 2022. Fast maximal clique enumeration on uncertain graphs: A pivot-based approach. In *SIGMOD*. 2034–2047.
[12] Qiangqiang Dai, Rong-Hua Li, Guoren Wang, Rui Mao, Zhiwei Zhang, and Ye Yuan. 2021. Core decomposition on uncertain graphs revisited. *IEEE Trans. Knowl. Data Eng.* 35, 1 (2021), 196–210.
[13] Fatemeh Esfahani, Venkatesh Srinivasan, Alex Thomo, and Kui Wu. 2019. Efficient computation of probabilistic core decomposition at web-scale. In *EDBT*. 325–336.
[14] Terrill L Frantz, Marcelo Cataldo, and Kathleen M Carley. 2009. Robustness of centrality measures under uncertainty: Examining the role of network topology. *Comput. Math. Organ. Theory* 15 (2009), 303–328.
[15] Kasimir Gabert, Ali Pinar, and Umit V Catalyurek. 2021. Shared-memory scalable k-core maintenance on dynamic graphs and hypergraphs. In *IPDPSW*. 998–1007.
[16] Sen Gao, Hongchao Qin, Rong-Hua Li, and Bingsheng He. 2023. Parallel colorful h-Star core maintenance in dynamic graphs. *Proc. VLDB Endow.* 16, 10 (2023), 2538–2550.
[17] Joy Ghosh, Hung Q Ngo, Seokhoon Yoon, and Chunming Qiao. 2007. On a routing problem within probabilistic graphs and its application to intermittently connected networks. In *INFOCOM*. 1721–1729.
[18] Bin Guo and Emil Sekerinski. 2023. Parallel order-based core maintenance in dynamic graphs. In *ICPP*. 122–131.
[19] Qiang-Sheng Hua, Yuliang Shi, Dongxiao Yu, Hai Jin, Jiguo Yu, Zhipen Cai, Xiuzhen Cheng, and Hanhua Chen. 2019. Faster parallel core maintenance algorithms in dynamic graphs. *IEEE Trans. Par. Distrib. Syst.* 31, 6 (2019), 1287–1300.
[20] Qiang-Sheng Hua, Xiaohui Zhang, Hai Jin, and Hong Huang. 2023. Revisiting core maintenance for dynamic hypergraphs. *IEEE Trans. Par. Distrib. Syst.* 34, 3 (2023), 981–994.
[21] Xin Huang, Wei Lu, and Laks VS Lakshmanan. 2016. Truss decomposition of probabilistic graphs: Semantics and algorithms. In *SIGMOD*. 77–90.
[22] Hai Jin, Na Wang, Dongxiao Yu, Qiang-Sheng Hua, Xuanhua Shi, and Xia Xie. 2018. Core maintenance in dynamic graphs: A parallel approach based on matching. *IEEE Trans. Par. Distrib. Syst.* 29, 11 (2018), 2416–2428.
[23] Haruko Kawahigashi, Yoshiaki Terashima, Naoto Miyauchi, and Tetsuo Nakakawaji. 2005. Modeling ad hoc sensor networks using random graph theory. In *CCNC*. 104–109.
[24] Lingli Li, Hongzhi Wang, Jianzhong Li, and Hong Gao. 2020. A survey of uncertain data management. *Front. Comput. Sci.* 14 (2020), 162–190.
[25] Min Li, Fangxiang Wu, Yi Pan, and Jianxin Wang. 2014. Detecting protein complexes based on uncertain graph model. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 01 (2014), 1–1.
[26] 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.
[27] Rong-Hua Li, Jeffrey Xu Yu, and Rui Mao. 2013. Efficient core maintenance in large dynamic graphs. *IEEE Trans. Knowl. Data Eng.* 26, 10 (2013), 2453–2465.
[28] David Liben-Nowell and Jon Kleinberg. 2003. The link prediction problem for social networks. In *CIKM*. 556–559.
[29] Zhe Lin, Fan Zhang, Xuemin Lin, Wenjie Zhang, and Zhihong Tian. 2021. Hierarchical core maintenance on large dynamic graphs. *Proc. VLDB Endow.* 14, 5 (2021), 757–770.
[30] Boge Liu, Long Yuan, Xuemin Lin, Lu Qin, Wenjie Zhang, and Jingren Zhou. 2020. Efficient $(\alpha, \beta)$-core computation in bipartite graphs. *The VLDB Journal* 29, 5 (2020), 1075–1099.
[31] Qing Liu, Xuliang Zhu, Xin Huang, and Jianliang Xu. 2021. Local algorithms for distance-generalized core decomposition over large dynamic graphs. *Proc. VLDB Endow.* 14, 9 (2021), 1531–1543.
[32] Feng Luo, Bo Li, Xiu-Feng Wan, and Richard H Scheuermann. 2009. Core and periphery structures in protein interaction networks. *BMC Bioinform.* 10, Suppl 4 (2009), S8.
[33] Qi Luo, Dongxiao Yu, Zhipeng Cai, Yanwei Zheng, Xiuzhen Cheng, and Xuemin Lin. 2023. Core maintenance for hypergraph streams. *WWW* 26, 5 (2023), 3709–3733.
[34] Wensheng Luo, Qiaoyuan Yang, Yixiang Fang, and Xu Zhou. 2023. Efficient core maintenance in large bipartite graphs. *PACM Mngt. Data* 1, 3 (2023), 1–26.
[35] Wensheng Luo, Xu Zhou, Kenli Li, Yunjun Gao, and Keqin Li. 2023. Efficient influential community search in large uncertain graphs. *IEEE Trans. Knowl. Data Eng.* 35, 4 (2023), 3779–3793.
[36] Yasir Mehmood, Francesco Bonchi, and David García-Soriano. 2016. Spheres of influence for more effective viral marketing. In *SIGMOD*. 711–726.
[37] Suyu Mei. 2022. A framework combines supervised learning and dense subgraphs discovery to predict protein complexes. *Front. Comput. Sci.* 16, 1 (2022), 161901.
[38] HW Mewes, D Frishman, KFX Mayer, M Munsterkotter, O Noubibou, P Pagel, T Rattei, M Oesterheld, A Ruepp, and V Stumpflen. 2006. MIPS: analysis and annotation of proteins from whole genomes in 2005. *Nucleic Acids Res.* 34, Suppl. 1 (2006), D169–D169.
[39] Xiaoye Miao, Yue Liu, Lu Chen, Yunjun Gao, and Jianwei Yin. 2022. Reliable community search on uncertain graphs. In *ICDE*. 1166–1179.
[40] Arko Provo Mukherjee, Pan Xu, and Srikanta Tirthapura. 2015. Mining maximal cliques from an uncertain graph. In *ICDE*. 243–254.
[41] Arko Provo Mukherjee, Pan Xu, and Srikanta Tirthapura. 2016. Enumeration of maximal cliques from an uncertain graph. *IEEE Trans. Knowl. Data Eng.* 29, 3 (2016), 543–555.
[42] You Peng, Ying Zhang, Wenjie Zhang, Xuemin Lin, and Lu Qin. 2018. Efficient probabilistic k-core computation on uncertain graphs. In *ICDE*. 1192–1203.
[43] Ganesan Ramalingam and Thomas Reps. 1996. On the computational complexity of dynamic graph problems. *Theor. Comput. Sci.* 158, 1-2 (1996), 233–277.
[44] Christopher Ré, Nilesh N Dalvi, and Dan Suciu. 2006. Query evaluation on probabilistic databases. *IEEE Data Eng. Bull.* 29, 1 (2006), 25–31.
[45] Ahmet Erdem Sarıyüce, Buğra Gedik, Gabriela Jacques-Silva, Kun-Lung Wu, and Ümit V Çatalyürek. 2016. Incremental k-core decomposition: algorithms and evaluation. *The VLDB Journal* 25 (2016), 425–447.
[46] Damien Seux, Fragkiskos Malliaros, Apostolos N Papadopoulos, and Michalis Vazirgiannis. 2017. Core decomposition of uncertain graphs using representative instances. In *Complex Networks*.
[47] Na Wang, Dongxiao Yu, Hai Jin, Chen Qian, Xia Xie, and Qiang-Sheng Hua. 2017. Parallel algorithm for core maintenance in dynamic graphs. In *ICDCS*. 2366–2371.
[48] Bo Wei, Jie Liu, Daijun Wei, Cai Gao, and Yong Deng. 2015. Weighted k-shell decomposition for complex networks based on potential edge weights. *Phys. A* 420 (2015), 277–283.
[49] Bai Wen, Yadi Chen, Di Wu, Huang Zhichuan, Yipeng Zhou, and Chuan Xu. 2022. Generalized core maintenance of dynamic bipartite graphs. *DMKD* 36, 1 (2022), 209–239.
[50] Dong Wen, Bohua Yang, Lu Qin, Ying Zhang, Lijun Chang, and Rong-Hua Li. 2020. Computing k-cores in large uncertain graphs: An index-based optimal approach. *IEEE Trans. Knowl. Data Eng.* 34, 7 (2020), 3126–3138.
[51] Tongfeng Weng, Xu Zhou, Kenli Li, Peng Peng, and Keqin Li. 2021. Efficient distributed approaches to core maintenance on large dynamic graphs. *IEEE Trans. Par. Distrib. Syst.* 33, 1 (2021), 129–143.
[52] Xiaoqun Wu, Wenbin Wei, Longkun Tang, Jinhu Lü, et al. 2019. Coreness and $h$-Index for weighted networks. *IEEE Trans. Circuits Syst. I* 66, 8 (2019), 3113–3122.
[53] 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.
[54] Zhiyang Yuan, Wenguang Zheng, Peilin Yang, Qingbo Hao, and Yingyuan Xiao. 2023. Evolving interest with feature co-action network for CTR prediction. *Data Sci. Eng.* 8, 4 (2023), 344–356.
[55] Feiteng Zhang, Bin Liu, Zhenming Liu, and Qizhi Fang. 2023. Order based algorithms for the core maintenance problem on edge-weighted graphs. *Theor. Comput. Sci.* 941 (2023), 140–155.
[56] Yikai Zhang and Jeffrey Xu Yu. 2019. Unboundedness and efficiency of truss maintenance in evolving graphs. In *SIGMOD*. 1024–1041.
[57] Yikai Zhang, Jeffrey Xu Yu, Ying Zhang, and Lu Qin. 2017. A fast order-based approach for core maintenance. In *ICDE*. 337–348.
[58] Gengda Zhao, Kai Wang, Wenjie Zhang, Xuemin Lin, Ying Zhang, and Yizhang He. 2022. Efficient computation of cohesive subgraphs in uncertain bipartite graphs. In *ICDE*. 2333–2345.

[59] Alexander Zhou, Yue Wang, and Lei Chen. 2023. Butterfly counting and bitruss decomposition on uncertain bipartite graphs. *The VLDB Journal* 32, 5 (2023), 1013–1036.
[60] Wei Zhou, Hong Huang, Qiang-Sheng Hua, Dongxiao Yu, Hai Jin, and Xiaoming Fu. 2021. Core decomposition and maintenance in weighted graph. *WWW* 24 (2021), 541–561.
[61] Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. 2010. Finding top-k maximal cliques in an uncertain graph. In *ICDE*. 649–652.
[62] Zhaonian Zou and Rong Zhu. 2017. Truss decomposition of uncertain graphs. *Knowl. Inf. Syst.* 50 (2017), 197–230.

# APPENDIX

## A THE PROOF OF THEOREM 4.2

We prove it by contradiction. Assume that the updated vertices do not form a connected subgraph, thus leading to the presence of two or more separate subgraphs. Now, we consider two separate subgraphs, denoted as $S_1$ and $S_2$. Since there is only one edge insertion, only one of these subgraphs, say $S_1$, can contain a vertex that acquires a new neighbor in $\mathcal{G}$. In contrast, none of the vertices in $S_2$ undergo modifications in their adjacency. This is a contradiction. If a vertex has its $\eta$-threshold increased, it must have either gained a new neighbor or at least one of its existing neighbors who has experienced an increase in $\eta$-threshold. Applying this recursively, we must find a vertex whose increased $\eta$-threshold stem form the addition of a new neighbor. However, for $S_2$, such a vertex does not exist. This also constitutes a contradiction.

## B THE PROOF OF THEOREM 4.5

Proof. We prove Theorem 4.5 from two distinct cases. (1) If $k \in [1, k_0]$, an edge or a vertex is deleted from the initial $(k, \eta(k, w))$-core, causing a change in $\eta(k, w)$. (2) If $k \notin [1, k_0]$, neither vertices nor edges are subjected to removal from the pre-existing $(k, \eta(k, w))$-core, thereby $\eta(k, w)$ remains unchanged. □

## C THE PROOF OF THEOREM 4.6

We prove it by contradiction. Assume that the updated vertices do not form a connected subgraph, thus leading to the presence of two or more separate subgraphs. Now, we consider two separate subgraphs, denoted as $S_1$ and $S_2$. Since there is only one edge deletion, it follows that merely one of the subgraphs, say $S_1$, can comprise a vertex who loss an old neighbor in $\mathcal{G}$. In contrast, none of the vertices in $S_2$ undergo modifications in their adjacency. This is a contradiction. If a vertex has its $\eta$-threshold decreased, it must have either lost a neighbor or at least one of its existing neighbors whose $\eta$-threshold is also decreased. By applying this recursively, it should be possible to trace back to a vertex from which the decrease in its lower $\eta$-threshold can be attributed to the loss of an old neighbor. However, in the case of $S_2$, such a vertex does not exist, leading to a contradiction.

## D THE FINDDELETE ALGORITHM

Similarly, **FindDelete** performs a BFS traversal to identify all candidates reachable from the root through vertices within the $\eta$-threshold range. The detailed pseudocode is given in Algorithm 6. After the deletion of an edge, there may be multiple roots, denoted as $V_c$ and $r$. If $V_c$ is not empty, the core number of a vertex $u$ in $V_c$ is reduced by 1, and we should delete the corresponding $\eta_k(u)$ (line 4). The neighbors directly affected by $u$, whose $\eta$-thresholds belong to the range $(lb, ub]$, are added to the queue $Q$ (lines 5-7). In addition, if a root in the array $r$ has not been added to $Q$, it indicates a high probability of disconnection from the subgraph induced by $V_c$. Therefore, the root is directly added to $Q$ (lines 8-9). The initialization of $Q$ is now complete. The subsequent steps follow the same approach as Algorithm 2. In particular, in the update theory in Theorem 4.8, the $\eta$-threshold range contains an upper-bound but not a lower-bound. Therefore, the corresponding judgment conditions need to be modified (line 10).

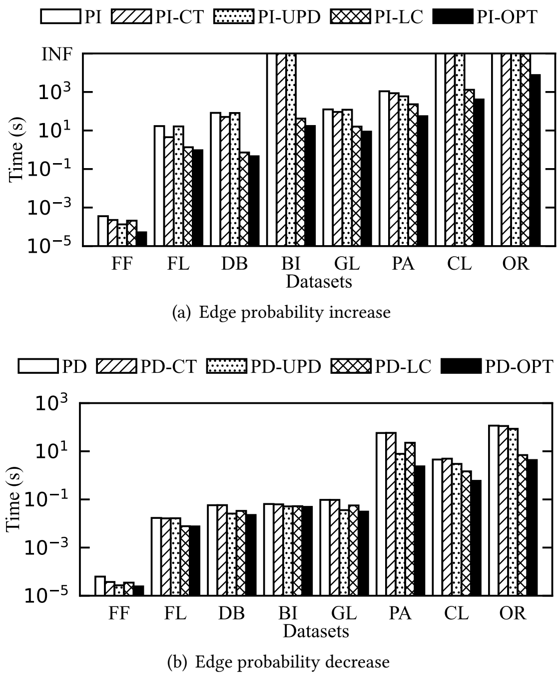
**Algorithm 6: FindDelete**

**Input:** updated graph: $\mathcal{G}$, set: $V_c$, the root set: $r$, low-bound: $lb$, upper-bound: $ub$, $k$-value: $k$
**Output:** candidate set $C$

1 initialize $C$, $visited[]$, $Q$ as lines 1-2 in Algorithm 2;
2 **if** $V_c \neq \varnothing$ **then**
3     **foreach** $u \in V_c$ **do**
4         delete $\eta(k, u)$;
5         **foreach** $(u, v) \in E$ **do**
6             **if** $\eta(k, v) \in (lb, ub] \wedge v \notin V_c \wedge \neg visited[v]$ **then**
7                 $Q$.push($v$); $visited[v] \leftarrow$ **true**;
8 **foreach** $u \in r \wedge \neg visited[u]$ **do**
9     $Q$.push($u$); $visited[u] \leftarrow$ **true**;
10 run lines 3-7 in Algorithm 2 by modifying the judgment symbols $\geq$ and $<$ to $>$ and $\leq$ respectively;
11 **return** $C$ ;

## E COMPLEXITY ANALYSIS IN SUBSECTION 4.3

Given an integer $k \in \mathcal{R}^+/\mathcal{R}^-$ and $\eta(k, u) < \eta(k, v)$, Algorithm 1 and Algorithm 4 have the following time complexities. Firstly, the DP algorithm for calculating the $k$-probability of a given vertex $u$ requires $O(k \cdot deg(u))$ time. Secondly, Algorithm 2 takes $O(deg(u))$ time for each $u \in C$ to identify all candidates. Thirdly, Algorithm 3 uses a minimum priority queue to maintain all vertices in $C$, where their $k$-probabilities serve as keys. It totally takes $O(|C| \cdot \log |C|)$ time to remove the vertex with the minimum $k$-probability. Consequently, the total calculation and update times are $O(\sum_{u \in C} k \cdot deg(u))$ and $O(\sum_{u \in C} deg(u) \cdot k \cdot deg(u))$ respectively. In the worst scenario, the time complexity of the above three steps is $O(|C| \cdot \log |C| + \sum_{u \in C} k \cdot deg^2(u))$. In addition, the core maintenance algorithm for deterministic graph has a worst-case time complexity of $O(|V_c| \cdot \sum_{u \in V_c} deg(u))$. It should be noted that $V_c \subseteq C$ must be true here. Therefore, considering each $k$, the total time complexity of Algorithm 1 or Algorithm 4 is $O(k_0 \cdot (|C| \log |C| + |V_c| \sum_{u \in V_c} deg(u)) + k_0^2 \cdot \sum_{u \in C} deg^2(u))$. Here, $V_c$ is bounded by $C$ and $k_0$ is bounded by the minimum core number of $u$ and $v$. Also, the size of $C$ is bounded by the maximum subgraph. Each vertex $w$ in the subgraph are connected to the root, and its $\eta(k, w)$ within the $\eta$-threshold range. Therefore, in practice, the maintenance algorithm is very fast because the visited subgraph during the search process is usually much smaller than $\mathcal{G}$, and $k_0$ is usually smaller than the largest core number in the deterministic graph.




(a) Edge probability increase



(b) Edge probability decrease

**Figure 8: Evaluation of optimizations in PI/PD**

# F THE EXPERIMENTAL RESULTS OF EI/ED

Figure 8 depicts the performance of three optimization techniques (-CT, -UPD, -LC ) on PI and PD. These optimizations are integrated into the basic core maintenance algorithms (PI and PD) and the execution times of PT-OPT and PD-OPT are reported. In Figure 8(a), the three optimizations yield similar performance improvement on the basic algorithm, as shown in edge insertion. The key distinction is that PI does not incorporate critical $\mathcal{R}^+$ processing, potentially reducing its time cost. For example, for FF, DB, and GL datasets, PI reduces the average insertion time of EI by 3.2, 7.1, and 1.3 times, respectively. Furthermore, as increasing the probability does not alter the core number, each $k$ corresponds to the for loop. Conversely, when processing $k_0$ in EI (lines 8-18 in Algorithm 1), it is possible to encounter case-2 (lines 11-13). This can result in a super small candidate subgraph of $V_C$, resulting in a potential increase in the time cost of PI. For example, for FF and CP datasets, the average time of PI is 1.6 and 1.3 times higher than that of EI. Both cases can be explained accordingly. Figure 8(b) presents the time taken for a probability decrease. The three optimizations yield similar performance improvements, as observed in edge deletion, across most datasets. Similarly, PD only consists of the for loop, reducing its time cost. The main difference is that PD must execute the for loop at each processing of $k_0$. This increases the performance advantage of PD-LC over PD, especially on large datasets like CL and CO.